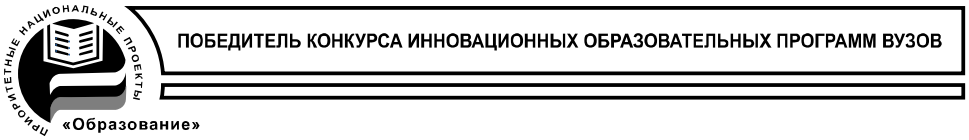
МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ



НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК

Выпуск 39

ИССЛЕДОВАНИЯ В ОБЛАСТИ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Труды молодых ученых

САНКТ-ПЕТЕРБУРГ
2007

Выпуск содержит материалы **IV межвузовской конференции молодых ученых**., организованной 10–13 апреля 2007 года Санкт-Петербургским государственным университетом информационных технологий, механики и оптики в сотрудничестве с
Балтийским государственным техническим университетом «Военмех»
Башкирским государственным университетом
Белорусским государственным педагогическим университетом им. Максима Танка
Белорусским государственным технологическим университетом
Белорусским государственным университетом информатики и радиоэлектроники
Дальневосточным государственным университетом
Дальневосточной академией государственной службы
Институтом аналитического приборостроения Российской Академии Наук (РАН)
Институтом Солнечно-Земной Физики СО РАН
Институтом химии высокочистых веществ РАН (г. Нижний Новгород)
Казанским государственным техническим университетом им. А.Н. Туполева
Казанским государственным университетом
Карельским государственным педагогическим университетом
Костромским государственным технологическим университетом
Красноярским государственным техническим университетом
Ленинградским государственным университетом им. А.С. Пушкина
Магнитогорским государственным техническим университетом им. Г.И. Носова
Морской государственной академией им. адмирала Ф.Ф. Ушакова
Московским государственным институтом электронной техники (техническим университетом)
Московским государственным техническим университетом им. Н.Э. Баумана
Московским педагогическим государственным университетом
Муромским институтом Владимирского государственного университета
Петербургским государственным университетом путей сообщения
Пятигорским государственным лингвистическим университетом
Российским государственным гидрометеорологическим университетом
Самарским государственным архитектурно-строительным университетом
Санкт-Петербургским государственным горным институтом им. Г.В. Плеханова (техническим университетом)
Санкт-Петербургским государственным инженерно-экономическим университетом (ИНЖЭКОН)
Санкт-Петербургским государственным политехническим университетом
Санкт-Петербургским государственным университетом
Санкт-Петербургским государственным университетом аэрокосмического приборостроения
Санкт-Петербургским институтом машиностроения (ЛМЗ-ВТУЗ)
Санкт-Петербургским университетом кино и телевидения
Санкт-Петербургской государственной академией физической культуры им. П.Ф. Лесгафта
Санкт-Петербургской государственной педиатрической медицинской академией
Северо-Западной академией государственной службы
Северо-Осетинским государственным университетом им. К.Л. Хетагурова
Тамбовским государственным университетом им. Г.Р. Державина
Татарским государственным гуманитарно-педагогическим университетом
Университетом Aix-Marseille II (Франция)
Университетом Прованса (Франция)
ФГУП ″ЦНИИ им. академика А.Н. Крылова″
Энгельсским технологическим институтом Саратовского государственного технического университета

В выпуске представлены работы, поддержанные финансированием в рамках:

– аналитической ведомственной целевой программы «Развитие научного потенциала высшей школы (2006–2008 гг.) (Федеральное агентство по образованию);

– Федеральной целевой программы развития образования на 2006–2010 гг. (Федеральное агентство по образованию);

– Российского фонда фундаментальных исследований,

а также инициативные разработки.

ПРОГРАММНЫЙ КОМИТЕТ КОНФЕРЕНЦИИ

Председатель – ректор СПбГУ ИТМО, д.т.н., профессор **В.Н. Васильев**
Сопредседатели – проректор по развитию, д.т.н., профессор **В.О. Никифоров,**
проректор по УО и АР, д.ф.-м.н., профессор **Ю.Л. Колесников,**
проректор по УМР, к.т.н., профессор **А.А. Шехонин,**
декан факультета ППО, д.т.н., профессор **В.Л. Ткалич**
Члены программного комитета – д.т.н., профессор **Ю.А. Гатчин**, д.т.н., профессор **В.М. Мусалимов**, д.т.н., профессор **С.Б. Смирнов**, д.т.н., профессор **В.А. Тарлыков**, д.т.н., профессор **Е.Б. Яковлев**, к.т.н. **Т.В. Точилина**

ОРГАНИЗАЦИОННЫЙ КОМИТЕТ КОНФЕРЕНЦИИ

Председатель – начальник НИЧ **Л.М. Студеникин**
Зам. председателя – к.т.н. **Т.В. Точилина**
Члены организационного комитета – П.А. Борисов, Н.Н. Валентик, И.Н. Жданов, С.Ю. Керпелева, Н.В. Когай, А.В. Козаченко, И.М. Кудрявцева, Д.В. Лукичёв, А.А. Малинин, Л.В. Можжухина, Ю.С. Монахов, Н.Б. Нечаева, М.В. Никитина, М.С. Петрищев, С.С. Резников, В.Н. Фролков

В 2007 году СПбГУ ИТМО стал победителем конкурса инновационных образовательных программ вузов России на 2007–2008 годы. Реализация инновационной образовательной программы «Инновационная система подготовки специалистов нового поколения в области информационных и оптических технологий» позволит выйти на качественно новый уровень подготовки выпускников и удовлетворить возрастающий спрос на специалистов в информационной, оптической и других высокотехнологичных отраслях экономики.

ISSN 1819-222X

1 ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ

РАЗРАБОТКА ПРОФИЛЕЙ ЗАЩИТЫ И ЗАДАНИЙ ПО БЕЗОПАСНОСТИ ДЛЯ СЕТЕЙ СВЯЗИ С ИСПОЛЬЗОВАНИЕМ МЕТОДА ОЦЕНКИ УРОВНЯ КРИТИЧНОСТИ СЕГМЕНТА

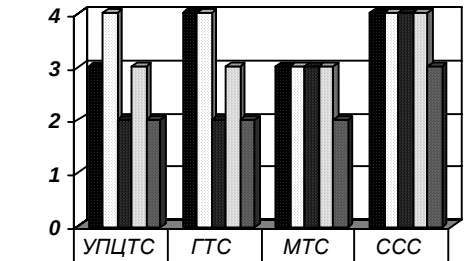
Н.А. Минакова
Научный руководитель – д.т.н., профессор Л.Г. Осовецкий

Рассмотрено влияние уровня критичности сегмента, определенного в ранее опубликованных материалах, на выбор функциональных требований безопасности и оценочных уровней доверия при формировании профилей защиты и заданий по безопасности для конкретного вида сети связи.

Введение

Формирование требований безопасности к объекту информатизации на основе ГОСТ Р ИСО/МЭК15408-2002 является одной из основных задач при разработки профилей защиты, заданий по безопасности и функциональных пактов. Проблема заключается в обосновании выбора. Метод оценки уровня критичности сегмента позволяет частично обосновать выбор ОУД, компонентов ФТБ и уровень аудита. В статье представлены зависимости между уровнем критичности сегмента и выбором требований по безопасности. Должна прослеживаться зависимость между объемом функциональных возможностей сегмента сети, принадлежащих ему угроз и требуемым объемом механизмов защиты для их устранения. Эта задача решается при создании функционального пакет ФТБ для вида сегмента. Детализация этого общего функционального пакета доходит до семейств, а уровень критичности сегмента позволяет определить иерархию конкретного компонента для обеспечения необходимого уровня защиты. Основной решаемой задачей является частичное обоснование выбора.

Уровень критичности сегмента и требования безопасности



	УПЦТС	ГТС	МТС	ССС
СУ	3	4	3	4
СК	4	4	3	4
СБ	2	2	3	4
СС	3	3	3	4
АУ	2	2	2	3

Рис. 1. Диаграмма уровней критичности сегментов сетей связи и систем коммутации: ССС – сотовые сети связи, УПТС – учережденческо-производственные сети, ГТС – городские телефонные сети, МТС – междугородние /международные телефонные сети, СУ – система управления, СК – система коммутации, СБ – система биллинга, СС – система сервисов, АУ – абонентские устройства [1]

Алгоритм работы метода оценки уровня критичности сегмента сети связи и системы коммутации (МОУКС) представлен в предыдущих статьях. Результаты работы МОУКС приведены на рис. 1, в которой отражен присвоенный уровень критичности каждому из 5-и сегментов 4-х видов сетей.

При разработки профилей защиты (ПЗ) и заданий по безопасности (ЗБ) уровень критичности сегмента (УКС) влияет на следующие их компоненты:

– ОУД (3 часть РД «БИТ»);

– ФТБ (иерархия компонента, уровень аудита) (2 часть РД «БИТ»).

Оценочные уровни доверия и уровень критичности сегмента

Анализ требований к оценочным уровням доверия (ОУД) из РД «БИТ» позволяет прямо сопоставить уровень критичности сегмента (УКС), как показано в табл. 1.

ОУД	Признак	УКС	Показатель
1	Угрозы безопасности не рассматривают как серьезные	**1**	$\{0;0,1552\}$
2	Отсутствии доступа к полной документации по разработке	**2**	$\{0,156;0,3104\}$
3	Всестороннее исследование ОО, без существенных затрат на изменение технологии проектирования	**3**	$\{0,311;0,4656\}$
4	Готовность нести дополнительные производственные затраты	**4**	$\{0,466;0,6208\}$
5	Для запланированной разработки	**5**	$\{0,621;0,776\}$

Таблица 1. Соотнесение УКС и ОУД

ОУД6 (оценка ценности защищаемых активов) не входит в задачи данной работы, ситуации высокого риска приравниваются к защите государственной тайны. Создаваемая модель рассчитана на защиту конфиденциальной информации. ОУД7 рассчитан на оценку СЗИ, а сети связи и системы коммутации не являются СЗИ.

Функциональные требования безопасности и уровень критичности сегмента

Формирование функциональных требований безопасности (ФТБ) – одна из основных задач при разработки профилей защиты (ПЗ) и заданий по безопасности (ЗБ).

При разработке ПЗ и ЗБ для сетей связи и систем коммутации рекомендуется создание пакетов ФТБ для каждого сегмента сети, так как нецелесообразно предъявлять одинаковый набор ФТБ для СУ и АУ в силу разных функциональных возможностей и проистекающих из них угроз.

Функциональные требования безопасности для сегмента определенного вида сети формируются с учетом:

1. сформированного общего пакета ФТБ для сегмента;
2. рекомендуемой иерархии компонента из выбранного семейства;
3. рекомендованного уровня аудита.

Функциональный пакет – предназначенная для многократного использования совокупность функциональных компонентов, объединенных для удовлетворения совокупности определенных целей безопасности

Для некоторых сетей по результатам работы МОУКС могут быть выданы рекомендации по исключению пакета ФТБ на какой-либо сегмент. Например, для МАТС – это сегмент абонентских устройств.

В данной модели создания профилей защиты для сетей связи и систем коммутации ФТБ пакета выбираются на уровне семейств без указания компонентов. Компонент определяется в зависимости от уровня критичности сегмента.

Функциональные семейства содержат один или несколько компонентов, каждый из которых может быть выбран для включения в ПЗ, ЗБ или функциональный пакет. Цель ранжирования компонентов – предоставить пользователям информации для выбора подходящего функционального компонента, если семейство идентифицировано пользователем как необходимая или полезная часть требований безопасности [2].

Далее перечисляются имеющиеся компоненты и приводится их логическое обоснование. Детализация компонента производится в описании каждого компонента [2].

Связи между компонентами в пределах функционального семейства могут быть иерархическими и неиерархическими. Компонент иерархичен (т.е. расположен выше по иерархии) по отношению к другому компоненту, если предлагает большую безопасность. Описание семейств содержит графическое представление иерархии компонентов [2].

На рис. 2 представлено графическое изображение иерархичности компонентов семейства.

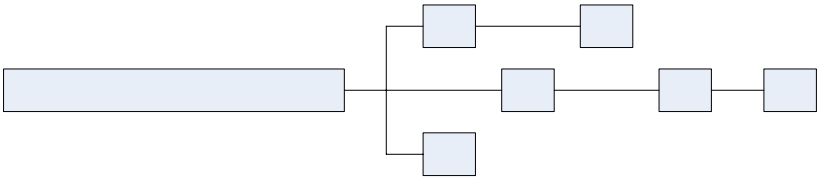


Рис. 2. Пример графического представления иерархии компонента из РД «БИТ»

ФТБ для сегмента определенного вида сети находится в зависимости от пакета ФТБ для вида сегмента, иерархии выбранного компонента и уровня аудита. Последние два находятся в зависимости от уровня критичности сегмента. Графически зависимости или алгоритм ПЗ и ЗБ представлены на рис. 3.

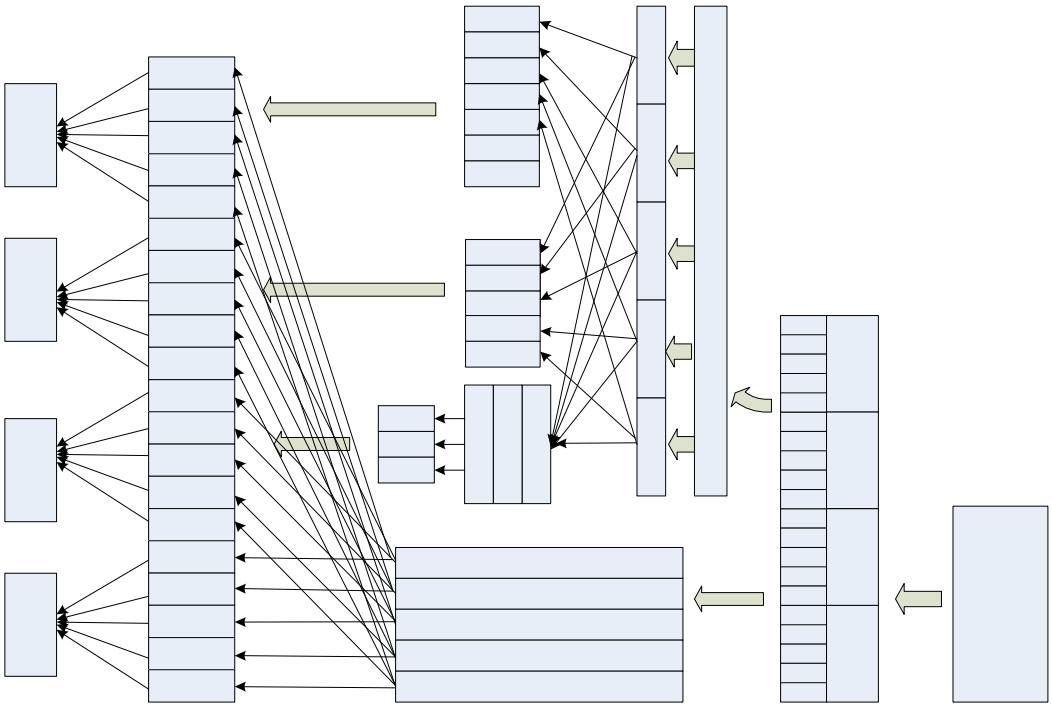


Рис. 3. Схема зависимостей при создании ПЗ для вида сети

Иерархия компонентов ФТБ и УКС. Выбор иерархии компонента ФТБ в зависимости от УКС проводился по заранее определенным ограничениям (см. табл. 2–6).

УКС	Показатель УКС	Иерархические компоненты				
		1	**2**	**3**	**…**	**n**
1	$\{0;0,1552\}$	1	1	1	…	N
2	$\{0,156;0,3104\}$	1	1	2	…	N
3	$\{0,311;0,4656\}$	1	2	2	…	N
4	$\{0,466;0,6208\}$	1	2	3	…	N
5	$\{0,621;0,776\}$	1	2	3	…	N

Таблица 2. Иерархия компонента и УКС

Ограничение 1: max/min.

Если существует иерархия компонентов, то

$I_{\max} = U_{\max}$;

$I_{\min} = U_{\min}$ (табл. 3).

КЛАСС/СЕМЕЙСТВО	УРОВНИ ИЕРАРХИИ КОМПОНЕНТОВ			МАХ КОЛ. КОМ	МАХ КОЛ. УР.	УРОВНИ КРИТИЧНОСТИ СЕГМЕНТОВ СЕТИ СВЯЗИ И СИСТЕМЫ КОММУТАЦИИ				
	1	**2**	**3**			**U1**	U2	U3	U4	**U5**
						1	1	2	2	**3**
РАЗДЕЛЕНИЕ ДОМЕНА (FPT_SEP)	1	2	3	3	3	**1**	1	2	2	**3**

Таблица 3. Пример действия ограничения 1

Если не существует иерархии компонента, то значение *i* = max укс = min укс (табл. 4).

КЛАСС/СЕМЕЙСТВО	УРОВНИ ИЕРАРХИИ КОМПОНЕНТОВ			МАХ КОЛ. КОМ	МАХ КОЛ. УР.	УРОВНИ КРИТИЧНОСТИ СЕГМЕНТОВ СЕТИ СВЯЗИ И СИСТЕМЫ КОММУТАЦИИ				
	1	**2**	**3**			**U1**	**U2**	**U3**	**U4**	**U5**
						1	**1**	**2**	**2**	**3**
ИМПОРТ ДАННЫХ ИЗ-ЗА ПРЕДЕЛОВ ДЕЙСТВИЯ ФБО (FDP_ITC)	1,2			**2**	**1**	1,2	1,2	1,2	1,2	1,2

Таблица 4. Пример действия ограничения 1

Ограничение 2: Минимизации требований.

Присвоение иерархии компонента (I) УКС происходит по принципу минимизации требований. Например, если иерархичных значений три, то

$I_{\max} = U_{\max}$;

$I_{\min} = U_{\min} u U_2$;

$I_{сред.} = U_3 u U_4$.

КЛАСС/СЕМЕЙСТВО	УРОВНИ ИЕРАРХИИ КОМПОНЕНТОВ			МАХ КОЛ. КОМ	МАХ КОЛ. УР.	УРОВНИ КРИТИЧНОСТИ СЕГМЕНТОВ СЕТИ СВЯЗИ И СИСТЕМЫ КОММУТАЦИИ				
	1	**2**	**3**			**U1**	**U2**	**U3**	**U4**	**U5**
						1	**1**	**2**	**2**	**3**
РАЗДЕЛЕНИЕ ДОМЕНА (FPT_SEP)	1	2	3	**3**	**3**	1	1	2	2	3

Таблица 5. Пример действия ограничения 2

Ограничение 3: Солидарность компонентов.

В случае, когда семейство состоит из иерархичных и неиерархичных компонентов, то на иерархичные компоненты распространяется ограничение 2, на неиерархичные – ограничение 1 (табл. 6).

КЛАСС/СЕМЕЙСТВО	УРОВНИ ИЕРАРХИИ КОМПОНЕНТОВ			MAX КОЛ. КОМ	MAX КОЛ. УР.	УРОВНИ КРИТИЧНОСТИ СЕГМЕНТОВ СЕТИ СВЯЗИ И СИСТЕМЫ КОММУТАЦИИ				
	1	**2**	**3**			**U1**	**U2**	**U3**	**U4**	**U5**
						1	**1**	**2**	**2**	**3**
НАДЕЖНОЕ ВОССТАНОВЛЕНИЕ (FPT_RCV)	1,4	2	3	**4**	**3**	1,4	1.4	2,4	2,4	3,4

Таблица 6. Пример действия ограничения 3

Уровень аудита ФТБ и УКС. Требования аудита содержат события, потенциально подвергаемые аудиту, для их отбора разработчиками ПЗ/ЗБ при условии включения в ПЗ/ЗБ требований из класса FAU «Аудит безопасности». Эти требования включают в себя события, относящиеся к безопасности, применительно к различным уровням детализации, поддерживаемым компонентами семейства FAU_GEN «Генерация данных аудита безопасности». Например, запись аудита какого-либо механизма безопасности может включать в себя на разных уровнях детализации действия, которые раскрываются в следующих терминах [2]:

- минимальный – успешное использование механизма защиты;
- базовый – любое использование механизма безопасности, а также информация о текущих значениях атрибутов безопасности;
- детализированный – любые изменения конфигурации механизма безопасности, включая параметры конфигурации до и после изменения.

Следует учесть, что категорирование событий, потенциально подвергаемых аудиту, всегда иерархично. Например, если выбрана базовая генерация данных аудита, то все события потенциально подвергаются аудиту, и поэтому входящие как в «минимальную», так и в «базовую» запись следует включить в ПЗ/ЗБ с помощью соответствующей операции назначения, за исключением случая, когда событие более высокого уровня имеет более высокий уровень детализации, чем событие более низкого уровня, и может просто заменить его. Когда желательна детализированная генерация данных аудита, все идентифицированные события, потенциально подвергаемые аудиту (для минимального, базового и детализированного уровней), следует включать в ПЗ/ЗБ [2].

Результаты сопоставления уровня аудита ФТБ и УКС с использованием приведенных ограничений представлены в табл. 7.

УКС	**Показатель**	**Аудит**
1	$\{0;0,1552\}$	Минимальный (1 из)
2	$\{0,156;0,3104\}$	Минимальный (все)
3	$\{0,311;0,4656\}$	Базовый (все)
4	$\{0,466;0,6208\}$	Детализированный (один из)
5	$\{0,621;0,776\}$	Детализированный (все)

Таблица 7. Уровень аудита ФТБ и УКС

Заключение

Решенная задача является частью модели создания ПЗ и ЗБ для сетей связи и систем коммутации. Показано, как критичность сегмента влияет на выбор ОУД, иерархии

компонентов ФТБ и уровня их аудита. Профиль защиты для сети связи собирается из функциональных пакетов ФТБ и ОУД, которые предварительно разрабатываются для вида сегмента сети связи, которых согласно модели пять.

На основании анализа требования ОУД, сегменты с 1-ой по 7-ой приравнены по уровню ОУД к УКС. Оставшиеся 6-ой и 7-ой отнесены к защите государственной тайны.

ФТБ для общего функционального пакета для вида сегмента необходимо определять на уровне семейств, а для определения компонента при включении функционального пакета в профиль защиты или задания по безопасности на вид сети обращаться к уровню критичности сегмента, который определяет степень иерархии компонента.

В случае использования семейства FAU_GEN «Генерация данных аудита безопасности» модель дает рекомендации для выбора уровня аудита в зависимости от уровня критичности сегмента.

В дальнейшем планируется пересмотр ранее созданного функционального пакета для системы управления с учетом положений, приведенных в статье. Для завершения работы планируется разработка общих функциональных пакетов для оставшихся четырех сегментов сетей связи: системы коммутации, системы биллинга, системы сервисов и абонентских устройств. Переработка их для включения в ПЗ на вид сети производится с учетом описанного в статье алгоритма и не должна занимать много времени.

Литература

1. Минакова Н.А. Метод определения уровня критичности сегмента (МОУКС) сетей связи и систем коммутации. // Теория и технология программирования и защиты информации. 2006 С. 39–42.
2. ГОСТ Р ИСО/МЭК15408-2002 Безопасные информационные технологии. Критерии оценки безопасности информационных технологий. / Федеральная служба по техническому и экспортному контролю. 2002 Ч. 1–3.

МАРКОВСКИЕ МОДЕЛИ СРЕДСТВ ЗАЩИТЫ АВТОМАТИЗИРОВАННЫХ СИСТЕМ СПЕЦИАЛЬНОГО НАЗНАЧЕНИЯ

К.В. Иванов (ОАО «АйСиЭл КПО ВС»)
Научный руководитель – д.т.н., профессор В.С. Моисеев
(Казанский государственный технический университет им. А.Н. Туполева)

В работе обосновывается применение терминов «информационное противоборство», «информационное оружие», обсуждаются характеристики образцов информационного оружия и строятся марковские модели функционирования коммутатора, межсетевого экрана, а также подсистемы защиты информации в целом. На основании построенных моделей в работе строится методика рассмотрения средств защиты с использованием теории марковских процессов.

Введение

В рамках перехода от индустриального к информационному обществу, жизнедеятельность которого зависит от процессов сбора, хранения, передачи и обработки гражданской и военной информации, становятся чрезвычайно актуальными вопросы надежной защиты информационной инфраструктуры государства, особенно его стратегической системы управления, а также оборонительных и ударных боевых систем от всех видов современного и перспективного информационного воздействия противника. В настоящее время такой вид воздействия отечественные и зарубежные специалисты связывают с применением в противоборстве государств информационного оружия [1–5].

Строгая военно-техническая проработка вопросов состава и применения информационного оружия с привлечением достижений современной информатики позволит успешно решать как проблему обороноспособности государства в возможных военных конфликтах XXI века, так и в рамках применения технологий двойного назначения проблему информационной безопасности в его мирном функционировании.

Вероятностный подход широко используется при построении моделей функционирования средств защиты информации, так как только вероятностные модели, которые строятся на основе существующей автоматизированной системы и системы защиты, на данный момент способны дать количественную характеристику уровня защищенности системы. В работах [6–9] функционирование подсистемы защиты рассматривается отдельно от функционирования автоматизированной системы управления специального назначения, или же затрагивает исключительно программное обеспечение [10]. Вместе с тем существуют работы [10], описывающие использование теории очередей для анализа проектирования компьютерных сетей. В силу изложенных причин весьма актуальным представляется построение вероятностных моделей функционирования защищенных АС в целом. Кроме того, модели не учитывают вероятность надежной работы системы, хотя надежность является важным показателем среди других параметров безопасности. В большинстве моделей используются методы теории графов, а также содержатся предпосылки использования при построении моделей безопасности теории марковских процессов, что позволяет нам производить оценку защищенности и надежности как отдельных образцов ИО, так и систем защиты в комплексе, используя методы этих дисциплин.

Таким образом, задача проработки вопросов построения образцов информационного оружия с использованием вероятностного подхода является весьма актуальной.

Целью данной работы является формализация понятия «информационное оружие» (далее – ИО), выделение его характеристик, разработка моделей ИО с использованием достижений теории графов, теории массового обслуживания (далее – ТМО) и теории марковских процессов, а также построение методики рассмотрения образцов ИО на примере существующих средств защиты.

Информационное оружие – средство ведения информационной борьбы

В работе [1] отмечается, что в переходный период к войнам нового поколения (примерно до 2010–2020 гг.) при сохранении многих элементов противоборства войн предыдущих поколений произойдет резкий скачок в процессе информатизации и автоматизации управления войсками и оружием. В дальнейшем такие системы будем называть автоматизированными системами управления специального назначения (далее – АСУ СН). Этот процесс обусловливается бурно развивающейся информатизацией всех сфер деятельности общества. В этой связи информационное противоборство, которое было присуще практически всем предыдущим военным конфликтам (добывание сведений о противнике, его дезинформация и т.п.) приобретает боевой характер и нуждается в более глубоком анализе его целей, задач и средств.

Следует отметить, что рассматриваемая здесь проблема информационного противоборства значительно шире и глубже проблемы обеспечения информационной безопасности, так как последняя в основном рассматривается в существующей литературе [4, 11 и др.] как совокупность методов и средств защиты информации от ее несанкционированного использования и ликвидации злоумышленниками. Именно такой узкий подход наряду с незавершенностью теории информационной безопасности общества объясняет успехи компьютерной преступности в нарушении конфиденциальности, целостности и доступности данных военных и гражданских систем различного уровня.

В отечественной литературе подробный анализ понятия «информационная война» приведен в работе [12]. В ходе анализа определений делается вывод, что принципиальной разницы между терминами «информационная война», «информационное противоборство» или «информационная борьба» нет.

Однако наиболее применимым для рассматриваемого вида противоборства является термин «информационная борьба», так как это понятие коррелирует с общеизвестным термином «радиоэлектронная борьба» (РЭБ), которая является, по сути, средством информационного противоборства в военных действиях. Этот вид борьбы является составной частью введенного понятия информационной борьбы. Более того, подразделения для ее ведения могут быть развернуты на основе существующих подразделений РЭБ с расширением их оснащения и функций.

В ходе рассмотрения документальных и научных источников по проблемам информационной борьбы выявляется существенное различие в подходах к сущности информационной борьбы как таковой: одни источники [12] на первые роли выводят гуманитарный, в первую очередь психологический компонент, другие же [13] делают акцент на техническом аспекте ведения информационных войн. Однако и те, и другие рассматривают исключительно ударные средства ведения информационной борьбы. Предлагаемый в существующих работах подход подразумевает пассивность оборонительной составляющей, что не позволяет повысить эффективность защиты собственной АСУ СН за счет активных действий против средств ИБ противника.

Термины «ударные» и «оборонительные» средства ИБ, введенные в работе [14], будем объединять понятием «информационное оружие». Под информационным оружием (ИО) будем понимать совокупность технических аппаратно-программных и программных средств ИБ оборонительного и ударного видов, с помощью которых осуществляется информационная борьба с соответствующей АСУ СН противника.

Любой образец ИО, как и любое сложное средство вооружения должен включать в себя две взаимодействующие компоненты:

- управляющая часть;
- исполнительная часть.

Назначением первой компоненты является управление исполнительной частью образца в процессе выполнения возложенных на него функций. Общая структура образца ИО приведена на рис. 1.

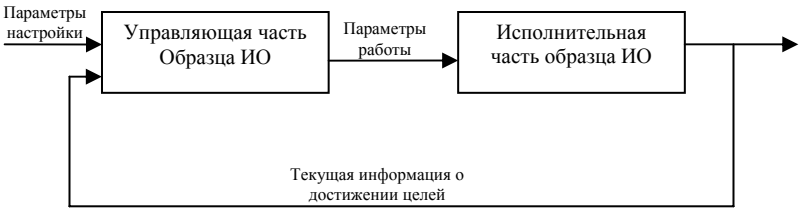


Рис. 1. Общая структура образца ИО

Образец ИО, функционирующий подобным образом, может быть рассмотрен в соответствии с моделью многодоступной вычислительной системы [15].

Рассмотрение модели ведется при помощи методов ТМО. Так как до сих пор мы рассматривали образец ИО в самом общем виде, необходима дальнейшая конкретизация, что влечет за собой рассмотрение отдельных классов и образцов ИО и, как следствие, построение новых, более детализированных моделей образцов ИО. Вместе с тем принципы [14], полагаемые в основу построения образцов ИО, налагают свои ограничения на совокупность характеристик образцов ИО.

Исходя из вышесказанного, для образцов ИО актуальными являются следующие группы характеристик:

- временные характеристики;
- емкостные характеристики;
- вероятностные характеристики.

Также необходимо принять во внимание, что ближайшим аналогом рассматриваемой нами в рамках данной работы АСУ СН является компьютерная сеть передачи данных на основе технологии Ethernet. Как показывает анализ литературы [10], методы ТМО в настоящее время находят широкое применение при проектировании, моделировании, а также анализе производительности, надежности и безопасности компьютерных сетей и автоматизированных систем в целом.

Рассмотрим подробнее некоторые компоненты компьютерной сети в качестве образцов ИО.

Марковская модель коммутатора

Коммутатор – активное устройство, осуществляющее процесс коммутации. В рамках настоящей работы будем считать, что коммутация – это процесс, при реализации которого организуется замкнутая линия связи.

Существующие уровни коммутации в рамках модели OSI рассмотрены в работе [16], где выделяется коммутация на 1–4 уровнях модели OSI. Вместе с тем из определения коммутации следует, что на каждом уровне коммутация может осуществляться независимо. Единственным условием является наличие линии связи на нижележащих уровнях. Таким образом, коммутация осуществляется на каждом уровне независимо, и для моделирования этого процесса достаточно рассмотреть коммутацию на любом уровне. В дальнейшем ограничимся рассмотрением процесса коммутации в традиционном понимании этого термина – на примере коммутаторов канального уровня модели OSI.

Использование ТМО для моделирования работы коммутаторов достаточно широко распространено [17, 18]. Однако рассматриваемые в этих работах модели коммутаторов позволяют получить либо емкостные (размеры буферов) [18], либо временные

характеристики коммутаторов [17]. Построим модель, позволяющую получить и емкостные, и временные характеристики.

Пусть потоки пакетов, поступающих на входы *N* портов коммутатора, распределены по экспоненциальному закону. Такой поток, называемый также стационарным пуассоновским, создает наиболее тяжелый режим работы для системы [19], и применение допущения о простейшем входном потоке позволяет получать предельные значения входных характеристик. Из этого следует, что если входной поток в реальной системе отличен от простейшего, то система будет иметь характеристики функционирования, по крайней мере, не хуже, чем при простейшем входном потоке.

Порт состоит из 2 трактов – приемного и передающего, которые параллельно обрабатывают поступающие на них пакеты. Кадры попадают на один из портов коммутатора с интенсивностью λ, где записываются во входной буфер. Далее они пересылается в выходной буфер соответствующего порта, а оттуда – в сеть. Попадание в каждый передающий тракт равновероятно. В этом случае модель в целом может быть представлена как две совокупности СМО типа M/M/1/L. Таким образом, процесс обработки пакета можно представить как двухфазную систему (рис. 2).

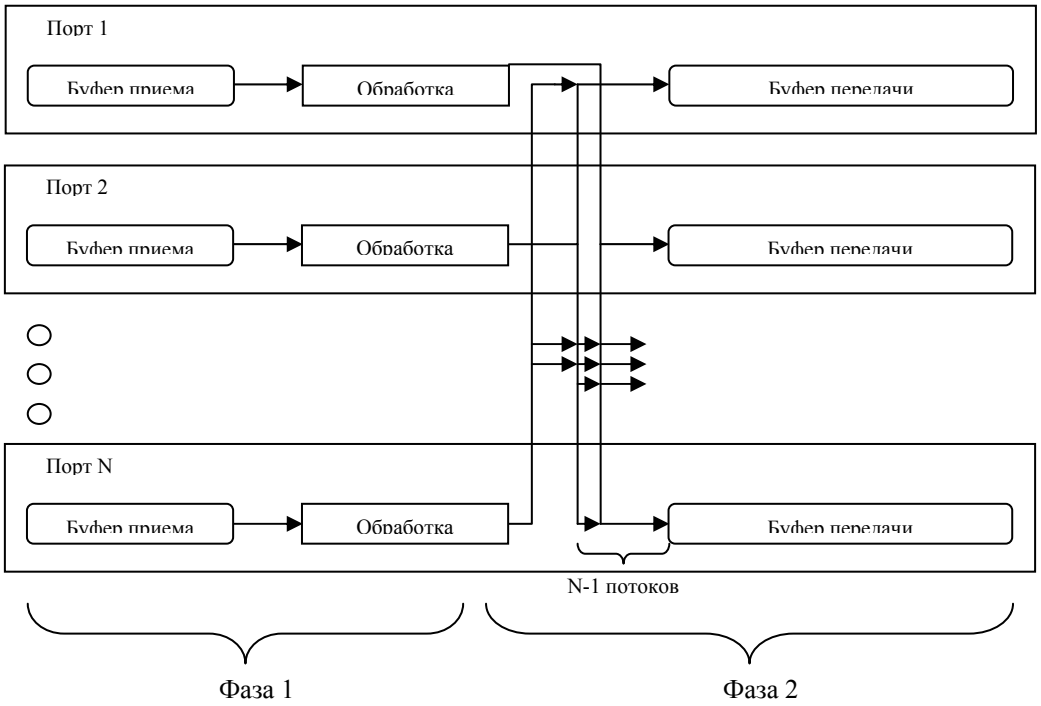


Рис. 2. Двухфазная система обработки пакета

Так как пакеты поступают на каждый порт независимо, а все рассматриваемые порты одинаковы, достаточно рассмотреть функционирование 1 порта коммутатора. Опуская промежуточные расчеты, приведем конечные формулы для искомых характеристик.

Средняя длина очереди может быть рассчитана, исходя из оценок для системы M/M/1, приведенных в работе [10]: $L = \frac{\rho^2}{1-\rho}$.

Емкостные, вероятностные и временные характеристики каждого порта можно рассчитать как сумму соответствующих характеристик для фазы приема и передачи.

Вероятность потери пакета $P_{ПОТ} = P_{0пр}\rho_{пр}{}^{L_{пр}+1} + P_{0пер}\rho_{пер}{}^{L_{пер}+1}$.

Среднее время пребывания пакета в коммутаторе равно

$$\overline{T}_{ПР} = \frac{1}{\mu_{пр}} + P_{0пр}\rho_{пр}{}^{2}\frac{1-(L_{пр}+2)\rho_{пр}{}^{L_{пр}+1}+(L_{пр}+1)\rho_{пр}{}^{L_{пр}+2}}{\lambda_{пр}(1-\rho_{пр})^{2}} +$$

$$\frac{1}{\mu_{пер}} + P_{0пер}\rho_{пер}{}^{2}\frac{1-(L_{пер}+2)\rho_{пер}{}^{L_{пер}+1}+(L_{пер}+1)\rho_{пер}{}^{L_{пер}+2}}{\lambda_{пер}(1-\rho_{пер})^{2}},$$

где $\lambda_{пр}$ – интенсивность потока, поступающего в приемный тракт порта, $\mu_{пр}$ – интенсивность его обработки, $\rho_{пр}=\lambda_{пр}/\mu_{пр}$ – коэффициент загрузки порта на фазе приема пакетов, $L_{пр}$ – емкость буфера (длина очереди) приема, $\lambda_{пер}$ – интенсивность потока, поступающего в передающий тракт порта, $\mu_{пер}$ – интенсивность его обработки, $\rho_{пер}=\lambda_{пер}/\mu_{пер}$ – коэффициент загрузки порта на фазе передачи пакетов, $L_{пер}$ – емкость буфера (длина очереди) передачи, P_0 – вероятность отсутствия в системе заявок.

Так как интенсивность передачи на каждый порт коммутатора принимается одинаковой, то поток пакетов, передаваемых с *i*-го порта, в общем случае имеет интенсивность $\lambda_{i\,пер} = \sum_{j=1}^{n-1}\lambda P_j$, где $i \neq j$, P_j – вероятность того, что пакеты, поступившие на *j*-й порт, имеют адрес назначения на *i*-м порту.

Дополнительно, в случае равновероятного попадания пакетов во все порты, вероятность нахождения адреса назначения пакета в *i*-м порту с учетом того, что с равной вероятностью они могут быть отброшены, может быть записана как $P=1/n$. Соответственно, $\lambda_{i\,пер} = \frac{\lambda(n-1)}{n}$.

Итак, рассмотрение коммутатора в виде многофазной СМО позволяет получить искомые характеристики, однако не отражает зависимость вероятностных характеристик работы от времени. Определим такую зависимость. Каждый порт представляет собой совокупность двух дисциплин обслуживания: дисциплина обслуживания принимаемых сообщений и дисциплина обслуживания приема и передачи. СМО может иметь следующие состояния:

- S_0 – на входе нет пакетов;
- S_1 – в СМО обрабатывается пакет;
- S_2 – в СМО обрабатывается 1 пакет, при этом в буфере ожидания находится 1 пакет;
- S_{L+1} – в СМО обрабатывается пакет, в очереди находится L пакетов;
- S_{L+2} – буфер переполнен, система отбрасывает пакеты.

Таким образом, система представляет собой классическую марковскую цепь с интенсивностями переходов из состояния в состояние μ и λ соответственно.

Связь между вероятностями нахождения системы во всех его возможных состояниях $p_i(t)$ выражается системой дифференциальных уравнений Колмогорова. Используем правила построения этих уравнений [20]: в левой части каждого уравнения записывается производная вероятности нахождения системы в рассматриваемом состоянии (вершине графа) $\dot{p}_i(t)$, а правая часть содержит столько членов, сколько ребер графа состояний связано с данной вершиной графа. Если ребро направлено из данной вершины, соответствующий член имеет знак «минус», если в вершину – знак «плюс». Каждый член равен произведению параметра (интенсивности) потока отказов (λ) или восстановлений, связанного с данным ребром, на вероятность нахождения в той вершине графа из которой исходит ребро $p_i(t)$.

Таким образом, необходимо решить систему дифференциальных уравнений:

$$\dot{p}_0(t) = -\lambda \cdot p_0(t) + \mu \cdot p_1(t),$$

$$\dot{p}_1(t) = \lambda \cdot p_0(t) - (\mu+\lambda)\cdot p_1(t) + \mu \cdot p_2(t),$$

$$\dot{p}_2(t) = \lambda \cdot p_1(t) - (\mu + \lambda) \cdot p_2(t) + \mu \cdot p_3(t),$$
$$\dot{p}_i(t) = \lambda \cdot p_{i-1}(t) - (\mu + \lambda) \cdot p_i(t) + \mu \cdot p_{i+1}(t),$$
$$\dot{p}_{L+2}(t) = \lambda \cdot p_{L+1}(t) - \mu \cdot p_{L+2}(t).$$

Решение данной системы позволяет получить функциональную зависимость вероятностных характеристик системы от времени.

Из вышеприведенных выкладок следует, что исследуемые характеристики весьма существенно зависят от интенсивности входного потока и интенсивности обработки кадров. Если интенсивность обработки кадров напрямую зависит от технологий, применяемых при синтезе коммутаторов, то интенсивность входного потока может меняться в широких пределах. Обсудим состав входного потока. Для передачи данных в сетях Ethernet в настоящее время используются следующие стандарты:

- Ethernet (10 Мбит/с). Максимальная пропускная способность сегмента Ethernet составляет 14880 кадр/с для кадров минимальной длины и 813 кадр/с для кадров максимальной длины. Соответственно, реальная максимальная производительность такой сети колеблется от 5,48 Мбит/с для кадров минимальной длины до 9,76 Мбит/с для кадров максимальной длины [21].
- Fast Ethernet (100 Мбит/с). Механизм CSMA/CD в сети Fast Ethernet работает так же, как и в сети Ethernet 10 Мбит/с, и пакеты имеют аналогичный размер, но их скорость распространения через среду передачи в десять раз выше за счет изменений в средствах физического уровня [21]. Исходя из этих предпосылок, будем считать, что максимальная пропускная способность сегмента Fast Ethernet составляет 148800 кадр/с для кадров минимальной длины и 8130 кадр/с для кадров максимальной длины, а максимальная производительность такой сети колеблется соответственно от 54,8 до 97,5 Мбит/с, что составляет 7986 пакетов/с.
- Gigabit Ethernet (1000 Мбит/с). На практике стандарт используется чрезвычайно редко, поэтому мы исключаем его из рассмотрения.

Таким образом, в расчетах интенсивность входного потока колеблется в пределах от 0,000813 кадр/мкс до 0,1488 кадр/мкс.

Необходимо отметить, что расчеты по приведенным исходным данным носят оценочный характер, однако успешно согласуются с экспериментом. Например, для равновероятного попадания кадра в каждый порт время пребывания пакета в коммутаторе, рассчитанное по приведенным формулам, в зависимости от длин буферов приема и передачи колеблется от 32 мкс для буферов нулевой длины до 168,406 мкс для буферов бесконечной длины.

Марковская модель межсетевого экрана

Межсетевые экраны (далее – МЭ) – это программный или аппаратно-программный комплекс, реализующий функции фильтрации сетевого трафика (информационных потоков) между двумя и более автоматизированными системами (АС) по некоторому набору правил (база правил или БП), определяемых политикой безопасности (ПБ) [22]. Необходимо отметить, что в современные МЭ зачастую включают дополнительные средства защиты. Это обусловлено тем, что МЭ устанавливается на границе нескольких АС, и расширение его функционала является весьма удобным.

Результаты рассмотрения существующих типов МЭ [22] иллюстрирует рис. 3.

В рамках данной работы рассмотрим модель межсетевого экрана-инспектора состояний.

Инспекторы состояний позволяют контролировать не отдельные пакеты трафика, а потоки трафика от источника к приемнику. Далее такие потоки мы будем называть соединениями. Каждое соединение контролируется в зависимости от используемых

протоколов на основе таблиц состояний, что позволяет отсеивать некорректно работающие соединения. Дополнительно контролируется время устаревания соединения: если между обработкой пакетов, принадлежащих одному потоку, проходит время, большее, чем установлено в МЭ как время устаревания соединения, такое соединение прекращается. Таким образом, в зависимости от рассматриваемого протокола инспектор состояний будет решать сходные задачи, тем не менее, трудно формализуемые в рамках одной модели. Построим методику рассмотрения работы инспекторов состояний на примере протоколов TCP.

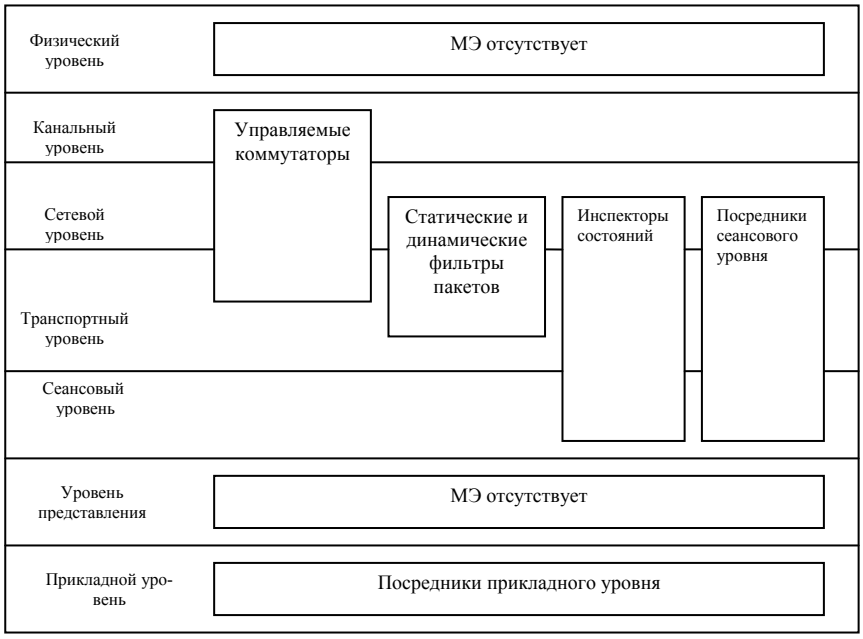


Рис. 3. Существующие типы МЭ

В работе [23] приводится схема переходов между состояниями TCP-соединения. Однако для МЭ не имеет значения, с какой стороны инициируется соединение. Схема переходов между состояниями TCP-соединения для межсетевого экрана изображена на рис. 4. Таким образом, у нас имеется граф переходов между состояниями МЭ. Состояние «Сброс соединения» возникает в случае аномального поведения хостов-участников соединения.

Аналогично тому, как это было сделано при построении модели коммутатора, построим систему дифференциальных уравнений:

$$\dot{p}_0(t) = -\lambda_{01} p_0(t)$$
$$\dot{p}_1(t) = \lambda_{01} p_0(t) - (\lambda_{16} + \lambda_{12}) p_1(t)$$
$$\dot{p}_2(t) = \lambda_{12} p_1(t) - (\lambda_{26} + \lambda_{23}) p_2(t)$$
$$\dot{p}_3(t) = \lambda_{23} p_2(t) - (\lambda_{36} + \lambda_{34} + \lambda_{35}) p_3(t)$$
$$\dot{p}_4(t) = \lambda_{34} p_3(t) - (\lambda_{46} + \lambda_{45}) p_4(t)$$
$$\dot{p}_5(t) = \lambda_{35} p_3(t) + \lambda_{45} p_4(t) - \lambda_{57} p_5(t)$$
$$\dot{p}_6(t) = \lambda_{16} p_1(t) + \lambda_{26} p_2(t) + \lambda_{36} p_3(t) + \lambda_{46} p_4(t) + \lambda_{56} p_5(t) - \lambda_{67} p_6(t).$$

Обсудим интенсивности переходов. В силу того, что на обработку каждого ip-пакета независимо от его длины МЭ тратит примерно равное время, наиболее тяжелый режим работы будет создаваться при обработке потока ip-пакетов минимальной длины.

Минимальный размер поля данных кадра Ethernet – 64 байт. Максимальный размер ip-заголовка – 60 байт. Таким образом, на область данных остается 4 байта, и в первом приближении можно считать, что количество ip-пакетов соответствует количе-

ству кадров канального уровня. Соответственно, для модели справедливы рассуждения, проведенные для коммутаторов, и интенсивность потока пакетов (интенсивность соединения) колеблется в пределах от 0,000813 кадр/мкс до 0,1488 кадр/мкс.

Пусть во время установления соединения перехват соединения нарушителем невозможен. Тогда интенсивности переходов между состояниями S0, S1, S2, S3 соответствуют интенсивности соединения. По аналогичным соображениям $\lambda_{57} = \lambda$.

Сброс соединений производится в соответствии со спецификацией [23], а также зависит от настроек, выставленных на МЭ. В рамках данной модели будем считать, что сбросом соединений в соответствии со спецификацией можно пренебречь. Так как время жизни пакета не превышает нескольких десятков секунд [23], в рамках данной работы будем считать его равным 50 с. Соответственно, настроим МЭ на разрыв соединения, если в течение 50 с не поступило ни одного пакета, принадлежащего этому соединению. Тогда $\lambda_{16} = \lambda_{26} = \lambda_{36} = \lambda_{46} = 0{,}02 \times 10^{-6}$ пакетов/мкс.

В соответствии с диаграммой состояний для завершения соединения необходимо не меньше 2 пакетов, и интенсивность перехода должна быть в 2 раза меньше $\lambda_{56} = 0{,}04 \times 10^{-6}$ пакетов/мкс.

Интенсивность переходов к завершению соединения зависит от времени, в течение которого используется сессия. Для получения численного результата будем считать, что передача данных длится в среднем 5 минут, и $\lambda_{45} = 0{,}003 \times 10^{-6}$ пакетов/мкс.

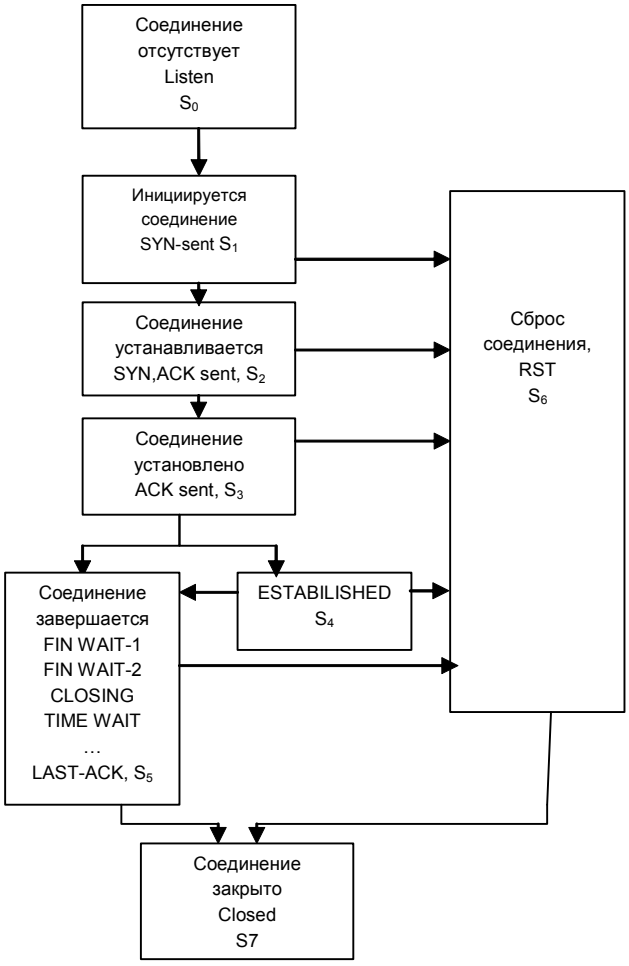


Рис. 4. Схема переходов между состояниями TCP-соединения для межсетевого экрана

Сброс соединения сразу после его создания $\lambda_{35} = 0{,}003 \times 10^{-7}$ пакетов/мкс. Как видно из графа, $\lambda_{67} = \lambda_{16} + \lambda_{26} + \lambda_{36} + \lambda_{46} + \lambda_{56}$ пакетов /мкс.

Значения вероятностных и временных характеристик, полученные в ходе оценочных расчетов, так же успешно согласуются с экспериментальными данными, как и для коммутаторов.

Заключение

Итак, введение терминов «информационная борьба» и «информационное оружие» обосновано необходимостью более широко и обобщенно рассматривать вопросы защиты и уничтожения информации, чем это делает теория информационной безопасности. Информационным оружием мы будем называть совокупность технических аппаратно-программных и программных средств, участвующих в информационной борьбе.

Из результатов анализа функционирования рассмотренных средств защиты следует вывод, что методика рассмотрения включает в себя следующие этапы.

- Производится анализ функционирования данного средства защиты или системы в целом и построение спецификации.
- Для расчета емкостных и временных характеристик строится многофазная система массового обслуживания. Необходимые характеристики получаются суммированием соответствующих характеристик для каждой фазы.
- Для расчета вероятностных характеристик производится формализация всех возможных состояний соответствующей системы.
- Выделяются безопасные и небезопасные состояния.
- Определяются интенсивности переходов средства защиты из одного состояния в другое.
- Строится граф переходов из состояния в состояние.
- В соответствии с графом строится система дифференциальных уравнений Колмогорова.
- По результатам решения системы уравнений Колмогорова определяются вероятностные характеристики состояний системы как функции времени.

Литература

1. Слипченко В.И. Войны шестого поколения. Оружие и военное искусство будущего. М.: Вече, 2002. 384 с.
2. Гриняев С.Н. Интеллектуальное противодействие информационному оружию. М.: Синтег, 1999. 232 с.
3. Прокофьев В.Д. Тайное оружие информационной войны. / Серия: Информатизация Россия на пороге XXI века. М.: Синтег, 1999. 152 с.
4. Киселев В.Д., Есиков О.В., Кислицын А.С. Защита информации в системах ее передачи и обработки. Под ред. Сухарева Е.М. М.: Солид, 2000. 200 с.
5. http://connect.design.ru/n6_7_96/oruzhie.html
6. Голубєв В.О. Розслідування комп'ютерних злочинів/Монографія. Запоріжжя: Гуманітарний університет "ЗІМТУ", 2003. 296 с.
7. Смагин В.А. Метод оценивания и обеспечения надежности сложных программных комплексов. http://sirine.da.ru
8. Ларионов А.М. и др. Вычислительные комплексы, системы и сети. Л.: Энергоатомиздат, 1987. 288 с.
9. Томашевський О.В. Визначення надійності технічних засобів захисту інформації / Інформаційні технології та захист інформації / Зб.наук. праць. 1999. №1. С. 97–103.

10. Вишневский В.М. Теоретические основы проектирования компьютерных сетей. М.: Техносфера, 2003. 512 с.
11. Мельников В.А. Защита информации в компьютерных системах. М.: Финансы и статистика, 1997. 368 с.
12. Расторгуев С.П. Информационная война. Проблемы и модели. Экзистенциальная математика. М.: Гелиос АРВ, 2006. 240 с.
13. Гриняев С.Н. Поле битвы – киберпространство: теория, приемы, средства, методы и системы ведения информационной войны. Мн.: Харвест, 2004. 448 с.
14. Иванов К.В. Системотехника средств поражения и защиты автоматизированных систем управления специального назначения. // Наука. Промышленность. Оборона. Труды VII всероссийской научно-технической конференции. Новосибирск: НГТУ, 2006. С. 172–176.
15. Авен О.П., Гурин Н.Н., Коган Я.А. Оценка качества и оптимизация вычислительных систем М.: Наука, Главная редакция физико-математической литературы, 1982. 464 с.
16. http://itelltd.kiev.ua/?page=articles&aid=35
17. http://jre.cplire.ru/jre/nov01/2/text.html#section3
18. Тихоненко О.М. Модели массового обслуживания в информационных системах. Мн.: УП «Технопринт», 2003. 327 с.
19. http://daily.sec.ru/dailypblshow.cfm?rid=45&pid=12453
20. Пискунов Н.С. Дифференциальное и интегральное исчисления для ВТУЗов. Изд-е 3-е. М. Физматгиз, 1961. 748 с.
21. Олифер В.Г., Олифер Н.А. Компьютерные сети: принципы, технологии, протоколы: учеб. пособие для студентов вузов 3-е изд.. Москва [и др.]: Питер, 2006. 957 с.
22. Лебедь С.В. Межсетевое экранирование. Теория и практика защиты внешнего периметра. М.: МГТУ им. Баумана, 2002, 304 с.
23. http://www.protocols.ru/files/RFC/rfc793.pdf

БАЗОВЫЕ ПАРАМЕТРЫ ФОРМАЛЬНЫХ МОДЕЛЕЙ ОЦЕНКИ ЗАЩИЩЕННОСТИ ИТ ПО «ОБЩИМ КРИТЕРИЯМ»

О.Е. Зайцев
Научный руководитель – к.т.н., доцент А.В. Любимов

В современных методах формального моделирования решающее влияние на результирующую модель оказывает выбор ее базовых свойств (назначение, точка зрения, границы моделирования), а также ее контекста. В работе представлены результаты анализа и выбора этих параметров в задаче построения функциональной модели оценки защищенности ИТ на основе ГОСТ ИСО/МЭК 15408 (Общие Критерии).

Введение

Национальный стандарт безопасности ГОСТ Р ИСО/МЭК 15408-2002 «Информационная технология. Методы обеспечения безопасности. Критерии оценки безопасности информационных технологий» (краткое название – «Общие критерии») действует в России с 1 января 2004 года. В его основе лежит стандарт в области оценки безопасности информационных технологий (ИТ) «Common Criteria» (CC) 1999 года, разработанный под эгидой Международной организации по стандартизации. Семилетний опыт использования «Общих Критериев» в мире и небольшой опыт, полученный при апробации в России, говорит о том, что применение методологии ОК способствует существенному повышению качества оценки и разработки продуктов и систем ИТ. Объемы работ по оценке и сертификации ИТ, выполняемые в настоящее время за рубежом и планируемые в России, с неизбежностью приводят к необходимости использования инструментальных программных средств поддержки деятельности по подготовке и проведению оценок. Современные методы разработки подобных средств предполагают широкое применение формальных моделей предметной области, по крайней мере, на стадиях специфицирования и высокоуровневого проектирования.

Решению перечисленных задач, как и многих других, способствует представление «Общих Критериев» в виде формальных моделей: функциональной модели деятельности по оценке защищенности ИТ, структурной модели защищенности ИТ и математической модели компонентов защищенности ИТ. Как правило, функциональная модель является основной и наиболее применяемой на практике, поэтому в данной статье будет рассматриваться построение функциональной модели защищенности ИТ, остальные формальные модели ОК будут рассмотрены в дальнейшем.

Функциональная модель ОК может быть построена с помощью методики структурного анализа и проектирования систем SADT, но, прежде чем приступать непосредственно к процессу моделирования, необходимо определить свойства этой модели и ее контекст.

В настоящей работе представлены результаты анализа объектной области и определение базовых свойств функциональной модели, а также построения контекстной модели оценки защищенности ИТ для дальнейшего построения полной функциональной модели на основе ГОСТ ИСО/МЭК 15408.

Постановка задачи выбора базовых свойств формальных моделей ОК

Основной причиной создания международного стандарта ИСО/МЭК 15408 «Критерии оценки безопасности информационных технологий» являлась необходимость унификации и взаимного признания национальных стандартов в области безопасности информационных технологий. Кроме того, единый набор международных стандартов, разработанный для достижения этой цели, позволяет упростить принятие решений при покупке программных продуктов и предоставляет возможность корпоративным заказчикам приобрести системы с более надежной защитой [1]. Применение ОК также спо-

собствует существенному повышению качества оценки и разработки продуктов и систем ИТ.

По прошествии некоторого времени использования стандарта ОК в области сертификации и оценки безопасности продуктов ИТ стало ясно, что предусмотренные стандартом средства не являются достаточно полными для полноценного их использования на практике и, безусловно, нуждаются в существенном расширении. Методология ОК в стандарте не имеет явного описания, ее элементы рассредоточены по тексту, который, вместе с сопутствующей нормативно-методической документацией, составляет около двух тысяч страниц. При этом значительная часть русскоязычной методической документации находится в стадии разработки или причислена к know-how и потому является недоступной.

Вскоре необходимость в более полном описании и расширении средств регламентации деятельности по оценке была восполнена новым нормативным документом «Общая методология оценки» (ОМО) [2], перевод английского варианта – «Common Methodology for Information Technology Security Evaluation» [3], сопровождающим ОК. Основным предметом рассмотрения ОМО являются именно действия по оценке защищенности с использованием критериев и свидетельств оценки, определенных в ОК [4].

Для реализации стандартов ОК и ОМО разработчикам пришлось рассмотреть общие подходы, методы и функции обеспечения защиты информации в организациях, а также описать, каким образом функции системы информационной безопасности обеспечивают выполнение требований конфиденциальности, целостности, достоверности и доступности информации [5]. Наличие таких обобщений позволяет применять методы научного анализа и в первую очередь – построение формальных моделей. В данной работе идет речь об одной из таких моделей – функциональной модели деятельности по оценке защищенности ИТ в рамках стандартов ОК и ОМО. Функциональная модель ОК должна отображать основные действия оценщика и разработчика, описанные, в основном, в ОМО.

Автоматизация многих действий по оценке и сертификации продуктов и систем ИТ возможна благодаря строгой регламентации деятельности оценщика и разработчика. Для проектирования, разработки и сопровождения соответствующего программного обеспечения необходимо иметь функциональные спецификации в стандартизованной электронной форме. Также стандартизованная функциональная модель предоставляет удобные средства контроля версий стандарта и обеспечивает возможность прослеживания последствий принимаемых в новых версиях изменений вплоть до уровня конкретных операций, что существенно облегчает как работу сотрудников испытательных лабораторий при проведении оценки, так и работу разработчиков при подготовке к ней. Наименее подготовленной к предстоящему внедрению ОК группой пользователей являются заказчики ИС и покупатели готовых продуктов. Для них формализованное графическое представление ОК является кратким справочником.

Перечисленные выше факторы обусловливают необходимость функционального моделирования методологии ОК.

Исходя из сравнительного анализа методик моделирования, для построения функциональной модели ОК обоснованно была выбрана методика SADT, поскольку в данной области моделирования она практически не имеет альтернативы, методики ARIS и CALS являются избыточными в данном случае. SADT-модель дает полное, точное и адекватное описание системы, имеющее конкретное назначение. Еще одним важным моментом перед построением модели, помимо выбора метода моделирования, является определение базовых свойств и контекста будущей модели согласно выбранной методике моделирования. Определение свойств проводится на самом раннем этапе проектирования. Таким образом, целью настоящей работы является представление базовых свойств и контекста функциональной модели.

Обзор зарубежных источников выявил лишь одну попытку функционального моделирования фрагментов ОК [6]. Однако при построении диаграмм реально не использовалась какая-либо определенная методика или метод, фактически они представляют собой иллюстрации, а не формализованную модель. Модель построена не по стандарту, что не позволяет говорить об ее дальнейшей применимости в области оценки и сертификации ИТ вследствие своей неполноты и субъективности. В отечественной литературе идея использования методики SADT в совокупности с методом DFD для построения формальной модели процессов оценки безопасности ИТ по стандарту ОК была предложена в [7].

Определение базовых свойств и контекста функциональной модели ОК

Свойства модели

В соответствии с методикой функционального моделирования, изложенной в международных [9] и российских [10] стандартах, определяющим свойством модели является ее назначение (Purpose), т.е. цель моделирования. Целью служит набор вопросов, на которые должна ответить модель. В результате анализа ситуации в области оценки безопасности ИТ было установлено, что основная цель моделирования – представление для целевой аудитории базовой системы процессов оценки защищенности ИТ по стандарту ОК в максимально компактной, наглядной и формализованной форме, допускающей как дальнейшую детализацию, так и коррекцию в соответствии с последующими версиями стандарта. Модель также может использоваться в качестве базовой части функциональных спецификаций для инструментального программного обеспечения поддержки процессов оценивания и в качестве справочника оценщиками при проведении оценки конкретной системы ИТ, и специалистами системы сертификации при разработке нормативно-методической документации.

Следующим свойством является точка зрения моделирования (Viewpoint). С этим свойством связана задача выбора точки зрения при моделировании, которая существенно влияет как на границы моделирования в целом, так и на объемлющий процесс детализации, осуществляемый в ходе построения самой модели. Точку зрения лучше всего представлять себе как место (позицию) человека или объекта, в которое надо встать, чтобы увидеть систему в действии. С этой фиксированной точки зрения можно создать согласованное описание системы так, чтобы в модели не смешивались несвязанные описания. В данном случае было принято решение проводить моделирование с точки зрения оценщика, так как эта роль является ключевой в процессе оценки, и ее использование позволяет построить наиболее универсальную функциональную модель. Цель и точка зрения – это основополагающие понятия SADT [10].

После фиксации цели и точки зрения моделирования стало возможным дать определение (Definition) модели, т.е., в соответствии с методом IDEF, описать ее содержание, а также определить границы (Scope) моделирования, т.е. дать описание процессов, лежащих вне области моделирования (или на ее границе) и определить степень детализации.

Определение модели. Модель содержит представление системы процессов оценивания безопасности ИТ, описывающей совокупность основных действий, которые должны выполнять заинтересованные стороны в соответствии со стандартом ОК версии 2.2. Эта система процессов дополнена процедурами, лежащими вне рамок ОК, но предусмотренных ОМО версии 1.1a (в частности – предварительными процедурами оценивания и процедурами завершения оценивания).

Границы моделирования. Модель содержит представление процессов подготовки, проведения и завершения оценки безопасности продуктов и систем ИТ (объектов оценки), предусмотренных ОК и ОМО. Детализация проводится до уровня, позволяю-

щего представить процессы генерации и использования основных документов, предусмотренных ОК и ОМО, их разделов и подразделов. Учитывая принятую точку зрения моделирования, подсистема процессов проведения оценки детализируется более подробно. Вне области моделирования лежат: функционирование системы аккредитации органов по сертификации, процессы деятельности системы сертификации ИТ, оценивание профилей защиты и разработка задания по безопасности. Оценка задания по безопасности (класс ASE) для рассматриваемого ОО включается в область моделирования как промежуточный процесс. Процессы разработки и регистрации нормативно-методических документов оценивания находятся вне границ моделирования, а сами нормативно-методические документы, наряду с объектом оценки, являются внешними ресурсами модели.

Деятельность владельцев (информационных) активов и иных пользователей системы ИТ, а также органов по сертификации составляет контекст модели. Элементы деятельности разработчиков и потребителей систем ИТ, а также заявителей включается в модель лишь в том случае, если они имеют непосредственное отношение к оцениванию. Остальные элементы деятельности этих агентов составляют контекст модели.

Контекст модели

В настоящее время актуальными являются два основополагающих документа ОК, описывающих функциональность деятельности по оценке защищенности ИТ, это – собственно Общие Критерии и сопровождающая их Общая методология оценивания. Часть I первого документа представляет собой общее введение, Часть II представляет собой справочник ФБО, и только часть III посвящена непосредственно элементам действий оценщика. Таким образом, для построения функциональной модели деятельности по оценке может использоваться только часть III ОК и, в небольшой степени, Часть I. Однако основным недостатком части III ОК является тот факт, что она представляет собой, по существу, просто справочник функциональных элементов, никак не связанных друг с другом. По ней не удается выделить общий поток работ при оценке конкретного продукта или системы ИТ при заданных условиях, а также трудно отделить работы, выполняемые на стороне разработчика, от работ, выполняемых на стороне оценщика, и представить их в виде взаимосвязанных и при этом явным образом различимых потоков.

Очевидно, что в плане организации процесса оценки эти задачи представляют существенную важность. В плане же функционального анализа решение этих задач предполагает правильное определение контекста модели:

(1) внешних агентов, т.е. внешних сущностей, деятельность которых осуществляется вне рамок модели;
(2) внешних ресурсов, т.е. информационных, материальных, финансовых и пр. ресурсов, которыми эти внешние сущности обмениваются с процессами, происходящими внутри модели.

В рассматриваемой предметной области роль внешних агентов очень важна, так как при выполнении многих действий по оценке весьма существенно, от кого именно оценщику приходит данный конкретный документ или кому именно оценщик направляет тот или иной свой вердикт, и в качестве нотации и метода функционального моделирования был выбран метод диаграмм потоков данных (DFD). Этот метод, по сравнению с более распространенным IDEF0, позволяет, во-первых, гораздо более полно отразить на диаграммах роли сущностей, которые инициируют, выполняют, заканчивают или используют результаты выполнения процессов, во-вторых, дает возможность гораздо более полно представить обмен ресурсами (в частности – документами) межу сущностями процессов. Более детальный сравнительный анализ преимуществ и недостатков методов IDEF0 и DFD приведен в работе [11].

В ходе анализа были сформированы четыре основные сущности процесса оценки. Первые три сущности являются внешними, четвертая (Оценщик) – внутренней.

Разработчик – организация или группа, ответственная за проектирование, разработку, тестирование и модификацию объекта оценки (ОО), в частности – за включение в ОО функций безопасности, описанных в ОК, и за устранение выявленных уязвимостей. При этом следует обратить внимание на то, что, как отмечено в ОМО, некоторые оценки (например, оценка на ОУД1 (оценочный уровень доверия)) могут вообще не требовать непосредственного участия разработчика. В этом случае сам заявитель представляет оценщику объект оценки и свидетельства оценки. Тем не менее, эта внешняя сущность определенно должна присутствовать в модели, так как ее роль возрастает с ростом заявляемого ОУД.

Заявитель – организация или группа, инициирующая оценку, т.е. являющаяся заказчиком оценки и отвечающая за обеспечение оценщика свидетельствами оценки. Заявитель заключает договор с оценщиком и оплачивает его работу. При формировании роли «Заявитель» также наблюдались содержательные проблемы. Например, в ОМО отмечается, что заявителем может быть разработчик ОО или организация, в которую входит разработчик (совмещение ролей), однако для полноты представления функциональности в модели эти роли было решено разделить.

Орган по сертификации – организация, прошедшая аккредитацию (ФСТЭК) на право проведения работ по сертификации по требованиям безопасности ИТ.

Оценщик – организация, уполномоченная органом по сертификации на право проведения работ по оценке продуктов с систем ИТ по требованиям безопасности (испытательная лаборатория).

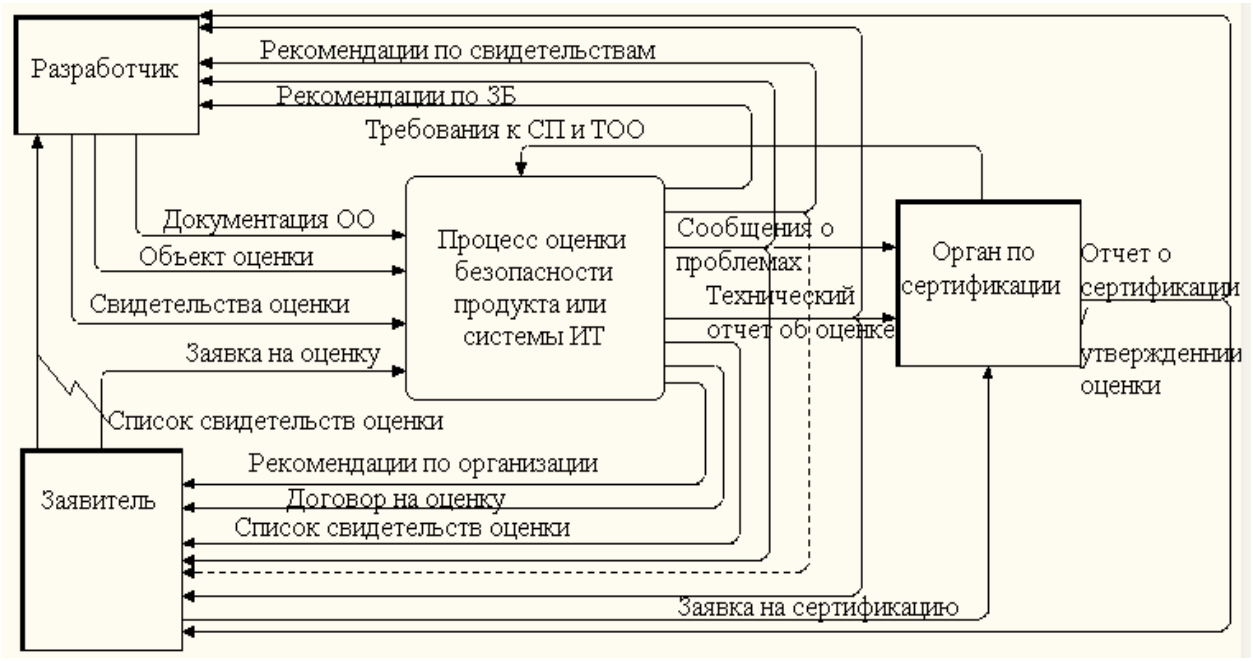


Рис. 1. Диаграмма контекстной модели деятельности по оценке защищенности ИТ

В завершение рассмотрения сущностей процесса оценки необходимо заметить, что в число четырех базовых внешних агентов не входит такая немаловажная роль, как Потребитель (организация, группа лиц или лицо, использующее ОО в конкретных условиях). Дело в том, что в рамках оценки ИТ по ОК функции, выполняемые потребителем (устанавливает цели безопасности, формирует специфические запросы по безопасности, решает, является ли продукт или система ИТ достаточно безопасной для ее предполагаемого применения), имеют только опосредованное отношение собственно к процессам оценки, за исключением лишь одного случая, когда потребитель может зака-

зать проведение анализа безопасности продукта или системы ИТ, т.е. оценку безопасности [12]. Однако в этом случае он просто-напросто выступает в роли Заказчика.

В ходе анализа были также выделены основные потоки данных (документов), которыми обмениваются основные внешние агенты, содержание которых, в основном, ясно из имен. В итоге была получена контекстная модель деятельности по оценке защищенности ИТ по стандартам ОК и ОМО, изображенная на рис. 1.

Научная новизна настоящей работы обусловлена тем фактором, что впервые к системе понятий и процессов, составляющих основное содержание Общих Критериев, применен в полном объеме стандартизованный и хорошо себя зарекомендовавший в других задачах метод функционального моделирования SADT. В результате описанные выше определенные базовые свойства и контекст позволили построить полную функциональную модель оценки защищенности ИТ по стандарту «Общие Критерии» в соответствии с методикой моделирования SADT и выбранной нотацией DFD, детализировав эту модель до уровня представления элементов действий оценщика и разработчика. На основании идей и эскизных начальных проработок по функциональному моделированию ОК, предложенных в [7], а также внесенных в методику SADT коррективов для учета особенностей моделируемого объекта, была получена модель, в состав которой входит 91 функциональная диаграмма, 241 процесс в иерархии и 410 ресурсов.

Заключение

Национальный стандарт безопасности ГОСТ Р ИСО/МЭК 15408-2002 «Информационная технология. Методы обеспечения безопасности. Критерии оценки безопасности информационных технологий» (краткое название – «Общие критерии») действует в России с 1 января 2004 года. Опыт использования «Общих Критериев» говорит о том, что применение методологии ОК способствует существенному повышению качества оценки и разработки продуктов и систем ИТ. Таким образом, оценка защищенности ИТ по ОК является перспективным направлением, а проблемы, связанные с использованием ОК – очень актуальными.

Решению многих задач способствует представление «Общих Критериев» в виде формальных моделей. Как правило, функциональная модель является основной и наиболее применяемой на практике. Функциональная модель ОК может быть построена с помощью методики SADT, но, прежде чем приступать непосредственно к процессу моделирования, необходимо было определить свойства этой модели и ее контекст.

В настоящей работе были определены базовые свойств функциональной модели, а также построена контекстная модель оценки защищенности ИТ для дальнейшего построения полной функциональной модели на основе ГОСТ ИСО/МЭК 15408.

Литература

1. Рэдклифф Д. Одна мера безопасности на всех // Computerworld. 2000. № 12. С. 7–8.
2. РД Безопасность информационных технологий. Общая методология оценки безопасности информационных технологий (проект). / ФСТЭК России, 2005.
3. Common Methodology for Information Technology Security Evaluation. Evaluation Methodology. January 2004. Version 2.2. Revision 256. CCIMB-2004-01-004.
4. Кобзарь М., Сидак А. Методология оценки безопасности информационных технологий по общим критериям // JetInfo. 2004. № 6 (133). 53 с.
5. Афанасьев В.Н. Общие критерии безопасности информационных систем. / Международная студенческая школа-семинар «Новые информационные технологии», г. Судак, 14–21 мая 2003, Тезисы докладов в 2-х томах М.: МГИЭМ, 2003. 641 с.

6. Prieto-Diaz R. The Common Criteria Evaluation Process. Process Explanation, Shortcomings, and Research Opportunities. / Commonwealth Information Security Center Technical Report CISC-TR-2002-03, December 2002 – CISC, James Madison University, USA.
7. Любимов А.В. Функциональная структура общих критериев оценки безопасности информационных технологий. / Труды 9-й научно-технической конференции «Теория и технология программирования и защиты информации. Применение вычислительной техники». Санкт-Петербург, 18 мая 2005 г. С. 20–24.
8. Federal Information Processing Standards Publication 183. Announcing the Standard for «Integration Definition for Function Modeling (IDEF0)». 21 December 1993.
9. РД IDEF0-2000. Методология функционального моделирования IDEF0. Руководящий документ. Издание официальное. Госстандарт России. Москва, 2000.
10. Марка Д., Макгоуэн К. Методология структурного анализа и проектирования SADT М.: МетаТехнология, 1993. 240 с.
11. Калянов Н., Козлинский А.В., Лебедев В.Н. Сравнительный анализ структурных методологий // Системы управления базами данных. 1997. №05. С. 12–17.
12. ГОСТ Р ИСО/МЭК 15408-1-2002. Информационная технология. Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий. Часть 1. Введение и общая модель. М.: Госстандарт России, 2002.

ПАКЕТ ВИЗУАЛИЗАЦИИ ГРАФА УПРАВЛЕНИЯ МОДЕЛИ МИШЕНИ КАК ЧАСТЬ ПРОГРАММНОГО КОМПЛЕКСА ПО ВЫЯВЛЕНИЮ ВРЕДОНОСНОГО КОДА И НЕДЕКЛАРИРОВАННЫХ ВОЗМОЖНОСТЕЙ ПО

Д.А. Гусарова
Научный руководитель – д.т.н., профессор Л.Г. Осовецкий

Группой разработчиков кафедры БИТ СПбГУ ИТМО разрабатывается программный комплекс по верификации программного обеспечения (ПО), выявлению недекларированных возможностей и вредоносного кода. В статье рассказывается об одной из компонент разрабатываемого программного комплекса, а именно – о пакете визуализации графа управления программы, названном Alpha.

Введение

Сегодня вирусные атаки прочно удерживают пальму первенства во всех хит-парадах угроз IT-безопасности. Эти вредители наносят прямой финансовый ущерб, а также служат отправной точкой для реализации многих других опасных угроз, среди которых – кража конфиденциальной информации и несанкционированный доступ к данным. В свою очередь, антивирусная индустрия предлагает несколько новых подходов к защите IT-инфраструктуры: проактивные технологии, форсированный выпуск критически важных вакцин, серьезное увеличение частоты обновления антивирусных баз и т.д.

Характерной чертой сегодняшнего дня является не только огромный ущерб, наносимый вирусными атаками, но и непрекращающийся рост числа самих вредоносных кодов. Отметим, что в 2005 году рост популяции компьютерных вредителей приобрел просто взрывной характер. Например, по данным «Лаборатории Касперского», количество ежемесячно детектируемых вирусов выросло на конец 2005 года в среднем до 6368 экземпляров, а по итогам года рост составил 117 %, в то время как в 2004 году рост составил только 93 %.

На данный момент ни одна антивирусная программа не дает стопроцентной гарантии обнаружения вредоносного кода. Их проактивный/эвристический анализ является скорее слабым звеном в поиске вредоносного кода и далеко не всегда обнаруживает неизвестные вирусы.

Основная часть

Группой разработчиков с кафедры БИТ СПбГУ ИТМО разрабатывается программный комплекс по верификации ПО (программного обеспечения), выявлению недекларированных возможностей и вредоносного кода. В статье рассматривается одна из компонент разрабатываемого программного комплекса, а именно – пакет визуализации графа управления программы, названный Alpha.

Пакет визуализации Alpha включает в себя 2 программы:

1. Alpha32 – 32-битная программа на языке Delphi, которая анализирует тестовую программу (далее мишень) и строит по ее коду граф управления.
2. Dot.exe – собственно сам визуализатор.

Основные принципы работы Alpha32

На вход программы подается файл дизассемблированного листинга мишени (далее – файл), полученный с помощью `W32Dasm` (см. рис. 1). Alpha32 находит в файле условные и безусловные переходы, запоминает их последовательность и адреса, а также адреса, на которые указывают эти переходы.

```
:0058EBBD 16                    push ss
:0058EBBE EC                    in al, dx
:0058EBBF 6801C1008B            push 8B00C101
:0058EBC4 11D8                  adc eax, ebx
:0058EBC6 207784                and byte ptr [edi-7C], dh
:0058EBC9 19709F                sbb dword ptr [eax-61], esi
:0058EBCC 07                    pop es
:0058EBCD E689                  out 89, al
:0058EBCF 5D                    pop ebp
:0058EBD0 F8                    clc
:0058EBD1 648F5711              pop fs:[edi+11]
:0058EBD5 C855FC8B              enter FC55, 8B
:0058EBD9 F8                    clc
:0058EBDA 4B                    dec ebx

* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:0058EB78(C)
|
:0058EBDB E43C                  in al, 3C
:0058EBDD DADD                  fcmovu st(0), st(5)
:0058EBDF B510                  mov ch, 10
:0058EBE1 9E                    sahf
:0058EBE2 8B2CD3                jnz 0058EC13

:0058EBE5 2CF0                  sub al, F0
:0058EBE7 FE40A5                inc [eax-5B]
:0058EBEA 5E                    pop esi
:0058EBEB 8531                  test dword ptr [ecx], esi
:0058EBED 8CDA                  mov dx, ds
:0058EBEF 1303                  adc eax, dword ptr [ebx]
:0058EBF1 F8                    clc
:0058EBF2 893E                  mov dword ptr [esi], edi
:0058EBF4 816FFF16210650        jmp 0058EBBD
```

Рис. 1. Пример файла дизассемблированного листинга мишени (primer.lst)

```
digraph G { node[shape = record]; edge[headport = n];
Start[label="START", shape = "box", style="filled"];
Start -> Addr0058EBBD;
Addr0058EBBD[label="0058EBBD:\lPUSH ss\lIN al, dx\lPUSH 8B00C101\lADC eax,
ebx\lAND byte ptr [edi-7C], dh\lSBB dword ptr [eax-61], esi\lPOP es\lOUT 89, al\lPOP
ebp\lCLC \lPOP fs:[edi+11]\lENTER FC55, 8B\lCLC \lDEC ebx\lIN al, 3C\lFCMOVU
st(0), st(5)\lMOV ch, 10\lSAHF \lJNZ 0058EC13"];
Addr0058EBBD -> Addr_EXIT[label = "Y (out of program)"]
Addr0058EBBD -> Addr0058EBE5[label = N]
Addr0058EBE5[label="0058EBE5:\lSUB al, F0\lINC [eax-5B]\lPOP esi\lTEST dword
ptr [ecx], esi\lMOV dx, ds\lADC eax, dword ptr [ebx]\lCLC \lMOV dword ptr [esi],
edi\l"];
Addr0058EBE5 -> Addr0058EBBD[label = Jump]
Addr_EXIT[label="EXIT", shape = "box", style="filled"];
}
```

Рис. 2. Файл описания графа управления программы на языке Dot. (primer.dot)

После этого происходит разделение содержимого файла на блоки. Каждый блок представляет собой последовательность операторов, которая заканчивается условным или безусловным переходом. Полученный набор блоков, их последовательность, а также другие параметры описываются с помощью специального языка dot. Результатом работы программы является файл описания графа управления программы на языке dot (рис. 2).

Dot.exe – собственно сам визуализатор

Данная программа разработана фирмой Graphviz и распространяется бесплатно. Сайт производителя: http://www.graphviz.org. На вход программы подается файл Dot описания (см. рис. 2), по которому программа «рисует» граф управления. На выходе получаем файл формата JPG – визуализированный граф управления мишени (рис. 3).

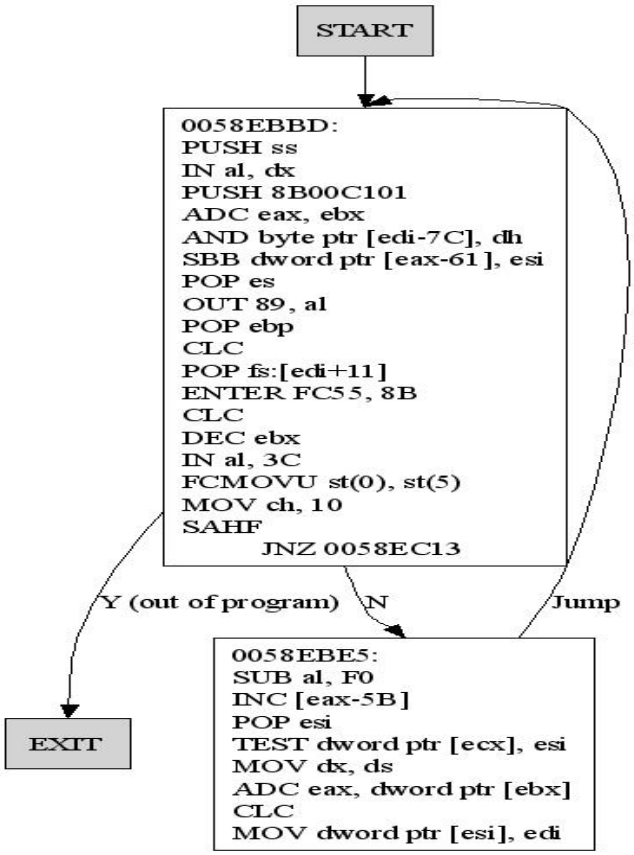


Рис. 3. Визуализированный граф управления мишени (primer.jpg)

Заключение

Работа над представленным пакетом далека от завершения. На данном этапе нет возможности обрабатывать мишени с вычисляемыми и шифрованными адресами, а также самомодифицирующимся кодом. Впоследствии предполагается добавить в пакет возможность динамического анализа и построения графа управления по загруженной (работающей в данный момент) модели мишени.

Литература

1. Шевченко А. Microsoft Vista против вирусов: кто кого? / http://www.viruslist.com – 2007.
2. Гостев А. Обзор вирусной активности, январь 2007 / http://www.viruslist.com – 2007.
3. Новое поколение уничтожителей информации. / http://www.novirus.ru – 2007.
4. Немолочнов О.Ф., Раков С.В. Автоматизация логического проектирования. / Учебное пособие. Санкт-Петербург: ИТМО, 1995. 60 с.

СТРУКТУРНОЕ МОДЕЛИРОВАНИЕ КОНКУРЕНТНОГО ВЗАИМОДЕЙСТВИЯ ИНДИВИДУУМОВ В КОРПОРАТИВНОЙ СРЕДЕ

М.В. Береговой
Научный руководитель – д.т.н., профессор Л.Г. Осовецкий

В статье исследовано многосубъектовое взаимодействие между субъектами корпоративной системы с использованием элементов тактической и стратегической конкуренции. Предложены структурные модели конкурентной борьбы субъектов корпоративной системы за ресурсы, принципы возникновения конкуренции и языковые связи между индивидуумами.

Введение

Технический прогресс в области информационных технологий (ИТ) и телекоммуникационных систем (ТКС) привел к созданию нового объекта взаимодействия между субъектами корпоративной системы – многосубъектового взаимодействия с использованием элементов тактической и стратегической конкуренции. В настоящее время создалась ситуация, когда качество взаимодействия во многом зависит от качества решения тактических и стратегических вопросов конкурирующего взаимодействия в корпоративной среде. В процессе развития информационных технологий наличие конкурирующего взаимодействия между компонентами системы и борьба за установление контроля отдельных компонентов над однородными общесистемными ресурсами привели к обострению вопросов защиты и безопасности информации.

Конкуренция может возникать как за ресурсы индивидуумов, не входящих в корпорацию, так и входящих в нее. Естественно, ни одна корпорация не стремится к одномоментному захвату ресурсов другой корпорации, так как это требует, как правило, слишком значительных затрат. В данной работе предполагается, что корпорации примерно равны, следовательно, приведенный выше силлогизм верен. Конкурирующая корпорация стремится взять под контроль индивидуумы и их ресурсы, тем самым ослабляя соперника.

Конкуренция в любой момент времени представляет собой взаимодействие двух индивидуумов. В данной статье рассмотрено именно такое парное взаимодействие. Один индивидуум при помощи языка и информации оказывает влияние на другого, например, передавая ложную информацию и/или выдавая себя за члена корпорации. Целью является получение доступа к части корпоративных ресурсов, которыми владеет индивидуум. Также возможна компрометация индивидуума и, как следствие, компрометация корпоративных ресурсов и нарушение нормального информационного взаимодействия в корпорации.

В работе рассмотрены принципы конкурентной борьбы за контроль за ресурсы на субъектовом уровне и методы воздействия на индивидуумы, использующие информацию и язык как инструмент. Конкуренция за обладанием ресурсами – один из основополагающих факторов развития корпоративной системы. Основываясь на корпоративной теории информации [1], в которой описаны методы построения корпоративных моделей информационной системы, и моделях эффективного распределения ресурсов [2], предложены модели конкурентной борьбы субъектов корпоративной системы за ресурсы, выполненные в форме структурных моделей [3]. Для моделирования были выбраны: методика OOAD и метод OMG UML.

Понятие корпорации, ресурсов и системы

Корпоративную систему можно рассматривать как совокупность субъектов, обладающих частью общих характеристик. Между такими субъектами существуют инфор-

мационные взаимодействия, т.е. взаимный или односторонний обмен данными. Из этого утверждения следует, что корпоративную систему можно представить как информационную систему, обладающую совокупностью субъектов, осуществляющих информационное взаимодействие. Особенность корпоративной системы как информационной системы заключается в корпоративном характере информационных процессов.

Информационное взаимодействие возникает только при наличии других подобных и конкурирующих субъектов, т.е. системы субъектов, что является основой для возникновения между субъектами с целью организации совместной и конкурентной борьбы за ресурсы развития и существования как системы в целом, так и субъектов этой системы. Естественно, подразумевается, что субъекты системы находятся в постоянном информационном взаимодействии между собой.

Субъекты такой системы разделяются по уровням сложности, и используемые ими ресурсы также можно разделить по уровням комплексности, т.е. чем сложнее субъект, тем большее разнообразие ресурсов он может использовать.

Объединение во временные объединения – корпорации – связано с необходимостью обеспечения «секретности», по Шеннону, от других членов [4]. Поэтому свойства языка корпорации определяются числом возможных объединений в корпорации, при соблюдении определенных правил безопасности. Также имеет смысл учитывать возможность или невозможность объединения субъектов в корпорацию. Здесь имеется в виду территориальная разрозненность субъектов и параметры каналов связи. Логично, что подобное объединение при сильном удалении субъектов друг от друга, потребует значительных затрат на физическое обеспечение взаимодействия.

В определенный момент возникает внутрикорпоративная конкуренция за ресурсы – как материальные, так и информационные. Защита и безопасность информации становятся важнейшей частью как существования и правильного функционирования системы в целом, так и индивидуального, заключающегося в оптимальном изменении и использовании ресурсов [2].

Время жизни и возникновение конкуренции

Образование, развитие и функционирование любой системы аналогично биологическим процессам в природе. Поэтому сравнение возможно, и далее будут использоваться термины «популяция», т.е. совокупность субъектов, и «индивидуум» – собственно сам субъект. Каждый индивидуум имеет время жизни, в течение которого создает, перераспределяет и приобретает информацию, участвует в создании новых индивидуумов и защищенных образований. Время жизни корпорации больше, чем у индивида. Время жизни корпорации тратится на приобретение ресурсов и увеличение количества индивидов. Для лучшей конкуренции необходимо большее количество субъектов, соответственно, корпорация растет. Как только численность индивидов в корпорации стабилизируется, конкуренция приобретает вид перераспределения и обработки информации.

Рассмотрим популяцию в ее простейшем виде, который изображен на рис. 1. Имеется набор индивидуумов, каждый из которых контролирует какую-то часть ресурсов системы. На данном этапе все индивидуумы контролируют одинаковую часть системных ресурсов. Здесь нет борьбы за распределение системных ресурсов между индивидуумами. При росте системы индивидуумы начинают пытаться овладеть одинаковой частью системных ресурсов, т.е. появляется взаимодействие. Для взаимодействия между индивидуумами необходим инструмент, этим инструментом являются информация и язык. Все индивидуумы примерно равны, соответственно, возникает необходимость в объединениях, которые будут обладать большими системными ресурсами, чем одиночные индивидуумы.

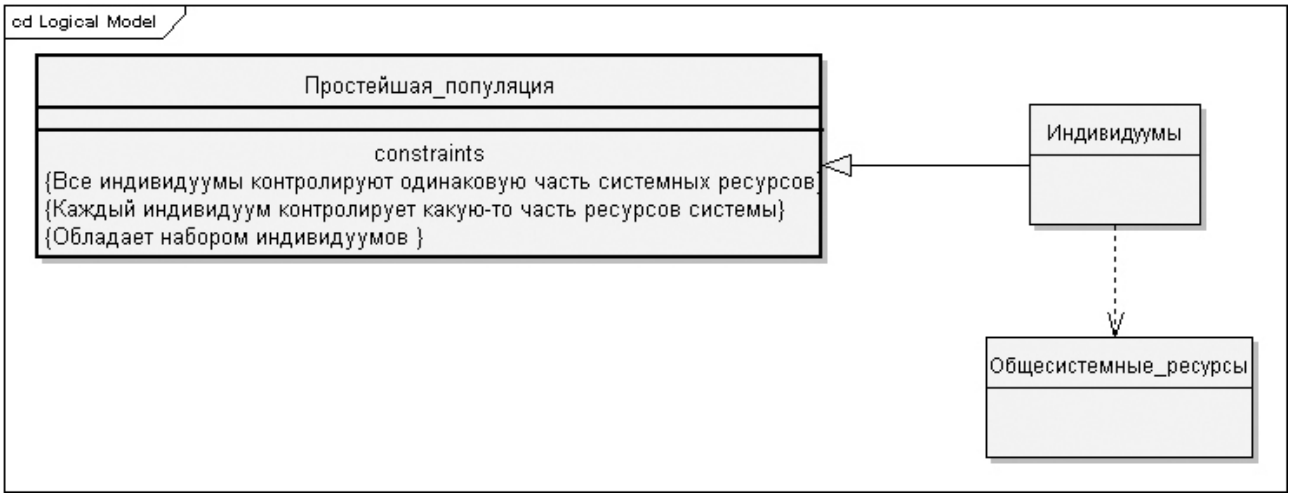


Рис. 1. Простейшая популяция

При наличии информации как инструмента борьбы за системные ресурсы возникают объединения, т.е. группы индивидуумов, контролирующих общую часть системных ресурсов. Здесь информация выступает в роли характеристики системных ресурсов и, значит, помогает противостоять остальным индивидуумам популяции. При помощи информации индивидуум в объединении контролирует часть системных ресурсов объединения, которая заведомо больше, чем часть ресурсов, которые контролирует отдельный индивидуум. Эта ситуация представлена на рис. 2.

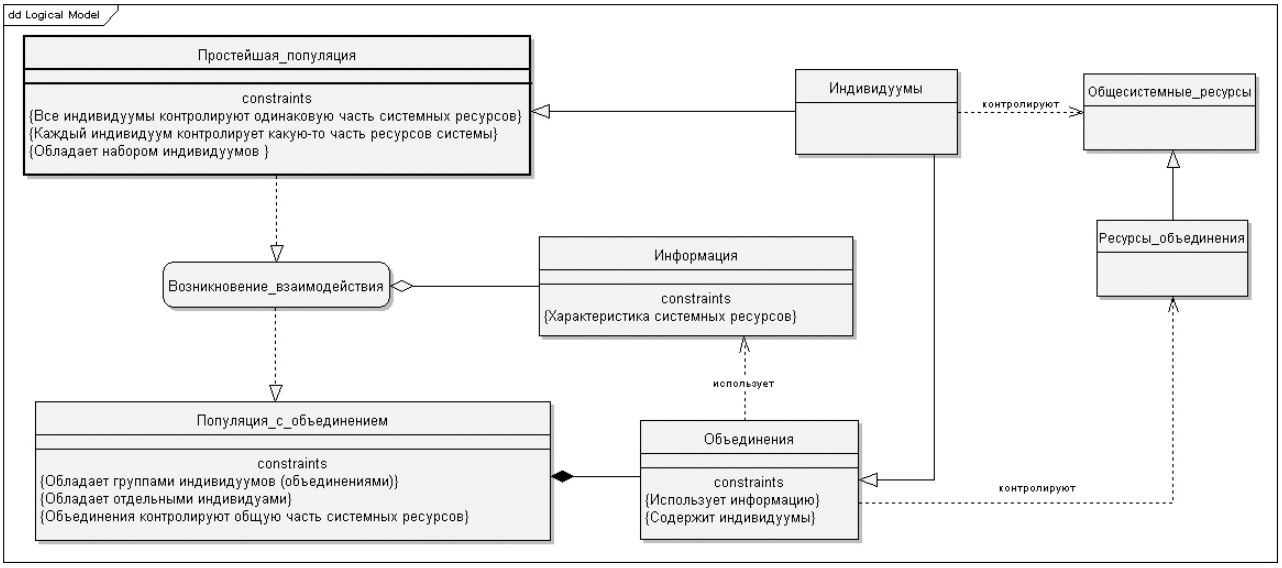


Рис. 2. Популяция с объединением

После того, как образовались несколько объединений, необходим новый элемент для борьбы за ресурсы, которые или еще не распределены по объединениям, или за ресурсы самих объединений, причем между индивидуумами этих объединений. В этот момент и появляется язык, который способствует распределению информации между индивидуумами. Появление языка влечет за собой появление корпораций, в которых индивидуумы объединены для получения большого количества ресурсов и существуют связи для обмена информацией, т.е. происходит как само накопление информации, так и ее распределение. Популяция с корпорацией приведена на рис. 3.

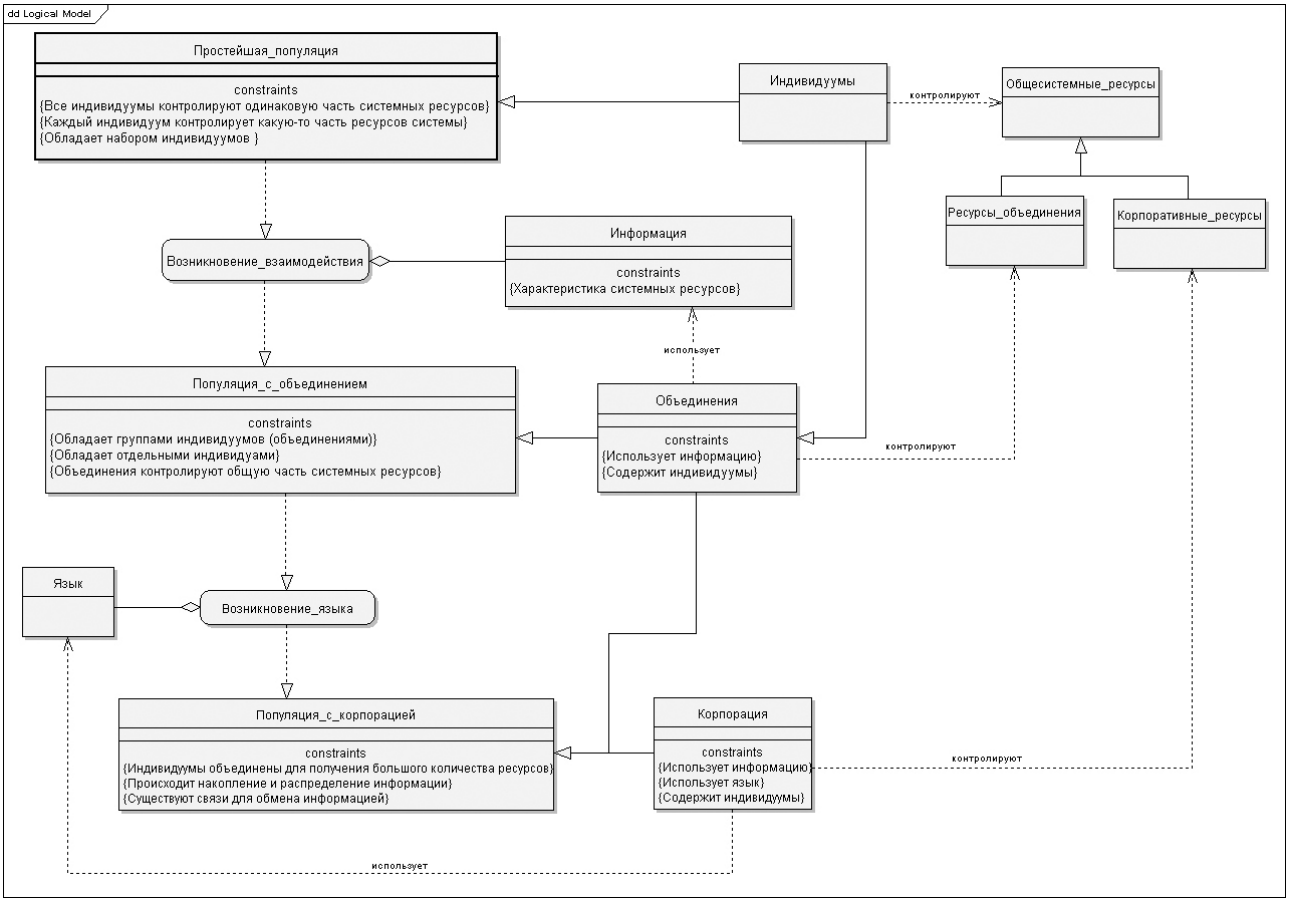


Рис. 3. Популяция с корпорацией

В итоговой модели отражен процесс развития популяции, ее составляющие и связи между ними.

В корпорации каждый индивидуум при помощи языка устанавливает различные отношения с другими индивидуумами. В результате таких отношений возникает распределение информации внутри корпорации. Такая деятельность корпорации направлена на накопление корпоративных ресурсов и их сохранение. Так как информация – это инструмент для накопления ресурсов, то возникает необходимость в ее защите. Потеря информации, которой располагают индивидуумы, не может привести к большим потерям корпоративных ресурсов.

Языковые связи

Рассмотрим простейший пример взаимодействия внутри корпорации из трех индивидуумов, отображенный на рис. 4.

Третий индивидуум устанавливает языковую связь со вторым и обменивается языковыми элементами. Этот обмен информацией направлен на увеличение объема контролируемых ресурсов, что происходит из-за получения новой информации. Здесь появляется понятие ценности информации – это величина корпоративной информации, которую теоретически можно использовать, относительно всей информации, располагаемой корпорацией. В данной ситуации ценность информации первого индивидуума – ноль, поскольку он не участвует в языковых связях, а второго и третьего – одна вторая, так как они образуют корпорацию, состоящую из двух индивидуумов, внутри объединения.

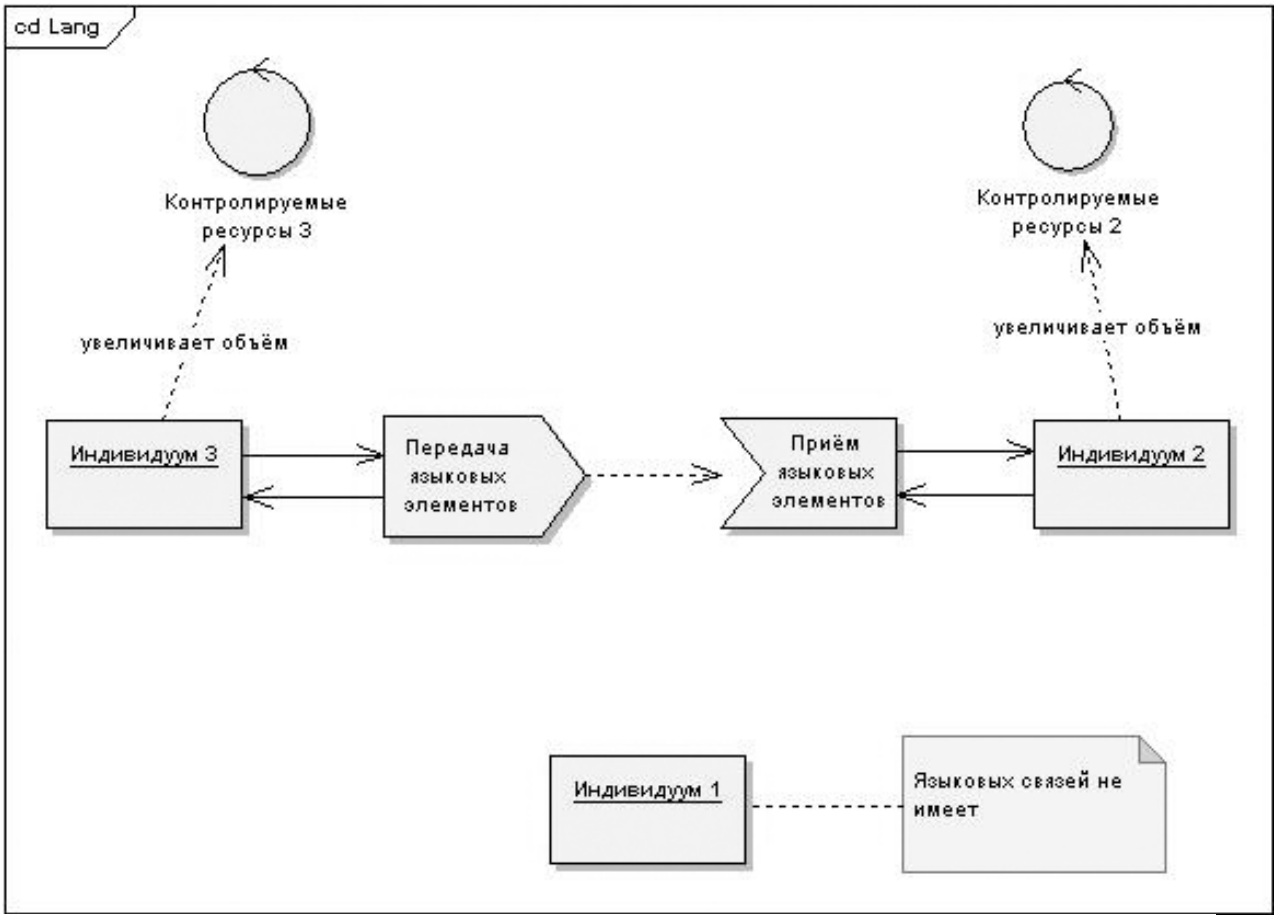


Рис. 4. Языковая связь в корпорации

Индивидуумы внутри корпорации разрабатывают набор языков. Эти языки используются очень долгое время по сравнению со временами жизни корпораций и индивидуумов. Языки развиваются вместе с корпорациями и сохраняются после исчезновения отдельных индивидуумов.

В разное или одно и то же время индивидуумы входят в состав разных корпораций. Поэтому одни и те же языки могут использоваться в разных связях и корпорациях.

Конкурентное воздействие на корпорацию через индивидуума

В подобных случаях возможны ситуации, когда один индивидуум передает другому ложную информацию, при этом пользуясь языками, которые свойственны компрометируемому индивидууму или корпорации, в которую он входит. Это позволяет получить доступ к информации или ресурсам. Такое воздействие может быть также направлено на получение доступа к корпоративным ресурсам через индивидуума или же на компрометацию части корпорации. Возможно воздействие на некоторую часть индивидуумов корпорации одновременно, т.е. масштабная атака, или разнесенное во времени воздействие. Для реализации первого варианта необходимы достаточно большие ресурсы, при этом атакующая корпорация может открыться для своих конкурентов. Такой вариант, как правило, предпринимается при неравных ресурсах и возможностях противодействующих корпораций, что ведет к поглощению более слабой.

Если корпорации примерно равны, то вариант с последовательной компрометацией – предпочтительнее. Нося стратегический характер, он позволяет более точно распланировать и распределить затраты.

Подобное воздействие на одного или нескольких индивидуумов нарушает и искажает информационные потоки внутри корпорации за счет компрометации индивидуу-

мов и внедрения ложной информации, тем самым ослабляя сопротивляемость корпорации.

Заключение

В статье проанализированы понятия корпорации, системы и ресурсов. Исследовался новый объект между субъектами корпоративной системы – многосубъектовое взаимодействие с использованием элементов тактической и стратегической конкуренции.

Корпоративную систему можно рассматривать как систему различных субъектов, между которыми происходят постоянные взаимодействия по обмену формализованными данными.

Различные по сложности и функциональным особенностям субъекты системы используют различные ресурсы, но некоторые ресурсы могут одновременно использоваться несколькими различными субъектами, в результате происходит борьба за ресурсы. Это – фактор наличия конкурирующего взаимодействия между компонентами системы за контроль отдельных компонент над однородными общесистемными ресурсами. В процессе развития информационных технологий это привело к обострению вопросов защиты и безопасности информации.

Предложены структурные модели конкурентной борьбы субъектов корпоративной системы за ресурсы, принципы возникновения конкуренции и языковые связи между субъектами.

Литература

1. Осовецкий Л.Г., Немолочнов О.Ф., Твердый Л.В., Беляков Д.А. Основы корпоративной теории информации СПб: СПбГУИТМО, 2004. 83 с.
2. Берзин Е.А. Оптимальное распределение ресурсов и теория игр. М.: Радио и связь, 1983. 216 с.
3. Буч Г., Рамбо Д., Джекобсон А. Язык UML. Руководство пользователя: Пер. с англ. М.: ДМК, 2000. 275 с.
4. Шеннон К. Математическая теория связи. М.: ИИЛ, 1963. 207 с.

РАЗРАБОТКА ТЕСТОВЫХ ПРОГРАММ ДЛЯ ПРОГРАММНОГО КОМПЛЕКСА ПО ВЫЯВЛЕНИЮ ВРЕДОНОСНОГО КОДА И НЕДЕКЛАРИРОВАННЫХ ВОЗМОЖНОСТЕЙ В ПРОГРАММНОМ ОБЕСПЕЧЕНИИ

А.Н. Карлыханов
Научный руководитель – д.т.н., профессор Л.Г. Осовецкий

В статье рассказывается о разработке тестовых программ как части программного комплекса по выявлению вредоносного кода и недекларированных возможностей в программном обеспечении.

Введение

На сегодняшний день информационные технологии встречаются практически в каждой отрасли жизнедеятельности человека. Развитие современной hоссийской экономики и экономики конкретного предприятия все больше зависит от информационных ресурсов и внедряемых технологий. В России ежегодно регистрируется до десяти тысяч преступлений в области компьютерной информации. В то же время большая часть компьютерных преступлений остается скрытой и не регистрируется правоохранительными органами. При этом процентное соотношение раскрытых и нераскрытых правонарушений пока не установлено. Как следствие? все острее встает вопрос обеспечения требуемого уровня информационной безопасности.

Одним из аспектов информационной безопасности является сохранение целостности информации, т.е. предотвращение изменения кода программ посредством вирусных атак и выявление программных закладок. К сожалению, современные антивирусные комплексы не могут справиться с постоянно изменяющимися и совершенствующимися вирусами, а так же гарантированно выявлять недекларированные возможности [1] (далее НДВ) в программном обеспечении.

Основная часть

Группой разработчиков кафедры безопасных информационных технологий СПбГУ ИТМО разрабатывается программный комплекс по верификации программного обеспечения, выявлению НДВ и вредоносного кода. На данном этапе разработки для решения этой задачи используются методы статического анализа, т.е. анализа кода программы без выполнения его на вычислительной машине. Используемый метод основан на анализе комплексных кубических покрытий логических условий программ. Такой метод позволяет наиболее точно описать логику исследуемых программ, что впоследствии позволит гарантированно обнаруживать НДВ и вирусы, а также выявлять некоторый класс ошибок в программном обеспечении.

В данной работе рассказывается об одной из компонент разрабатываемого программного комплекса, а именно о создании тестовых программ для проверки работоспособности комплекса. В эту задачу входило написание небольших программ, содержащих несколько логических веток выполнения. Такие программы называются мишенями (рис. 1).

Обязательное наличие логических ветвлений в программах мишенях обусловлено тем, что разрабатываемый комплекс анализирует логику программы, а не ее код целиком. Поэтому в мишенях использовались условные и безусловные переходы, вызовы функций и циклы.

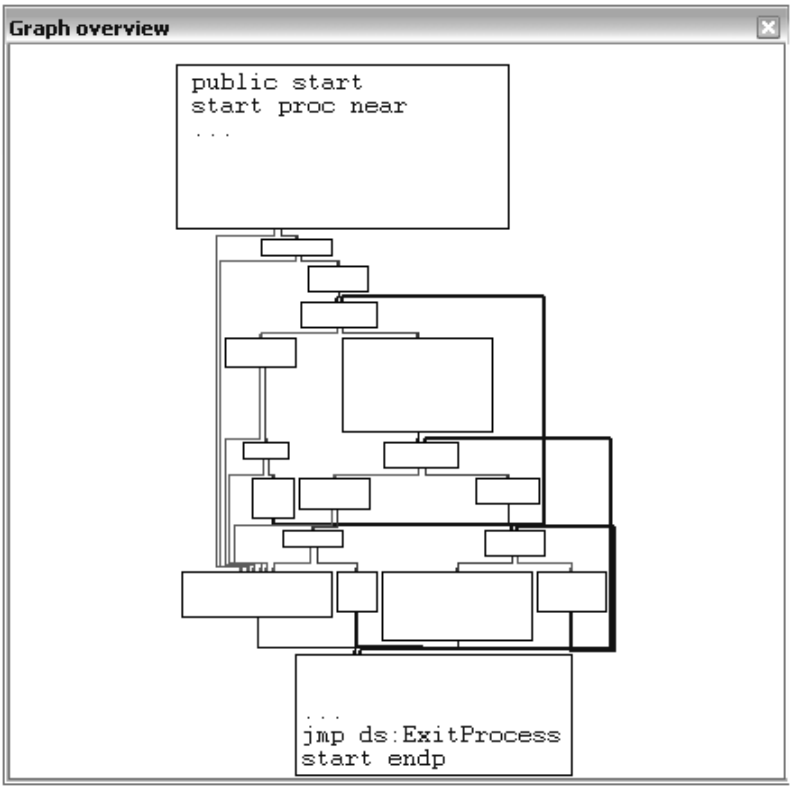


Рис. 1. Блок-схема мишени, сгенерированная дизассемблером IDA

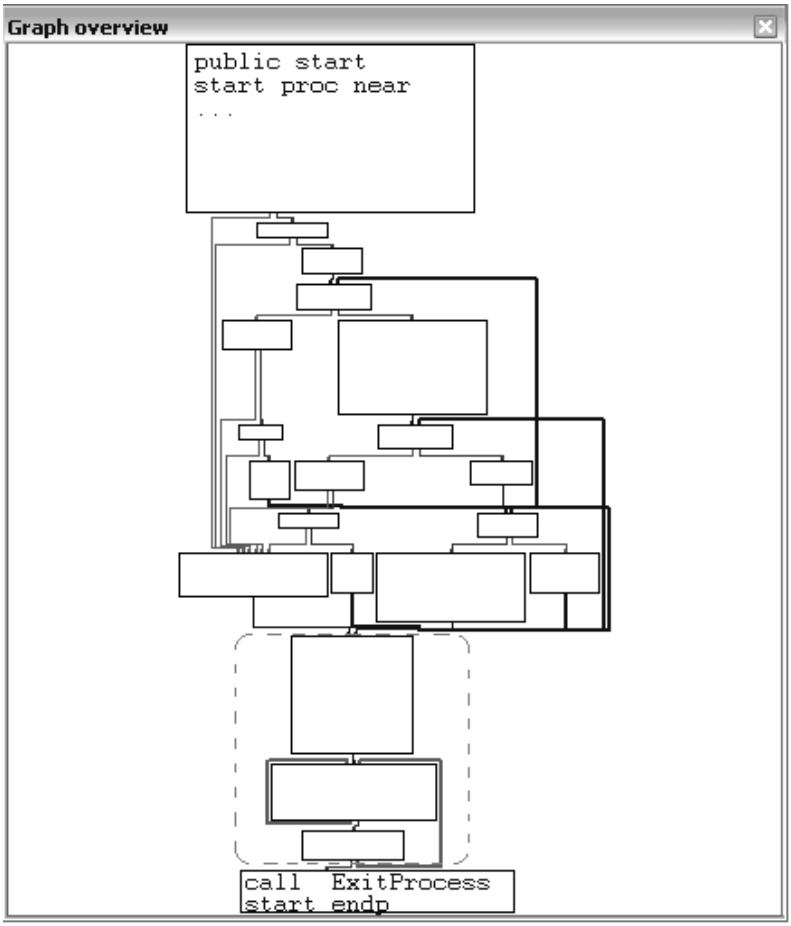


Рис. 2. Блок-схема смоделированной мишени с НДВ, сгенерированная дизассемблером IDA

На следующем этапе было необходимо смоделировать внедрение НДВ в готовые мишени. Внедрение НДВ моделировалось таким образом, что сначала отрабатывала функциональная часть программы мишени, а затем, незаметно для пользователя, функции, заложенные в НДВ (рис. 2). В качестве дополнительных функций исполняемых в полученных моделях мишеней использовали функции, свойственные обычным троянским программам, такие как регистрация своей копии для автозапуска (рис. 3, 4).

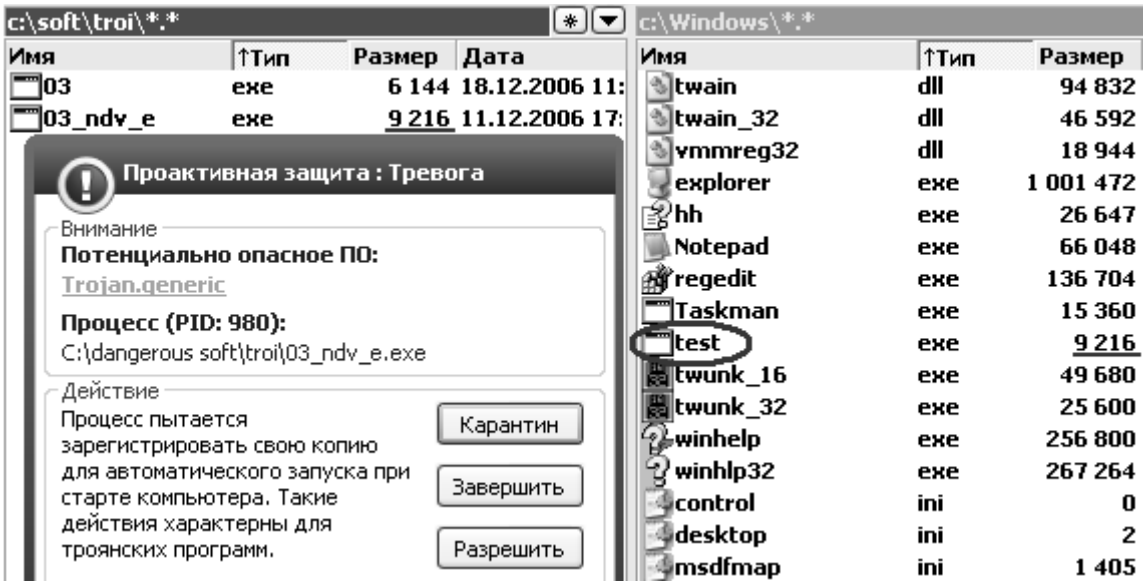


Рис. 3. Результат работы модели 03_ndv_e.exe

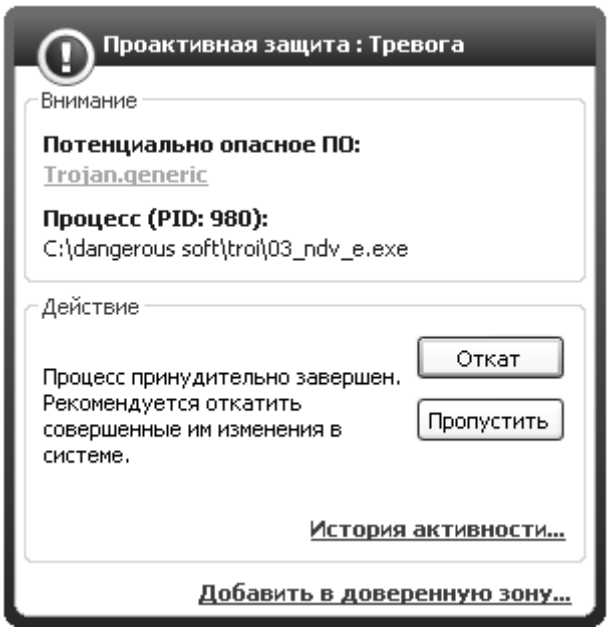


Рис. 4. Реакция антивируса Касперского

Результатом работы разрабатываемого комплекса будет анализ исходных мишеней и моделей, построенных на их основе, имитирующих вредоносный код. Уже на данном этапе комплекс способен выявить внесенные в разработанные мишени изменения. Такие изменения обозначены пунктиром на рис. 2.

Заключение

На сегодняшний день ни одна из антивирусных программ не может дать стопроцентной гарантии по обнаружению вредоносного кода в программном обеспечении. Даже эвристический анализ далеко не всегда способен обнаружить новые вирусы или видоизмененные старые. Под еще большей угрозой находятся пользователи антивирусных программ, не обладающих проактивной защитой, например, такой, как у антивируса Касперского (рис. 4) или не использующих таких программ вовсе. Таким образом, важность разрабатываемого комплекса невозможно переоценить. Однако для его успешной реализации необходимо не только решать существующие проблемы, но и предугадывать поведение компьютерных злоумышленников. Именно поэтому создание тестовых программ является незаменимой частью в процессе данной разработки.

Литература

1. Руководящий документ. Защита от несанкционированного доступа к информации. Часть 1. Программное обеспечение средств защиты информации. Классификация по уровню контроля отсутствия недекларированных возможностей. М.: Гостехкомиссия России, 1998.
2. Немолочнов О.Ф., Раков С.В. Автоматизация логического проектирования. Учебное пособие. СПб: СПбГУ ИТМО, 1995, 60 с.
3. Peter Sezor The art of computer virus research and defense // Addison Wesley Professional, February 03 2005, 744 p.

ФУНКЦИОНАЛЬНАЯ МОДЕЛЬ СИСТЕМЫ УПРАВЛЕНИЯ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТЬЮ КАК СРЕДСТВО ВНЕДРЕНИЯ СТАНДАРТОВ ЛИНЕЙКИ ISO/IEC 2700X (BS 7799)

Н.В. Андреева
Научный руководитель – к.т.н., доцент А.В. Любимов

В статье представлено краткое описание линейки стандартов British Standards Institute в области управления информационной безопасностью, обоснование выбора одного из стандартов в качестве основного (ведущего), актуальность и необходимость формального моделирования системы управления информационной безопасностью (СУИБ) по требованиям выбранного стандарта, а также дан краткий обзор базовых параметров функциональной модели СУИБ.

Введение

На данный момент в условиях постоянного усложнения информационных систем, а также расширения сферы их использования уже почти невозможно обойтись одноразовым построением системы информационной безопасности – без обучения персонала, анализа меняющихся рисков и учета других важных факторов она будет малоэффективна. Таким образом, кроме построения системы информационной безопасности, в организации необходимо также создать систему *управления* информационной безопасностью, которая будет являться частью общей, интегрированной системы управления организацией (Integrated Management Systems, IMS).

Линейка стандартов в области управления информационной безопасностью изначально была разработана BSI (British Standards Institute, Британский Институт Стандартов), после чего данные стандарты прошли адаптацию в ISO (International Standards Organization, Международная Организация по Стандартизации), получив статус международных.

Построение функциональной модели СУИБ на основе требований одного из этих стандартов будет весьма полезно для продвижения и эффективного применения всей линейки.

Обзор зарубежных и российских источников не выявил разработок, подобных данной. Среди моделей, построенных на основе других стандартов в области безопасных информационных технологий, можно назвать:

- среди зарубежных – комплексную модель процесса оценки информационных технологий по Общим критериям [1];
- среди отечественных – модель Системы Менеджмента Качества по стандарту ISO 9001 [2] и формальную модель процессов оценки безопасности информационных технологий по методологии Общих критериев [3]. В обеих отечественных моделях для построения используется методика SADT в совокупности с методом DFD.

Состав линейки стандартов BSI по управлению информационной безопасностью

Большой вклад в разработку международных стандартов по управлению организацией в целом и созданию интегрированной системы управления внес Британский Институт Стандартов (British Standards Institute, BSI). В этом учреждении были разработаны стандарты-прототипы серий ISO 900x (управление качеством), 1400x (управление окружающей средой), 2000x (управление IT-сервисами) и, конечно, 2700x (управление информационной безопасностью), которые были приняты на основе аналогичных стандартов линейки BS 7799.

Стандарт BS 7799 был утвержден в качестве государственного стандарта Великобритании и, в конце концов, получил общемировую известность. На данный момент он состоит из трех частей, каждая из которых была принята в качестве самостоятельного стандарта (таблица) [4].

Стандарт BSI			**Соответствующий стандарт ISO/IEC**		
Индекс	Название	Год принятия	Индекс	Название	Год принятия
7799-1	Code of Practice for Information Security Management	1995	17799 (27002)	Information technology – Code of practice for information security management	2000, 2005 (запланирован на 2007)
7799-2	Information Security management – Specification for ISMS	1999	27001	Information technology – Security techniques – Information security management systems – Requirements	2005
7799-3	Information security management systems. Guidelines for information security risk management	2006	27005	–	Запланирован на 2007

Таблица. Линейка стандартов BS 7799 (ISO/IEC 2700x)

Стандарт BS 7799-1 (Практические правила управления информационной безопасностью) был разработан в качестве практических рекомендаций по организации системы управления информационной безопасностью в любой (независимо от типа) компании.

Вторая часть – BS 7799-2 (Спецификация системы управления информационной безопасностью) – определяет спецификацию СУИБ, включая обязательные требования к ее созданию, внедрению, эксплуатации, мониторингу, анализу, поддержке и улучшению, необходимые для сертификации.

BS 7799-3 (Руководство по управлению рисками информационной безопасности). предназначен для определения основных факторов риска и подходов к его оценке и обработке.

Специалисты считают, что BS 7799 не является техническим стандартом, но зато дает компании инструмент, позволяющий управлять конфиденциальностью, целостностью и сохранностью важного актива компании – информацией, и, к тому же, может использоваться для защиты любых видов информации, включая финансовую информацию, кадровую информацию, информацию по поставщикам, любые другие данные компании и, что немаловажно, информацию, принадлежащую партнерам / клиентам организации [4]. Стандарт намеренно разрабатывался именно таким – предназначенным для любых видов организаций и внешних окружений документом по управлению информационной безопасностью, а не каталогом ИТ продуктов [5].

Конечной практической целью рассмотренной линейки стандартов является построение СУИБ в конкретных организациях. Из трех рассмотренных стандартов всесторонне этот процесс представлен в стандарте BS 7799-2 (ISO/IEC 27001), в нем полностью описан требуемый результат, и именно на соответствие его требованием проводится сертификация СУИБ.

Стандарт BS 7799-3 (ISO/IEC 27005) выпущен недавно и еще не успел приобрести подобной популярности, к тому же в нем рассматривается только система управления

рисками, а не вся СУИБ. Первая же часть стандарта BS 7799 (ISO/IEC 17799) является, по сути, лишь сборником практических рекомендаций, помогающим определить для себя принципы построения СУИБ и представить в общем, чем должна обладать организация для эффективного управления информационной безопасностью.

Однако при практическом подходе к построению СУИБ в организации ее функционирования использование только текста стандарта BS ISO/IEC 27001:2005 является неудобным из-за отсутствия наглядности. С другой стороны, данный стандарт является средством регламентации всей деятельности по управлению информационной безопасностью организации, а существующие методы и средства системного анализа позволяют представить функциональную структуру любой регламентированной деятельности в формализованной графической форме, являющейся при этом весьма сжатой и наглядной.

Таким образом, использование функциональной модели СУИБ позволяет наглядно представить СУИБ в контексте общей системы управления организацией, с установленными логическими связями между всеми составляющими.

Актуальность формального моделирования СУИБ

Под моделированием подразумевается метод исследования объектов различной природы на их аналогах (моделях) для определения или уточнения характеристик существующих или вновь конструируемых объектов [6]. Моделирование различных информационных систем и систем управления в настоящее время очень популярно, поскольку позволяет организовать эффективное управления за счет обеспечения оптимальной связи бизнеса и информационных технологий.

Функциональное моделирование СУИБ организации необходимо для наглядного представления ее в целом, в контексте общей системы управления организацией – с установленными логическими связями между всеми составляющими. Данную модель можно использовать также для проектирования, разработки и сопровождения программного обеспечения поддержки деятельности по управлению информационной безопасностью организации. Она будет полезна для обучения сотрудников организации, установления их ролей в части управления информационной безопасностью и разработки должностных инструкций. Отдельно хотелось бы отметить, что функциональная модель СУИБ поможет получить представление о том, что необходимо для прохождения сертификации по данному стандарту.

Сертификация СУИБ на соответствие требованиям BS ISO/IEC 27001 в настоящее время является достаточно популярной – как в мире в целом, так и в России, в частности – поскольку посредством нее организация дает возможность своим внешним партнерам, инвесторам и клиентам убедиться в том, что подсистема информационной безопасности построена правильно и функционирует эффективно, что, несомненно, повышает конкурентоспособность компании [4]. Сертификацию СУИБ организации на соответствие требованиям BS ISO/IEC 27001:2005 имеют право проводить только организации, имеющие аккредитацию Аккредитационной службы Великобритании (United Kingdom Accreditation Service, UKAS). Соответственно, компания, прошедшая данную сертификацию, получает официальный сертификат международного образца и регистрацию в едином реестре UKAS, что гарантирует ее международное признание.

Согласно данным группы пользователей СУИБ, поддерживающей международный реестр сертификатов, по состоянию на август 2006 года в мире зарегистрировано более 2800 организаций из 66 стран, сертифицированных по ISO 27001 (BS 7799-2), в том числе и четыре российские компании. Среди сертифицированных организаций – крупнейшие ИТ-компании, организации банковской и финансовой сферы, предприятия ТЭК и телекоммуникационного сектора.

Ожидается, что количество обладателей сертификатов в России в 2007 году достигнет нескольких десятков [5]. Росту популярности сертификации по стандарту BS 7799-2 в России, скорее всего, будет способствовать и введение в действие с 1 января 2007 года ГОСТа Р ИСО/МЭК 17799-2005, а также ожидаемого принятия ГОСТа Р ИСО/МЭК 27001 [7, 8]. В связи с этим уже сейчас существует необходимость подготовки значительного числа специалистов по BS 7799 как для центров оценки и сертификации, так и для организаций-пользователей.

Объемы работ по оценке и сертификации СУИБ достаточно велики – строгая регламентация позволяет автоматизировать выполнение многих действий по построению СУИБ, а функциональные спецификации в стандартизованной электронной форме будут полезны для проектирования, разработки и сопровождения инструментальных программных средств поддержки деятельности по управлению информационной безопасностью организации [3].

Таким образом, представление базовых положений стандарта BS 7799-2 и их взаимосвязей в единой структурированной форме – в виде формальных моделей, а также разработка системы таких моделей является необходимым условием как для продвижения стандарта, так и для его эффективного применения. Начинать разработку системы моделей можно с функциональной модели – для того, чтобы определить и наглядно представить взаимосвязи между функциями (процессами) СУИБ, после чего станет возможным построение структурной (для отражения информационной структуры системы процессов) и математической моделей.

Обзор базовых параметров функциональной модели СУИБ

В современных методах формального моделирования решающее влияние на результирующую модель оказывает выбор ее базовых свойств (назначение, точка зрения, границы моделирования и т. д.), а также ее контекста и нотации. Функциональное моделирование СУИБ предлагается выполнять по хорошо себя зарекомендовавшей методике создания структурированных моделей деятельности в некоторой предметной области – SADT, которая подразумевает графическое представление блочного моделирования. Модель SADT представляет собой серию диаграмм с сопроводительной документацией, разбивающих сложный объект – систему – на составные части.

В качестве инструментального средства моделирования предлагается использовать классический инструмент функционального моделирования – продукт All Fusion Process Modeler (BPwin 4.0) компании Computer Associates. All Fusion Process Modeler поддерживает сразу три нотации моделирования: бизнес-процессов (IDEF0), потоков работ (IDEF3) и потоков данных (DFD). Для построения разработанной модели выбран стандарт DFD, одним из преимуществ которого является возможность отражения связей СУИБ организации с внешними (по отношению к моделируемой системе) заинтересованными сторонами, что необходимо для соответствия модели PDCA («Plan-Do-Check-Act» – Цикл: Создание – Внедрение и эксплуатация – Мониторинг и анализ – Поддержка и улучшение), которая используется в стандарте BS ISO/IEC 27001:2005 [9].

Базовыми понятиями методологии SADT являются цель и точка зрения моделирования, которые составляют основное свойство модели – ее назначение [10]. Основная цель моделирования – описание процессов создания и функционирования СУИБ и их взаимодействия в соответствии со стандартом BS ISO/IEC 27001:2005.

Функциональную модель целесообразно строить с точки зрения потенциального разработчика СУИБ – руководителя отдела Информационной безопасности. Она содержит представление взаимосвязанных процессов создания, внедрения, эксплуатации, мониторинга, анализа, поддержки и улучшения СУИБ. В модели СУИБ представляется полезным выполнить максимальную детализацию процессов – в соответствии с пунк-

тами и подпунктами стандарта BS ISO/IEC 27001:2005. Процессы в разработанной модели можно условно разделить на «основные», соответствующие модели PDCA, и «дополнительные», выходящие за рамки этой модели, но рассмотренные в стандарте.

Заключение

Система управления информационной безопасностью требуется во многих организациях, поскольку сама по себе разработанная система защиты информации малоэффективна без учета и анализа множества изменяющихся факторов, влияющих на нее. Стандарты в области управления информационной безопасностью были разработаны в Великобритании и затем адаптированы в ISO. Среди этих стандартов в качестве основного можно выделить BS ISO/IEC 27001:2005 поскольку именно в нем всесторонне, с описанием требуемых результатов представлен процесс организации СУИБ.

BS ISO/IEC 27001:2005 приобрел достаточную популярность, но еще не используется повсеместно. Одной из возможных причин этого (помимо малодоступности его для бесплатного ознакомления) является отсутствие наглядности при использовании текста стандарта в практических целях – для построения СУИБ в конкретной организации. Устранить этот недостаток можно с помощью разработки функциональной модели СУИБ, которая также будет полезна, например, для подготовки специалистов в области консультационного и сертификационного аудита СУИБ.

Литература

1. Prieto-Diaz, R. The Common Criteria Evaluation Process. Process Explanation, Shortcomings, and Research Opportunities. Commonwealth Information Security Center Technical Report CISC-TR-2002-03, 2002 – CISC, James Madison University, USA.
2. Любимов А.В. Модели процессов СМК по стандарту ISO 9001:2000. Препринт кафедры Распределенных вычислений и компьютерных сетей. СПб: СПбГТУ, 2004.
3. Любимов А.В. Функциональная структура общих критериев оценки безопасности информационных технологий. / Труды 9-й научно-технической конференции «Теория и технология программирования и защиты информации. Применение вычислительной техники». Санкт-Петербург, 18 мая 2005. С. 20–24.
4. Горобец Н.И. BSI и BS 7799 – Видение разработчиков. 2005. http://www.globaltrust.ru/security/Pubs/Pub10_NIG_BS7799.htm
5. Астахов А. BS 7799 – прародитель международных стандартов. // Средства защиты информации и бизнеса 2006. Аналитический обзор. – CNews Analytics online, 2006. http://www.cnews.ru/reviews/free/security2006/articles/bs/
6. Азимов Э.Л., Щукин А.И. Словарь методических терминов (теория и практика преподавания языка). СПб: Златоуст, 1999. 472 с.
7. С 1 января 2007 года вводится в действие ГОСТ Р ИСО/МЭК 17799-2005 "Информационная технология. Практические правила управления информационной безопасностью". // Пресс-релиз компании «Электронные офисные системы» от 01.12.2006. http://eos.ru/eos/244730
8. Панасенко Е. Законодательство и регулирование отрасли ИБ // Средства защиты информации и бизнеса 2006. Аналитический обзор. CNews Analytics online, 2006. http://www.cnews.ru/reviews/free/security2006/articles/legislation
9. Компания "Интерфейс Ltd.". О BPwin 4.0. 2001. http://www.interface.ru/home.asp?artId=2736.
10. Марка Д.А., МакГоуэн К. Методология структурного анализа и проектирования SADT. Электронная библиотека, 1999. http://www.interface.ru/fset.asp?Url=/case/sadt0.htm

АНАЛИЗ И ИССЛЕДОВАНИЕ МЕТОДОВ И СРЕДСТВ ОБНАРУЖЕНИЯ НЕДЕКЛАРИРОВАННЫХ ВОЗМОЖНОСТЕЙ

О.Д. Темнов
Научный руководитель – д.т.н., профессор Л.Г. Осовецкий

В статье рассмотрены существующие подходы к обнаружению недекларированных возможностей. В качестве решения проблемы их гарантированного обнаружения предлагается использовать комплексные кубические покрытия.

Введение

Основная особенность информационных технологий состоит в том, что их определяющим компонентом является программное обеспечение. И именно оно выступает источником угроз информационной безопасности в большинстве случаев. Значительная часть ошибок в программном обеспечении закладывается не на этапе проектирования программ, когда проектировщик мыслит абстрактными категориями в привычной для человека форме, а на этапе разбивки исходной задачи на подзадачи и их кодирования в выбранном языке программирования. Кроме этого, программное обеспечение может содержать функциональные возможности, не описанные или не соответствующие описанным в документации, при использовании которых возможно нарушение конфиденциальности, доступности или целостности обрабатываемой информации. Именно такие функциональные возможности понимаются под недекларированными возможностями в [12].

Для предоставления гарантий защищенности данных, циркулирующих в программном обеспечении, производят его сертификацию. Наиболее сложной и трудоемкой областью при сертификации является исследование программного обеспечения на наличие недекларированных возможностей, поиск закладных элементов и анализ уязвимых мест. В большинстве испытательных лабораторий исследования на наличие недекларированных возможностей осуществляется путем «ручного» анализа исходных текстов программ. Ввиду большой сложности современного программного обеспечения, на проведение таких работ затрачивается большое время, исчисляемое месяцами, и требуется привлечение значительного числа высококвалифицированных специалистов, что определяет очень высокую стоимость выполнения указанных работ и практическую невозможность их повторения.

Для ускорения времени анализа программного обеспечения на наличие недекларированных возможностей существуют различные инструментальные средства. Но у каждого из них есть свои недостатки. Общим же недостатком является то, что данные средства не дают гарантий обнаружения недекларированных возможностей. В работе предлагается использовать комплексные кубические покрытия для задач гарантированного обнаружения недекларированных возможностей.

Методы обнаружения недекларированных возможностей

Подходы к уменьшению ошибок в программном обеспечении можно сгруппировать в две категории: уменьшающих ущерб от ошибок или удаляющих сами ошибки. Наглядно эти группы представлены на рис. 1.

Ограничение ущерба

Способы ограничения ущерба от ошибок программного обеспечения включают модификацию уже готовых программ, вставляющую в них проверки различных условий, или запуск приложений в особом окружении, ограничивающем их возможности. Другие проекты заключаются в том, чтобы разобрать безопасные библиотеки или мо-

дификации компилятора специально для предотвращения переполнения буфера. Эти подходы требуют минимальных усилий от разработчиков программ.

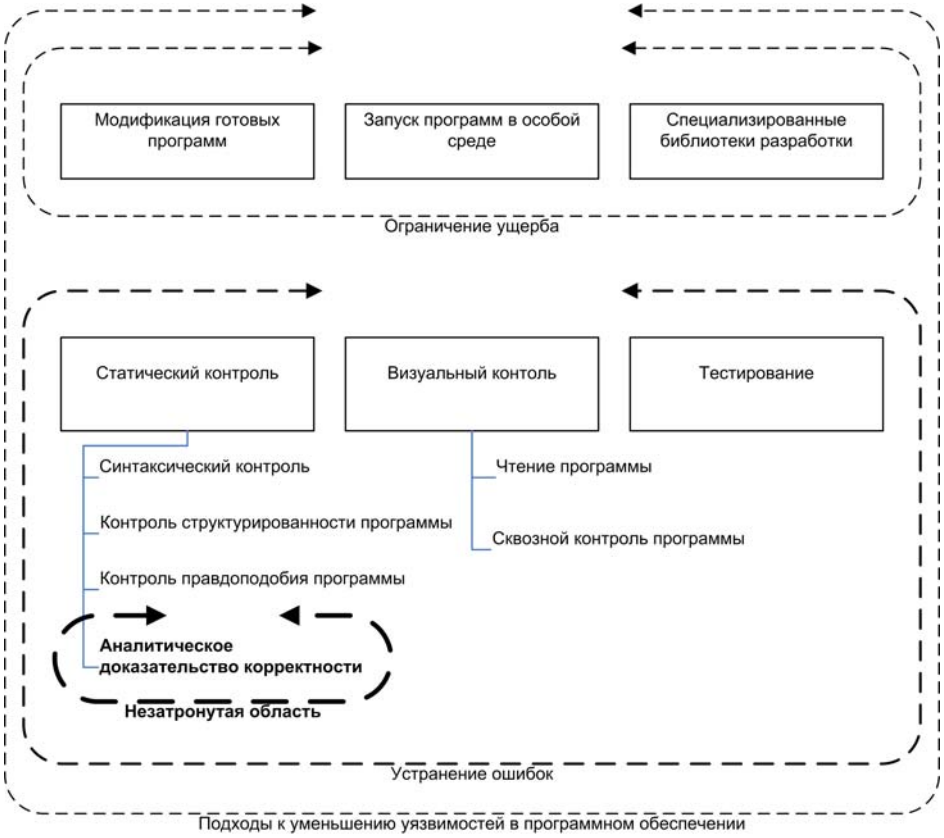


Рис. 1. Подходы к уменьшению уязвимостей в программном обеспечении

Первым недостатком таких подходов является снижение производительности. Однако наиболее существенным недостатком является то, что эти подходы не устраняют ошибки, а заменяют их отказом в доступе. Восстановление после обнаруженной проблемы обычно требует завершения программы. Хотя критичные к ошибкам безопасные системы должны применять способы ограничения ущерба, но они не должны заменять подходы, направленные на устранение ошибок.

Устранение ошибок

Способы распознавания и исправления уязвимостей включают в себя тестирование, просмотр исходного текста человеком, статический анализ.

В настоящий момент основными уязвимостями в программном обеспечении выступают недекларированные возможности. Для их гарантированного обнаружения, наряду с другими методами, требуется проводить аналитическое доказательство корректности программ. Отсутствие инструментальных средств существенно усложняет этот процесс, а порой и просто делает его невозможным в связи с большим объемом исходного текста. В настоящий момент разрабатывается инструментальное средство, основанное на комплексных кубических покрытиях, призванное существенно изменить сложившуюся картину.

Комплексные кубические покрытия

Основная сложность вычислительных процессов зависит не от количества линейных операторов, а от числа безусловных и условных точек ветвления процесса, которые

и определяют логику принятия решений. Все команды конкретного процессора делят на две категории:

- команды обработки данных (команды пересылки, арифметические команды, команды сдвигов, команды обработки десятичных знаков и т.д.);
- команды управления последовательностью управления (безусловные и условные переходы, команды вызова процедур и команды возврата).

Таким образом, программа P может быть представлена в виде булева графа, содержащего линейные и условные вершины. Линейная вершина содержит одну точку входа и одну точку выхода, это может быть либо одна линейная команда (т.е. команда, не являющаяся командой передачи управления), либо последовательность линейных команд. Условная вершина имеет одну точку входа и две точки выхода, задающие адреса ветвления (две ветви) в зависимости от выполнения или невыполнения условия, задаваемого в вершине.

Обозначив подобный граф $BG(P)$, рассматривают всевозможные пути выполнения программы, которые будут соответствовать путям на графе $BG(P)$, соединяющим начальную и конечную вершину. Каждый путь l на графе $BG(P)$ состоит из последовательности условных и линейных вершин. Условные вершины определяют потоки управления или логику программы, а линейные вершины определяют вычисление переменных или выполнение других линейных операций.

Путь l можно задать булевыми переменными, описывающими условия прохождения данного пути и принимающими значение 1 или 0 в зависимости от выполнения или невыполнения условия. Получив таким образом логические условия, заданные булевыми переменными, можно построить кубическое покрытие $C(P) = \{c_1, c_2, ..., c_n\}$, где каждый куб c_i содержит в себе условия, определяющие путь l_i на графе $BG(P)$. Координаты куба c_i могут принимать значения 0, 1 и x (значение x соответствует условию, не влияющему на формирование данного пути). Покрытие $C(P)$ полностью соответствует определению покрытия в исчислении кубических комплексов, поэтому к покрытиям $C(P)$ и кубам c_i применимы алгебро-топологические операции: пересечение ($\cap$), звездчатое произведение (*) и вычитание (#) [8]. Эти операции позволяют устанавливать соотношения включения, поглощения и эквивалентности между различными покрытиями.

Покрытие $C(P)$ содержит в себе логическую часть программы, т.е. наборы логических условий, определяющих пути выполнения программы. При дополнении покрытия $C(P)$ множеством, определяющим результат работы программы для каждого пути, получают так называемое комплексное кубическое покрытие $\widetilde{C}(P)$. В комплексном кубическом покрытии каждому кубу C_i из покрытия $C(P)$ ставится в соответствие множество R_i, содержащее результат выполнения программы. В широком смысле множество R_i содержит линейную последовательность операторов, через которые проходит путь l_i. В более узком определении R_i может быть формулой или набором формул, определяющих выходную переменную функции или глобальную переменную программы [11].

Применение комплексных кубических покрытий

Кроме уже указанной выше цели, область применения комплексных кубических покрытий можно разделить на три направления, представленных на рис. 2.

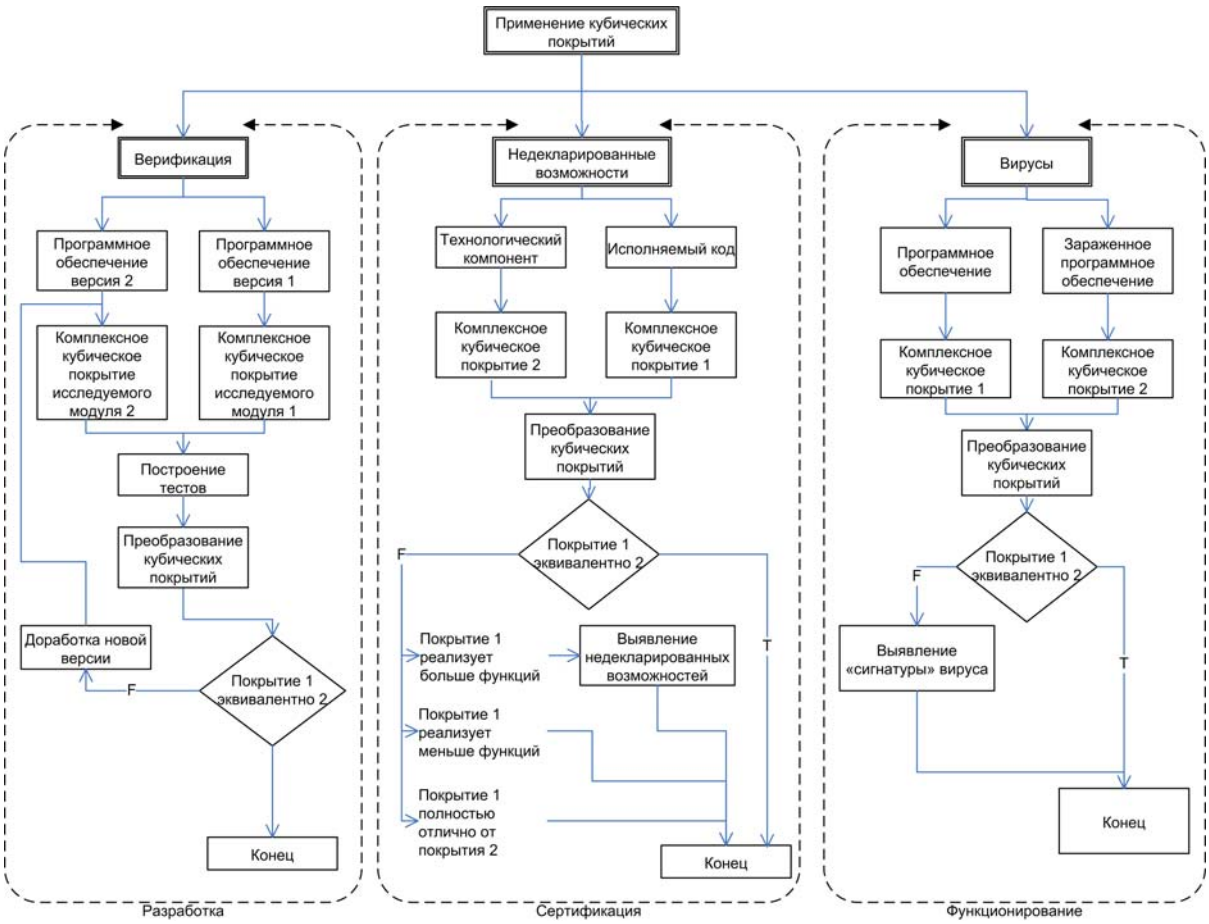


Рис. 2 Применение кубических покрытий

Верификация

Под верификацией понимается установление соответствия между различными программами. Необходимость в этом действии может возникнуть по следующим причинам:

- иногда программный продукт может разрабатываться различными группами проектантов. В этом случае можно верифицировать различные варианты реализаций между собой для повышения объективности и качества выбора конкретной версии;
- в ходе жизненного цикла программа может подвергаться различным модификациям – обновление версии или устранение ошибки. Верификация в данном случае позволяет зафиксировать наличие и/или устранение ошибки, подтвердить идентичность логики программ, установить измененные элементы логики.

Выявление недекларированных возможностей

В данном направлении сравнивается кубическое покрытие логических процессов программ, полученное по исполняемому коду программы, с кубическим покрытием, построенным на основе полной спецификации программы. Основная трудность данного направления заключается в построении кубического покрытия по спецификации программы. Важную роль в данном случае играет выбранная при проектировании технология программирования, так как она определяет правила описания (кодирования) процессов.

Для конкретной технологии программирования можно создать предопределенный репозиторий кубических покрытий процессов. Впоследствии, объединяя кубические покрытия небольших программных модулей, можно переходить к более крупным модулям.

Данное направление требует автоматизации процесса построения кубических покрытий по спецификации. В настоящий момент ведутся разработки автоматизации процесса построения кубических покрытий по спецификациям, созданными при помощи инструментальных средств IBM Rational.

Исследование вирусов

Как и при поиске недекларированных возможностей, исследуемым объектом является логика работы программ. Предполагается использовать разницу кубических покрытий зараженного и «чистого» экземпляров программы. Найденная разница отображает логику вируса. В дополнении с данными о командах обработки данных назовем ее сигнатурой вируса. Используя подобные сигнатуры, можно:

- исследовать программы на наличие вирусов;
- исследовать логику работы вируса.

Данное направление требует отдельных исследований и дополнительных экспериментальных данных.

Заключение

В настоящее время основной уязвимостью в программном обеспечении является наличие недекларированных возможностей. Ни одно из существующих инструментальных средств поиска недекларированных возможностей не дает гарантий их обнаружения. Применение комплексных кубических покрытий позволит гарантированно решить данную задачу. Кроме этого существуют перспективы использования кубических покрытий в других областях безопасности информации.

Литература

1. Грис Д. Наука программирования. М.: Мир, 1984.
2. Жоголев Е.А. Технология программирования. М.: Мир, 1990.
3. Компаниец Р.И. Инструментальные средства анализа и защиты программного кода от использования недокументироаннных возможностей / Конференция «Защита информации в современных условиях – основа сохранения и развития бизнеса», 2006.
4. Лаздин А.В., Немолочнов О.Ф. Метод построения графа функциональной программы для решения задач верификации и тестирования. // Научно-технический вестник СПб ГИТМО (ТУ). Выпуск 6. Информационные, вычислительные и управляющие системы / Гл. ред. В.Н. Васильев. СПб: СПбГИТМО (ТУ), 2002.
5. Лаздин А.В., Немолочнов О.Ф. Оценка сложности графа функциональной программы. // Научно-технический вестник СПбГИТМО (ТУ). Выпуск 6. Информационные, вычислительные и управляющие системы / Гл. ред. В.Н. Васильев. СПб: СПбГИТМО (ТУ), 2002.
6. Липаев В.В. Отладка сложных программ. Методы, средства, технология. М.: Энергоатомиздат, 1993.
7. Марков А.С., Миронов С.В., Цирлов В.Л. Выявление уязвимостей в программном коде // Открытые системы, 2005. № 12.
8. Немолочнов О.Ф. Методы технической диагностики. Методические указания к курсовой работе. Л.: ЛИТМО, 1976.
9. Немолочнов О.Ф., Зыков А.Г., Поляков В.И. Комплексные кубические покрытия и графо-аналитические модели как средство описания вычислительных процессов программ / Международная конференция «Интеллектуальные системы», 2006.

10. Немолочнов О.Ф., Зыков А.Г., Поляков В.И. Кубические покрытия логических условий вычислительных процессов и программ. // Научно-технический вестник СПбГУ ИТМО. Выпуск 14. Информационные технологии, вычислительные и управляющие системы / Гл. ред. В.Н. Васильев. СПб: СПбГУ ИТМО, 2004.
11. Сидоров А.В. Верификация вычислительного процесса программы с использованием комплексных кубических покрытий / II межвузовская конференция молодых ученых. СПб: СПбГУ ИТМО, 2005.
12. Гостехкомиссия России. Руководящий документ. Защита от несанкционированного доступа к информации. Часть 1. Программное обеспечение средств защиты информации. Классификация по уровню контроля отсутствия недекларированных возможностей. М.: Гостехкомиссия России, 1999.
13. Концептуальные вопросы оценки безопасности информационных технологий [Электронный ресурс] / Jet Info; ред. Трубачев А., Режим доступа: http://www.jetinfo.ru/1998/5-6/2/article2.5-6.1998.html, свободный. Загл. с экрана.
14. Инструментальные средства анализа и защиты программного кода от использования недекларированных возможностей [Электронный ресурс] / ООО "НовоБИТ", Компаниец Р.И., Режим доступа: http://www.novobit.nov.ru/present/1.ppt, свободный. Загл. с экрана.
15. CERT Vulnerabilities Statistics [Электронный ресурс] / Cert Coordination Center, Режим доступа: http://www.cert.org, свободный. Загл. с экрана.
16. CVE version: 20061101 [Электронный ресурс] / Common Vulnerabilities and Exposures, Режим доступа: http://cve.mitre.org, свободный. Загл. с экрана.
17. Statistics Query Page [Электронный ресурс] / National Vulnerability Database, Режим доступа: http://nvd.nist.gov/statistics.cfm, свободный. Загл. с экрана.

АНАЛИЗ СЕТЕВОГО ТРАФИКА КОРПОРАТИВНОЙ СЕТИ ПО ПРОТОКОЛУ SMTP НА ПРЕДМЕТ УТЕЧЕК ИНФОРМАЦИИ

А.И. Спивак

Научный руководитель – д.т.н., профессор Л.Г. Осовецкий

Контроль за почтовым трафиком сети должен осуществлять специалист по защите информации. Существующие системы контроля работают при условии переконфигурирования почтового сервера сети. Представлен продукт, работающий на уровне работы сети, а не на уровне работы почтовой программы.

Введение

Постоянное совершенствование и развитие области информационных технологий вызывает повышение степени сложности данной сферы человеческой деятельности. Это, несомненно, сказывается и на обеспечении безопасности. Большое количество новых технологий уже невозможно контролировать руками одного, а подчас и нескольких человек. Усложнение вызвано не только появлением новых технологий, но и расширением области применения старых. Используемые информационные инфраструктуры в настоящее время включают в себе самые различные механизмы и способы взаимодействия различных систем. Все это позволяет достигать наиболее эффективного решения поставленных перед ними задач, а также быструю адаптацию к решению новых задач. Наравне с плюсами такого построения инфраструктуры есть и сложности. Прежде всего, это учет всех механизмов взаимодействия, которых может быть много в таких гетерогенных системах. Их контроль должен быть таким же тщательным, как и в типовых схемах построения информационных систем. Безопасность в данном случае должна обеспечиваться с применением различных систем. Многие инструменты решения этих задач уже представлены широкой общественности и используются специалистами по защите информации. Но не во всех направлениях обеспечения безопасности имеются необходимые средства. В работе будет представлено средство контроля за сетевым трафиком по протоколу SMTP.

Почтовые сообщения являются одним из широко используемых средств обмена информации между людьми. Контроль за данным каналом обмена информации также должен осуществляться специалистом по обеспечению безопасности. Важно отметить, что существующие в данный момент средства анализа сетевого трафика не обладают необходимым набором функций для удовлетворения всех потребностей для контроля. Сетевые сниферы позволяют только обнаружить факт передачи информации от одного узла сети к другому без дополнительной информации о содержимом данного взаимодействия. Они производят перехват пакетов и вывод информации об адресах и портах узлов, участвующих в передаче. Этого, как правило, недостаточно для принятия решения о безопасности данного взаимодействия. Сложность заключается в отделении потоков, которые представляют собой вполне законные почтовые сообщения, от возможных утечек информации. Для этого необходимо сосредоточить свое внимание только на одном сетевом протоколе и производить доскональный анализ циркулирующей по нему информации.

Методы анализа почтового трафика

Перехват можно осуществлять в нескольких принципиально отличных точках – на уровне работы сети и на уровне работы приложений. В первом случае должен производиться анализ трафика, а во втором необходимо вмешиваться в логику работы почтовой программы с целью реализации механизма проверки почтовых сообщений на предмет утечек информации. Как правило, необходимо переконфигурирование почто-

вого сервера, чтобы он передавал проходящую почту контролирующей программе. При этом надо учитывать, что многие приложения имеют закрытый исходный код и не предоставляют интерфейса, позволяющего каким-либо образом получить доступ к почтовым сообщениям, проходящим сквозь них. Существуют программы с открытым исходным кодом, а также программы с реализацией вышеупомянутого интерфейса. В обоих случаях достаточно сложно реализовать контроль за содержимым почтовых сообщений, так как нужно вмешиваться в работу сторонней программы и учитывать нюансы ее работы. У данного метода существует еще ряд недостатков. Контроль будет осуществляться только за тем почтовым трафиком, который проходит через почтовый сервер с системой анализа. Ведь существует возможность отправки напрямую, минуя почтовый сервер, на котором производится контроль трафика. Правда, этому можно воспрепятствовать, правильным образом сконфигурировав межсетевой экран. Другим недостатком можно назвать зависимость работы всей почтовой системы от работы программы анализа – сбой приведет к остановке функционирования почтового сервиса в рамках всей организации. Существуют у данного метода и достоинства. Это, прежде всего, возможность блокировки отсылки письма, которое вызвало подозрение у системы анализа. Другим плюсом является гарантированная проверка всех писем, проходящих через сервер. Но, несмотря на это, перехват сетевого трафика на уровне работы сети представляется более целесообразным. Преимущества данного метода заключаются в том, что не нужно ориентироваться на логику сторонней программы, а тем более вносить изменения в конфигурацию почтового сервера. Контролирующую машину (если используется выделенная машина для контроля) можно защитить гораздо сильнее, чем почтовый сервер, вплоть до того, что вообще не присваивать ей IP-адрес, чтобы полностью исключить удаленную работу с ней. Кроме того, она требует практически нулевого времени на обслуживание. При необходимости можно исключить доступ даже сотрудников отдела администрирования к серверу мониторинга. При использовании данного метода можно вообще исключить риск потери функциональности почтовой системы в случае любых сбоев в системе контроля. Также немаловажную роль играет возможность внедрения практически в любую точку, где проходит почтовый трафик информационной системы, что позволяет масштабировать и легко переносить рассматриваемую в проекте систему. Недостатком можно назвать невозможность заблокировать отсылку письма, которое можно интерпретировать как утечку информации. В случае использования слабого или сильно загруженного сервера для мониторинга и высокого объема почтового трафика (например, в пиковые моменты), некоторые сообщения могут быть пропущены монитором [1].

Использоваться могут оба рассмотренных метода, и выбор их всецело зависит от условий работы в конкретном случае. Разработанная система анализа почтового трафика является примером сетевого подхода к контролю пересылки корреспонденции.

Самой большой группой продуктов по мониторингу сетевого трафика, выполняющих схожие по функциям действия, являются антивирусы, особенно антивирусы для работы с почтовым трафиком. Использование фильтра Milter позволяет прослушивать SMTP-диалог и модифицировать SMTP-ответы сервера Sendmail. Другой группой являются разнообразные сниферы, работающие под управлением операционной системы Windows и обладающие широким спектром возможностей, в частности, выделение из трафика сообщений электронной почты.

Прямыми коммерческими аналогами являются: InfoWatch Mail Monitor, ДозорДжет, MIMEsweeper for SMTP. Отличие от разрабатываемой системы у данной группы продуктов одинаковое – необходимость переконфигурирования почтового сервера для перенаправления всего почтового трафика на компьютер с установленной системой мониторинга.

Описание работы программы

Наиболее предпочтительной для разработки операционной системой является Unix за счет стабильности, повсеместного использования в качестве сетевых сервисов, производительности, отказоустойчивости и прозрачности работы, а также наличия исходных текстов самой операционной системы.

Реализацию в среде Unix можно производить, основываясь на нескольких способах перехвата сетевого трафика. Выделим следующие методы: на уровне коммутаторов, bpf, libpcap, RAW-сокеты, divert-сокеты, система протоколирования пакетных фильтров (ULOG), kernel modules. Библиотека libpcap была выбрана для написания программного продукта как наиболее подходящая и удовлетворяющая всем требованиям. Главными можно назвать следующие: работа на пользовательском уровне, для исключения возможности краха системы вследствие ошибки в программе, широкое использование (библиотека использовалась для написания таких известных программ, как tcpdump и snort), отсутствие необходимости реализации многих функций обработки – они уже заложены в саму библиотеку.

Функционирование программы осуществляется в соответствии со схемой (рис. 1).

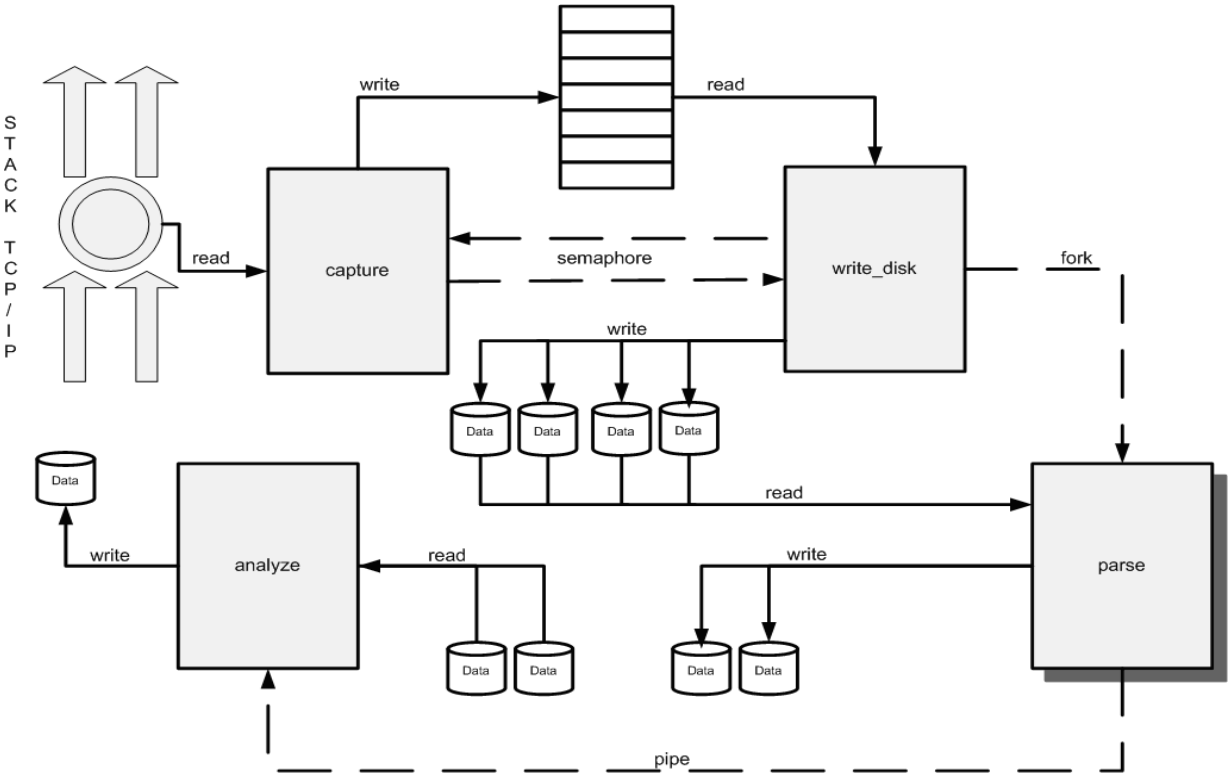


Рис. 1. Схема работы программы

Рассмотрим подробно каждый компонент программного продукта.

Основываясь на том предположении, что многозадачность в среде UNIX, как правило, реализуется наличием нескольких процессов, данная программа выполнена в виде нескольких демонов.

Программа представляет собой консольное приложение, которому необходимо при запуске передать два параметра: директорию для хранения промежуточных и конечных данных, полученных в ходе перехвата трафика, и файл в этой директории с записанными в нем шаблонами для поиска ключевых слов.

Затем производится создание процесса, который представляет собой демон захвата трафика, для этого выполняется ряд действий. Организуются собственная группа и сеанс, не имеющий управляющего терминала. Это позволяет демону избавиться от сиг-

налов, генерируемых терминалом (SIGINT или SIGHUP), например, при нажатии определенных клавиш или выходе пользователя из системы. Лидером группы и сеанса может стать процесс, если он еще не является лидером. Поскольку, как правило, предыстория запуска данной программы неизвестна, необходима гарантия, что процесс не является лидером группы. Для этого порождается дочерний процесс. Так как его PID уникален, то ни группы, ни сеанса с таким идентификатором не существует, а значит, нет и лидера. При этом родительский процесс немедленно завершает выполнение, поскольку он не нужен. Существует еще одна причина необходимости порождения дочернего процесса. Если демон был запущен из командной строки командного интерпретатора shell не в фоновом режиме, последний будет ожидать выполнения демона, и, таким образом, терминал будет заблокирован. Порождение процесс и завершение выполнение родителя имитируют для командного интерпретатора завершение работы демона, после чего shell выведет свое приглашение. Затем дочерний процесс с помощью системного вызова setsid() становится лидером новой группы, сеанса и не имеет ассоциированного терминала. Далее происходит смена текущего каталога на корневой. Если этого не сделать, а текущий каталог, допустим, находится на примонтированной файловой системе, последнюю нельзя будет размонтировать. Самым надежным выбором является корневой каталог, всегда принадлежащий корневой файловой системе [2]. Здесь же происходит открытие однонаправленного канала обмена между двумя родственными процессами: процессом разбора файла сессии и процессом-анализатором.

В связи с тем, что в программе производится интенсивный ввод-вывод, а также чтение с сетевого интерфейса, блокирование чтения записью приводит к потерям пакетов. При использовании механизмов межпроцессного взаимодействия (каналы, FIFO, очереди сообщений) падение производительности связано с тем, что данные, передаваемые с помощью этих объектов, копируются из буфера передающего процесса в буфер ядра и затем в буфер принимающего процесса. Механизм разделяемой памяти позволяет избавиться от накладных расходов передачи данных через ядро, предоставляя двум или более процессам возможность непосредственного получения доступа к одной области памяти для обмена данными. Безусловно, процессы должны предварительно согласовать правила использования разделяемой памяти. Например, пока один из процессов производит запись данных в разделяемую память, другие процессы должны воздержаться от работы с ней. Задача кооперативного использования разделяемой памяти, заключающаяся в синхронизации выполнения процессов, решается с помощью семафоров.

Примерный сценарий работы с разделяемой памятью выглядит следующим образом.

1. Сервер получает доступ к разделяемой памяти, используя семафор.

2. Сервер производит запись данных в разделяемую память.

3. После завершения записи сервер освобождает разделяемую память с помощью семафора.

4. Клиент получает доступ к разделяемой памяти, запирая ресурс с помощью семафора.

5. Клиент производит чтение данных из разделяемой памяти и освобождает ее, используя семафор.

В связи с необходимостью управления доступом к разделяемой памяти создается группа семафоров.

Для вывода отладочной и иной информации в процессе работы используется вывод в специальный системный журнал с помощью функции syslog(), при этом сообщение от демона получает демон журналирования, работающий в любой операционной системе UNIX. Каждая строчка в системном журнале содержит имя программы и мет-

ки: важности и тип сообщения. В данном случае используется метка INFO, DAEMON указывает на то, что сообщение получено от программы, выполняющейся как демон.

Теперь подробнее коснемся каждого компонента программы. Первым порождается процесс-анализатор. В его функции входит открытие файлов, содержащих результаты работы функции parse, и поиск в этих файлах строк по шаблону. Результат выводится в result.tx, при этом он содержит информацию о шаблоне, файле, в котором найдено соответствие, а также строку, содержащую шаблон. Информацию о файле, в котором необходимо производить поиск, процесс получает из pipe, который был создан в процессе инициализации.

Затем создается процесс записи на диск. Его логика такова, что он в цикле проверяет состояние разделяемой памяти, если там имеется хотя бы одна запись, он ее оттуда выбирает и начинает ее обработку. Разделяемая память выполнена в виде очереди: запись производится сверху очереди, а выборка снизу, при достижении максимального указателя очереди указатель на вершину сбрасывается в нуль для создания кольцевого заполнения сегмента памяти. Такая организация позволяет производить выборку в том же порядке, в котором производилась запись (что невыполнимо при создании очереди в виде стэка), а также избавляет от необходимости двигать элементы очереди, как если бы выборка велась только первого элемента очереди.

Выбираемые из очереди элементы представляют собой перехваченные с сетевого интерфейса пакеты. Пакеты с установленными флагами SYN и ACK, а также FIN не обрабатываются, также пропускаются пустые пакеты, они, как правило, представляют собой подтверждения о доставке. При получении пакета с флагами FIN и ACK производится переименование текущего файла smtp сессии и создание процесса для его обработки.

Процесс обработки не является демоном – он вызывается только в том случае, когда нужно обработать файл smtp сессии, т.е. разобрать его на составляющие, а именно тело письма, текст сессии и вложение. При необходимости вложение подвергается перекодировке. Получается, что для каждой сессии порождается свой процесс обработки, что позволяет повысить производительность системы за счет параллельной работы нескольких процессов. Перед завершением процесса он пишет в pipe имя файла, который нужно проанализировать, это сигнализирует процессу анализатору о необходимости проанализировать файл в соответствии с заданными шаблонами.

Процесс захвата трафика представляет собой функцию, используемую для создания цикла обработки захватываемых пакетов. Одним из параметров является указатель на обработчик каждого поступившего пакета. Он сделан максимально эффективным, так как малейшая задержка в обработке пакета вызовет пропуск последующего, что недопустимо. Поэтому в теле функции-обработчика производится только запись в разделяемую память с использованием семафора. Дальнейшая обработка уже производится процессом записи на диск.

Рассмотрим функционирование на примере обработки одного почтового сообщения.

1. Получение потока данных от сетевого интерфейса.
2. Запись пакетов в разделяемую память процессом захвата трафика.
3. Выборка пакетов из разделяемой памяти процессом записи на диск.
4. Запись потока в файл до достижения признака окончания smtp сессии.
5. Порождение процесса обработки сессии с целью разбора файла на данные сессии, тело письма и вложение.
6. Запись в pipe процессом обработки информации о файлах для анализа.
7. Выборка из pipe имени файла процессом анализатором и анализ содержимого файла на наличие ключевых слов.
8. Запись результатов в файл процессом анализатором.

Заключение

Рассмотренный программный продукт реализован как набор демонов, написанных на языке С и функционирующих под операционной системой UNIX. Данной метод проектирования позволил добиться производительности, расширяемости и стабильности, за счет использования преимуществ выбранной для разработки операционной системы.

Представлено описание алгоритма работы программного продукта, с перечнем компонентов и выполняемых ими функций. Использование высокоскоростных методов взаимодействия между компонентами позволило добиться максимальной производительности продукта в целом.

Дальнейшим развитием рассмотренного в работе продукта может быть добавление функциональных возможностей. Использование некоторого механизма, позволяющего добавлять проверяемые при работе протоколы, несомненно, повысит эффективность. Это можно сделать с помощью языка описания протокола, реализация которого должна быть спроектирована в программе. Кроме того, реализация обратной связи позволит предотвращать обнаруженную утечку информации по заданному протоколу проверки.

Дальнейшее развитие программного продукта позволит специалисту по защите информации получать большее количество информации о трафике контролируемой им корпоративной сети.

Литература

1. Мониторинг электронной почты [Электронный ресурс] / ред. Поляков Я., Режим доступа: http://union.kz/ru/biz/bezop/infosec/monitoringl/pda.shtml, свободный. – Загл. с экрана.
2. Робачевский А. Операционная система UNIX // СПб: БХВ-Петербург, 1997. 528 с.

ИССЛЕДОВАНИЕ ВОЗМОЖНОСТИ ИСПОЛЬЗОВАНИЯ КРИПТОПРОВАЙДЕРА КРИПТОПРО JCP ДЛЯ СОЗДАНИЯ ПРОГРАММНЫХ СРЕДСТВ ЭЛЕКТРОННОЙ ЦИФРОВОЙ ПОДПИСИ

И.В. Головков
Научный руководитель – к.т.н., доцент А.В. Птицын

Построение систем защищенного электронного документооборота основано на использовании электронной цифровой подписи (ЭЦП). Базовые криптографические функции реализуются средствами криптопровадеров – службами, предоставляющими криптографические примитивы для формирования и проверки ЭЦП. Представлен программный продукт, реализующий функции нового российского криптопровайдера КриптоПро JCP.

Введение

Последние несколько лет ознаменовались постепенной заменой бумажной технологии обработки информации ее электронным аналогом. Со временем можно ожидать полного вытеснения бумажного документооборота электронным. Однако представление традиционных бумажных документов в виде электронных последовательностей, состоящих из нулей и единиц, обезличивает последние. Защитных атрибутов бумажных документов: подписей, печатей и штампов, водяных знаков, специальной фактуры бумажной поверхности и других – у электронного представления документов нет. Но электронные документы нужно защищать не менее тщательно, чем бумажные. Поэтому возникает задача разработки такого механизма электронной защиты, который бы смог заменить подпись и печать на бумажных документах. Необходимо разработать механизм цифровой подписи (digital signature), которая представляет собой дополнительную информацию, приписываемую к защищаемым данным. Цифровая подпись зависит от содержания подписываемого документа и некоего секретного элемента (ключа), которым обладает только лицо, участвующее в защищенном обмене [1]. Таким образом, механизм электронной цифровой подписи должен обеспечивать:

- аутентификацию – возможность однозначно идентифицировать отправителя. В системах криптографической защиты обеспечивается электронной цифровой подписью и сертификатом;
- целостность – информация должна быть защищена от несанкционированной модификации как при хранении, так и при передаче. В системах криптографической защиты обеспечивается электронной цифровой подписью и имитозащитой;
- неотрекаемость – отправитель не может отказаться от совершенного действия. В системах криптографической защиты обеспечивается электронной цифровой подписью и сертификатом.

Если обратиться к документации на различные системы, реализующие ЭЦП, то можно заметить, что производители, особенно в России, уделяют максимум внимания математическим аспектам реализованных алгоритмов. Десятки страниц посвящены тому, какова криптостойкость алгоритма и сколько лет потребуется злоумышленнику на подделку подписанного документа. Но, как ни парадоксально это прозвучит, на практике пользователей очень мало волнуют эти вопросы. Если система реализует некоторый стандарт, то для конечного пользователя этого факта достаточно. Тем более что проверить правильность приводимых в документации выкладок сможет только квалифицированный математик-криптограф. В первую очередь пользователи интересуются потребительскими свойствами предлагаемых систем, возможностям их встраивания в уже существующую технологию обработки информации. Перечислим основные факторы, оказывающими влияние на выбор криптографического средства.

Скорость – это один из основных параметров, особенно в системах связи, в которых осуществляется очень интенсивный обмен данными и передаваемая информация должна защищаться от подделки. Данный параметр слагается из двух составляющих – скорости генерации подписи и скорости ее проверки – и существенно зависит от скорости выработки хэш-функции, а также типа ЭВМ, на которой осуществляется генерация или проверка ЭЦП.

Поскольку при приобретении системы цифровой подписи, как правило, у заказчика уже сложилась информационная инфраструктура, то очень часто на первое место выходит вопрос об интеграции приобретаемой системы в принятую технологию обработки информации. Если система ЭЦП не поддерживает используемое у заказчика программное обеспечение (например, потому что оно разработано самим заказчиком), то поставщик должен поставлять интерфейс (API) для встраивания возможностей работы с цифровой подписью в систему заказчика. Такую возможность предлагают многие российские производители (например, Крипто-Про, МО ПНИЭИ, ЛАН КРИПТО, НИП «Информзащита»). Причем желательно, чтобы данный интерфейс существовал для различных операционных систем и платформ (Windows NT, Linux, Solaris, FreeBSD).

Необходим сертификат, удостоверяющий соответствие требованиям российских стандартов криптографии. При внедрении сертифицированного средства ЭЦП в систему электронного документооборота корпорации последняя получает возможности принципиально нового уровня: обмен электронными документами приобретает юридическую значимость.

В наиболее полной мере соответствует перечисленным выше факторам продукт российской компании ООО Крипто-Про – криптопровайдер «КриптоПро JCP». «Крипто-Про JCP» является средством криптографической защиты информации, реализующим российские криптографические стандарты и разработанным в соответствии со спецификацией JCA (Java Cryptography Architecture). Интеграция КриптоПро JCP с архитектурой Java позволяет использовать стандартные процедуры, такие как создание и проверка ЭЦП, шифрование, генерацию ключей, вычисление кодов аутентификации в Java Cryptography Extension (JCE) в соответствии со спецификациями Java Cryptography Extension (JCE) на различных операционных системах и аппаратных платформах. Криптопровайдер «КриптоПро JCP», разработанный в соответствии с требованиями фирмы Sun, является одним из немногих криптопровадеров, позволяющих реализовывать российские криптографические алгоритмы в среде Java [2].

Криптопровайдер «КриптоПро JCP» появился на рынке сравнительно недавно, и коммерческих продуктов, реализующих его функционал, на данный момент не существует.

Расширение криптографической архитектуры Java средствами криптопровайдера КриптоПро JCP

Платформа Java предоставляет все необходимые средства для разработки современных надежных приложений. Изначально разработчики платформы Java руководствовались следующим принципом: «Написано однажды, работает везде» («Write Once, Run Anywhere»), т.е. разработчик приложения на Java не должен задумываться над тем, на какой платформе будет выполняться написанный им код. Это привело к тому, что платформа Java стала одним из самых популярных средств разработки распределенных приложений. Естественно, что данная область программного обеспечения предъявляет повышенные требования к безопасности приложений.

Криптография – неотъемлемая часть современных средств информационной безопасности. Без криптографии невозможны надежная аутентификация, контроль целостности и сохранение конфиденциальности.

Криптографическая архитектура Java (Java Cryptography Architecture – JCA) включает в себя как часть набора инструментальных средств разработки из пакета java.security, которая относится к криптографии, так и ряд соглашений и спецификаций, принятых в JCA. Расширения криптографии Java (Java Cryptography Extension – JCE) обеспечивают структуру и выполнение функций шифрования, генерации ключей, выработку имитовставки, осуществляет поддержку симметричных и асимметричных алгоритмов шифрования.

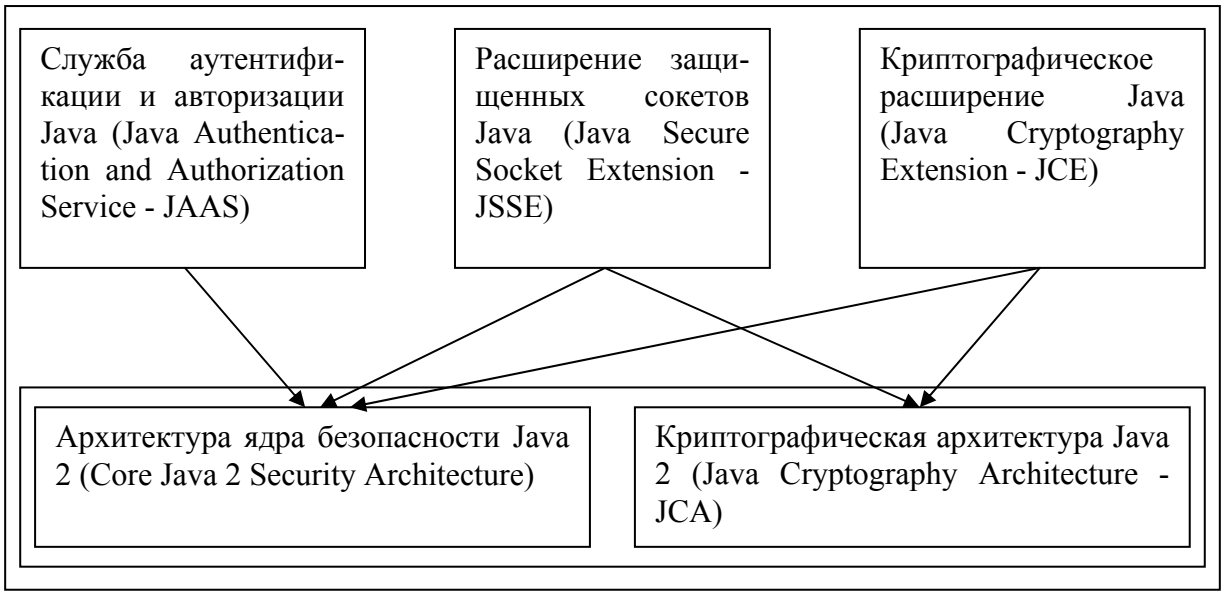


Рис. 1. Стандартные компоненты системы безопасности Java 2

Ранее расширения криптографии Java реализовывались отдельным пакетом (дополнением) для средств разработчика Java. Это происходило из-за того, что технологии симметричного шифрования и выработки ключевого материала подпадали под экспортные ограничения США. Однако, начиная с версии 1.4, криптографические расширения интегрированы в набор инструментальных средств разработки.

Программный интерфейс JCE позволяет реализовывать:

- блочное симметричное шифрование (DES, RC2, IDEA);
- потоковое симметричное шифрование (RC4);
- асимметричное шифрование (RSA);
- шифрование, основанное на паролях;
- ключевые спецификации;
- выработку имитовставки.

JCE в пределах JDK включает два компонента программного обеспечения:

- структуру, которая определяет и поддерживает предлагаемые криптопровайдерами услуги шифрования;
- собственно криптопровайдер «SunJCE».

Криптографическая архитектура Java 2 (рис. 1) разработана для предоставления следующих сервисов:

- постановка/проверка электронной подписи;
- вычисление хэш-функции;
- генерация пар ключей открытый/секретный;
- создание сертификатов, подтверждающих аутентичность открытых ключей;
- хранение ключей, а также сертификатов надежных партнеров;
- преобразование ключей из представления со скрытой структурой в общепонятное представление и наоборот;

- управление параметрами криптографических алгоритмов;
- генерация параметров криптографических алгоритмов;
- генерация (псевдо)случайных чисел;
- симметричное шифрование;
- выработка общего ключевого материала,

и удовлетворяет следующим технологическим требованиям:

- обеспечивать независимость от алгоритмов и их реализаций;
- обеспечивать взаимную совместимость реализаций;
- обеспечивать расширяемость набора алгоритмов и их реализаций.

Платформенная независимость достигается путем использования архитектуры криптопровадеров. Криптопровайдер (Cryptographic Service Provide – криптографический сервис-провайдер) представляет собой класс, реализующий одну или несколько криптографических функций, которые подчиняются интерфейсам, определенным в JCA. В стандартный пакет поставки включен криптопровайдер фирмы Sun, реализующий следующие алгоритмы и механизмы:

- алгоритм DSA, описанный в стандарте США NIST FIPS 186;
- алгоритмы MD5 (RFC 1321) и SHA-1 (NIST FIPS 180-1);
- механизм генерации ключевой пары для алгоритма DSA;
- механизм генерации и управления параметрами алгоритма DSA;
- другие.

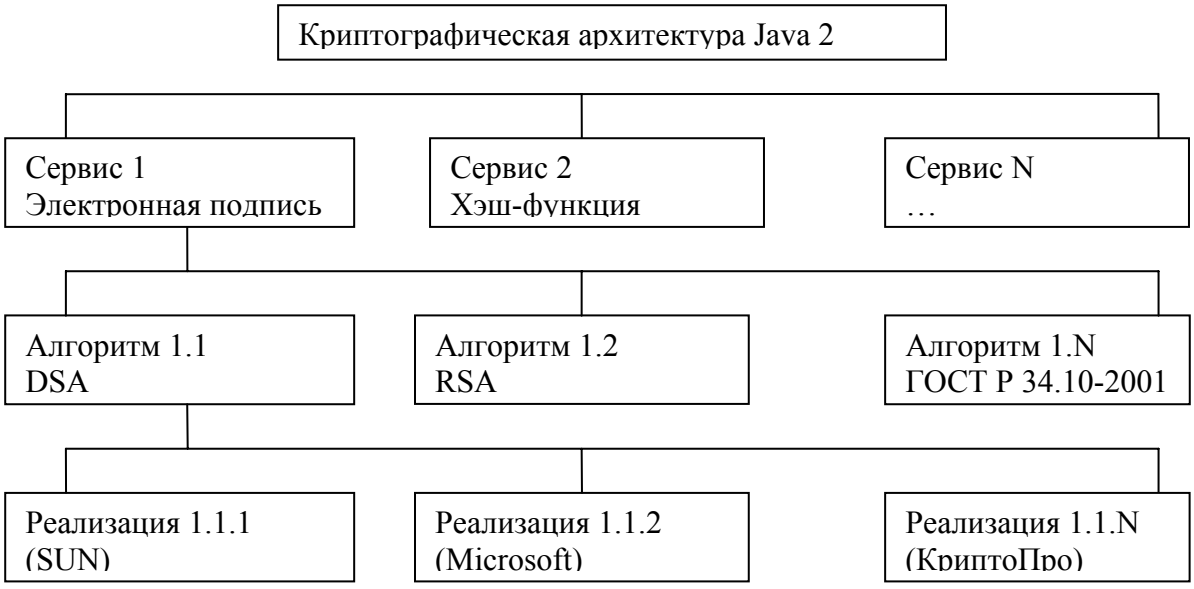


Рис. 2. Соотношение между криптографическими сервисами, алгоритмами и реализациями

Алгоритмическая независимость достигается путем определения криптографических механизмов (служб) и программных классов, определяющих функционал этих механизмов. Вот основные классы криптографических механизмов JCA:

- MessageDigest – работа с хэшем;
- Signature – работа с цифровой подписью;
- KeyPairGenerator – генерация ключевой пары;
- KeyFactory и KeyStore – работа с ключами и их хранение;
- CertificateFactory – создание сертификатов открытых ключей;
- AlgorithmParameters – задание параметров криптографических алгоритмов;
- AlgorithmParametersGenerator – создание параметров для алгоритмов;
- SecureRandom – генерация случайных чисел;

- CertPathBuiler – построение цепочек сертификации;
- CertPathValidator – верифицирование цепочек сертификации;
- CertStore – использование хранилища сертификатов и списков отозванных сертификатов.

Соотношение между криптографическими сервисами, алгоритмами и реализациями представлено на рис. 2. Каждый сервис может обеспечиваться несколькими алгоритмами, каждый из которых, в свою очередь, может иметь несколько реализаций [3]. Например, для вычисления хэш-функции предназначены алгоритмы MD5/SHA-1 (равно как и российский ГОСТ «Функция хэширования»), для выработки и проверки электронной подписи – алгоритмы RSA/DSA, российский ГОСТ «Процессы формирования и проверки электронной цифровой подписи» в реализации компании КриптоПро.

Описание работы программы

Программное средство реализует следующие криптографические функции, основанные на сервисах, предоставляемых криптопровайдером КриптоПро JCP:

1. генерация ключевой пары (открытый и секретный ключи);
2. генерация самоподписанного сертификата открытого ключа;
3. запись ключевой пары и сертификата в хранилище;
4. экспорт сертификата открытого ключа в файл;
5. формирование электронной цифровой подписи;
6. проверка электронной цифровой подписи.

Работа по созданию ключевой пары, сертификата, формированию и проверке электронной цифровой подписи осуществляется с помощью мастеров – наборов экранных форм, позволяющих сделать выполнение перечисленных функций простым и интуитивно понятным.

Алгоритм заботы программы представлен на рис. 3.

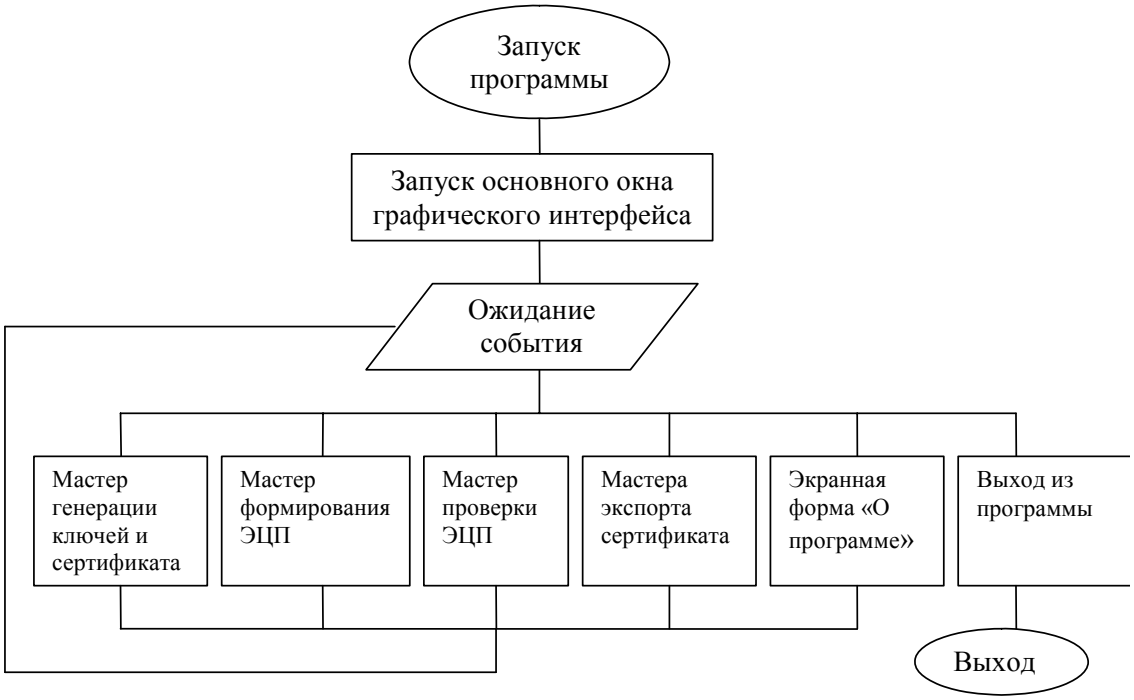


Рис. 3. Алгоритм работы программы

Заключение

Рассмотренный программный продукт позволяет генерировать ключевую пару и сертификат открытого ключа пользователя, формировать цифровую подпись для файлов любых форматов и проводить проверку подлинности цифровой подписи с помощью нового российского криптопровайдера КриптоПро JCP.

Данное программное средство, как и исследуемый криптопровайдер, реализовано на языке Java, что имеет ряд положительных сторон: язык Java платформенно независим, компилятор для этого языка свободно распространим.

Результатом написания программы стало обоснованное подтверждения возможности использования криптопровайдера КриптоПро JCP для создания программных средств электронной цифровой подписи.

Дальнейшим развитием рассмотренного программного продукта может быть добавление функциональных возможностей по интеграции с существующими на российском рынке программными продуктами, реализующими функции удостоверяющих центров, а также добавление возможности работы со службами актуальных статусов сертификатов и штампов времени. Данное расширение функциональных возможностей необходимо для создания систем защищенного, юридически значимого электронного документооборота.

Литература

1. Как обеспечить подлинность электронных документов [Электронный ресурс] / ред. Лукацкий А.В., Режим доступа: http://docs.luksian.com/security/crypto/digisign/, свободный. Загл. с экрана.
2. КриптоПро JCP [Электронный ресурс] Режим доступа: http://www.cryptopro.ru/cryptopro/products/jcp/default.htm, свободный. Загл. с экрана.
3. Java Cryptography Architecture [Электронный ресурс] Режим доступа: http://java.sun.com/j2se/1.5.0/docs/guide/security/CryptoSpec.html, свободный. Загл. с экрана.

ОСНОВНЫЕ ЗАДАЧИ ПРИКЛАДНОЙ ТЕОРИИ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ АСУ

П.И. Тутубалин (Казанский государственный технический университет им. А.Н. Туполева)
Научный руководитель – д.т.н., профессор В.С. Моисеев (Казанский государственный технический университет им. А.Н. Туполева)

Проблема информационной безопасности является одним из важнейших аспектов развития современного общества. Следует отметить, что работа в этих направлениях ведется на эмпирической базе в связи с отсутствием общепринятой теории информационной безопасности современных АСУ. В данной статье приведены подходы для создания подобной теории.

Введение

Существующие работы в области теории безопасности информационных систем применяют весьма абстрактный математический аппарат, использование которого при решении реальных задач анализа и синтеза средств защиты разрабатываемой, эксплуатируемой системы практически невозможно. Обзор литературы показал, что в настоящее время отсутствуют вероятностные модели, описывающие средства защиты информации (СЗИ). В этой связи весьма актуальным является создание прикладной теории безопасности информационных систем (ИС), основной целью которой является разработка научно обоснованных методик разработки и эксплуатации оптимальных средств защиты конкретной системы. Сформулируем основные принципы прикладной теории безопасности ИС.

1) Принцип комплексности применяемых средств защиты. 2) Принцип экономичности СЗИ. 3) Принцип максимальной защиты критических компонентов КИС. 4) Принцип прогнозирования угроз и применения средств нападения. 5) Принцип обеспечения максимальной неопределенности применяемых стратегий защиты для противника.

Рассмотрим модели, которые могут быть использованы при построении прикладной теории безопасности ИС.

Модели информационной системы и подходы к обеспечению ее информационной безопасности

На наш взгляд, в основу математического моделирования информационных процессов и систем можно положить понятие абстрактной математической модели вида:

$$m = \{M, R_1, R_1, ..., R_n\}. \quad (1)$$

Здесь M – множество объектов и процессов, отражаемых в рассматриваемой модели; $R_1 R_2, ..., R_n$ – совокупность отношений, связывающих между собой элементы множества M. Построение базовых моделей теории безопасности естественно начать с моделирования объектов защиты – информационных систем и реализованных в их составе информационных технологий.

Концептуальная модель информационной системы

Построим концептуальную модель современной ИС, используя в качестве основы модель вида (1). В составе множества элементов модели выделим следующие подмножества: Λ – множество подразделений организации, охваченных рассматриваемой версией ИС; Π – множество пользователей ИС; A – множество аппаратных средств сис-

темы; Р – множество системных и прикладных программ ИС; В – множество локальных банков данных.

Как известно, любая система представляет собой совокупность элементов и связей между ними. Эти связи будем описывать следующими отношениями: 1) закрепление пользователей за конкретными АРМ-ми $R_1 \subseteq \Pi \times \mathrm{A}$; 2) распределение ПО системы по ее аппаратным средствам $R_2 \subseteq A \times P$; 3) связь программ и данных в системе $R_3 \subseteq P \times B$; 4) размещение аппаратных средств системы по подразделениям организации $R_4 \subseteq \Lambda \times \mathrm{A}$. Тогда модель ИС можно представить в виде:

$$IS = \{\Lambda, \Pi, \mathrm{A}, \mathrm{P}, \mathrm{B}, \mathrm{R}_1, \mathrm{R}_2, \mathrm{R}_3, \mathrm{R}_4\}.$$

От рассмотрения моделей ИС можно перейти к модели прикладной информационной технологии (ИТ). Прикладные ИТ в качестве выходного результата всегда имеют некоторое управленческое, проектное и другое решение. Субъектами прикладной ИТ являются лица, готовящие решения (ЛГР), а также лицо (лица), принимающие решения (ЛПР).

Будем считать, что объектом любой прикладной ИТ, т.е. средством ее реализации в составе любой системы, является вполне определенный ИТ-продукт. Введем в рассмотрение следующие множества: $\Pi_1 \subset \Pi$ – множество ЛГР; $\Pi_2 \subset \Pi$ – множество ЛПР; $\Re$ – множество решений, принимаемых ЛПР в рассматриваемой ИС. Отметим, что при этом должно иметь место условие: $\Pi_1 \cup \Pi_2 = \Pi$, где Π – множество пользователей ИС. Совокупность используемых в ИС прикладных ИТ будем описывать отношением $IT \subseteq \Pi_1 \times A \times B \times P \times \Pi_2 \times \Re$. Элементами этого отношения являются кортежи $(\pi_{1i}, a_j, b_k, p_r, \pi_{2s}, \rho_l)$, где $\pi_{1i} \in \Pi_1$, $a_j \in A$, $b_k \in B$, $p_r \in P$, $\pi_{2s} \in \Pi_2$, $\rho_l \in \Re$. Смысл кортежа состоит в том, что *i*-е ЛПР, используя *j*-е аппаратное средство, *k*-ю базу данных и *r*-е программное средство, готовит варианты решений *s*-у ЛПР, который принимает *l*-е решение.

Рассмотрим концептуальную модель системы защиты ИТ-продуктов. Будем считать, что используемые в составе рассматриваемой ИС объекты защиты представляют собой ИТ-продукты, включающие в себя множество аппаратных (А), программных (Р) средств и файлов (Ф), используемых при реализации соответствующих функциональных задач ИС и множество линий (каналов) связи *L*.

Введем в рассмотрение множество объектов защиты *Z* в рассматриваемой ИС, которое определим как $Z = A \cup P \cup \Phi \cup L$. Обозначим через *U* множество угроз ИС. Для исключения этих и других потенциальных угроз в ИС используется множество *M* средств защиты. Зададим первичные отношения модели видов «угроза-объект» и «объект-средство защиты» как $V_1 \subseteq U \times Z$, $V_2 \subseteq Z \times M$. Тогда концептуальную модель системы защиты можно представить в виде

$$M_{C3} = \{Z, V_1, V_2\}.$$

Вероятностные характеристики информационной безопасности АСУ

Для решения задачи анализа и синтеза СЗИ с применением широко развитых в настоящее время вероятностных методов необходим метод определения допустимого, с точки зрения заказчика, значения вероятности обеспечения информационной безопасности (ИБ) создаваемой АСУ.

Пусть q – вероятность обеспечения ИБ при функционировании АСУ. Обозначим через Z_ϕ затраты от несанкционированного вмешательства в систему, которые зависят от имеющегося в АСУ уровня ИБ. Будем считать, что эта величина принимает максимальное значение при $q = 0$ и значение, равное нулю, при $q = 1$. Таким образом, имеем

функцию вида $Z_ф = Z_ф(q), q \in [0,1]$. Пусть Z_c – затраты на СЗИ, которые описываются функцией вида $Z_c = Z_c(q), q \in [0,1]$. Эта функция при $q = 0$ равна нулю, а при $q = 1$ принимает максимальное значение.

Сформулируем двухкритериальную задачу оптимизации [1]:

$$(Z_ф, Z_c) \to \min_{0<q<1}.$$

Парето-оптимальное решение этой задачи будем строить путем минимизации линейной свертки критериев:

$$L(q,\lambda) = \lambda Z_ф(q) + (1-\lambda) Z_c(q) \to \min_{0<q<1}.$$

В работе [2] был предложен подход к формированию допустимого значения вероятности $P_{ИБ}^{доп}$ нарушения ИБ разрабатываемой АСУ. Предполагая, что хотя бы одно нарушение таких основных свойств ИБ, как конфиденциальность, целостность или доступность данных, ведет к потере безопасности АСУ, имеем, что $P_{ИБ} = 1-(1-P_{конф})(1-P_{цел})(1-P_{дост})$, где $P_{конф}, P_{цел}, P_{дост}$ – соответственно, вероятности нарушения конфиденциальности, целостность и доступности информации в АСУ. Переходя к вероятностям противоположных случайных событий, эти условия можно переписать:

$$Q_{конф} Q_{цел} Q_{дост} = Q_{ИБ}^{доп}.$$

Заказчик АСУ может определить среднюю интенсивность атак [3, 4] на компоненты конфиденциальности, целостности и доступности АСУ. Пусть количество попыток нарушения конфиденциальности информации в АСУ по оценкам заказчика равно $\lambda_{конф}$, а для целостности и доступности информации, соответственно, $\lambda_{цел}$ и $\lambda_{дост}$. Тогда можно определить среднее время между попытками нарушения конфиденциальности, целостности и доступности информации в АСУ:

$$\tau_{конф} = \frac{1}{\lambda_{конф}},\ \tau_{цел} = \frac{1}{\lambda_{цел}},\ \tau_{дост} = \frac{1}{\lambda_{дост}}.$$

Определим оценки важности рассматриваемых аспектов обеспечения ИБ, с учетом проведенных выше рассуждений, в следующем виде:

$$\alpha = \frac{\tau_{конф}}{\tau_{конф} + \tau_{цел} + \tau_{дост}}, \beta = \frac{\tau_{цел}}{\tau_{конф} + \tau_{цел} + \tau_{дост}}, \gamma = \frac{\tau_{дост}}{\tau_{конф} + \tau_{цел} + \tau_{дост}}.$$

Тогда вероятности $Q_{конф}^{доп}$, $Q_{цел}^{доп}$ и $Q_{дост}^{доп}$ можно определить по формулам

$$Q_{конф}^{доп} = \left(Q_{ИБ}^{доп}\right)^{\alpha}, Q_{цел}^{доп} = \left(Q_{ИБ}^{доп}\right)^{\beta}, Q_{дост}^{доп} = \left(Q_{ИБ}^{доп}\right)^{\gamma}$$

или в общем виде:

$$Q_m^{доп} = \left(Q_{ИБ}^{доп}\right)^{\alpha_m},\ \alpha_m = \frac{\tau_m}{\sum_{m=1}^{M} \tau_m}.$$

Рассмотрим методику формирования допустимых значений вероятностей обеспечения конфиденциальности процессов обработки информации (ОИ) для компонент АСУ, обеспечивающих требуемый уровень конфиденциальности $Q_{конф}^{доп}$.

Для описания процесса ОИ введем в рассмотрение следующие множества [5]: D – множество данных, циркулирующих в системе, Z – множество задач (прикладных программ, процедур, приложений и т.д.), реализующих функции рассматриваемой АСУ; T – множество ТС рассматриваемой системы.

В множестве D выделим подмножества входных D^B и выходных D^V данных. Структуру формирования в рассматриваемой АСУ выходных данных D^V на основе решения множества задач Z, использующих определенные входные данные D^B, будем в общем виде представлять с использованием аппарата n-арных отношений [6]:

$R = D^B \times Z \times Z \times D^V$.

Пусть в рассматриваемой АСУ определенно заказчиком упорядоченное по возрастанию множество K уровней конфиденциальности информации. Сделаем следующие предположения: 1) в множестве задач Z, решаемых в АСУ, не используются «закрытые» алгоритмы, 2) уровни конфиденциальности задач и формируемых при их решении выходных данных зависят только от уровней конфиденциальности используемых входных данных, 3) если некоторый элемент АСУ имеет несколько уровней конфиденциальности, то ему должен быть присвоен максимальный из имеющихся у него уровень.

Разобьем множество данных D^B на подмножества D_k^B, $k \in K$. Множество конфиденциальных входных данных $D_{конф}^B = D^B \setminus D_o^B$. Для каждого множества данных $D_k^B \subseteq D_{конф}^B$ введем в рассмотрение вероятности Q_k^B обеспечения заданного уровня их конфиденциальности $k \in K$, $k \neq 0$. Пусть $\tau_k^{конф}$ – среднее время между попытками нарушения ИБ данных k-го уровня конфиденциальности, тогда вероятности Q_k^B равны

$$Q_k^B = \left(Q_{конф}^{доп}\right)^{\alpha_k^{конф}}, \ \alpha_k^{конф} = \frac{\tau_k^{конф}}{\sum_{k=1}^{K} \tau_k^{конф}}.$$

Рассмотрим частные бинарные отношения вида $R_1 = D^B \times Z, R_2 = Z \times D^V$, описывающие используемые при решении каждой задачи $z_j \in Z$ входные данные $d_i^B \in D^B$, а также формируемые при этом выходные данные $d_r^V \in D^V$. Мощности рассмотренных выше множеств, соответственно, равны $\left|D^B\right| = n$, $|Z| = m$, $\left|D^V\right| = l$, $|T| = t$. Следуя работе [6], отношения R_1, R_2 могут быть описаны булевскими матрицами $B_1 = [b_{ij}^{(1)}]_{n \times m}$ и $B_2 = [b_{jr}^{(2)}]_{m \times l}$. Используя матрицу B_1, определим для каждой задачи $z_j \in Z$ подмножество $D_j^B \in D^B$ используемых ею входных данных. Получим решающее правило для определения вероятностей обеспечения требуемого уровня конфиденциальности Q_j^{ZB} задачи $z_j \in Z$ по ее входным данным:

$$Q_j^{ZB} = \arg \max_{k \in K} \left\{ Q_k^B \middle| D_{jk}^B \neq \varnothing \right\}, \quad j = \overline{(1,m)}.$$

Аналогичным образом назначаются уровни конфиденциальности выходным данным $d_r^V \in D^V$.Рассмотрим частное бинарное отношение $R_3 \subseteq Z \times Z$, которое опишем с помощью матрицы смежности $B_3 = [b_{hj}^{(3)}]_{m \times m}$. Для каждой задачи $z_j \in Z$ выделим подмножество смежных ей задач $Z_j = \left\{ z_h \in Z \middle| b_{hj}^{(3)} = 1, h = \overline{(1,m)} \right\}, j = \overline{(1,m)}$. Для задачи $z_j \in Z$ определим вероятность обеспечения требуемого уровня конфиденциальности:

$$Q_j^z = \arg \max_{z_h \in Z_j} \{ \max_{k \in K} \left(Q_k^{ZB}, Q\left(d_r \in D_h^V\right) \right) \middle| Z_j \neq \varnothing, D_h^V \neq \varnothing \}, \quad j = \overline{(1,m)}, \quad h = \overline{(1,m)}.$$

Рассмотрим множества задач Z и ТС T. Построим частное бинарное отношения вида $R_4 = Z \times T$. Тогда, следуя работе [6], это отношение может быть описано булев-

ской матрицей $B_4 = [b_{jf}^{(4)}]_{m\times t}$. Обозначим через Q_{jf} вероятности обеспечения требуемого уровня конфиденциальности ТС $t_f \in T$ при его использовании для решения задачи $z_j \in Z$. По аналогии с рассуждениями, представленными выше, получим решающие правила для определения значений вероятностей Q_{jf}:

$$Q_{jf} = b_{jf}^{(4)}\left(Q_j^z\right)^{\alpha_{jf}}, \quad j = \overline{(1,m)}, \ \alpha_{jf} = \frac{b_{jf}^{(4)}\tau_{jf}}{\sum_{f=1}^{t} b_{jf}^{(4)}\tau_{jf}}.$$

Окончательное значение вероятности Q_{jf}^o обеспечения требуемого уровня конфиденциальности каждого применяемого в АСУ ТС вычисляется как

$$Q_{jf}^o = \max_{z_j \in Z} Q_{jf}, \quad f = \overline{(1,t)}, j = \overline{(1,m)}.$$

Рассмотрим требования, предъявляемые к целостности информации. Определим среднее интервалы времени между попытками нарушения ИБ данных и задач $\tau_D = \frac{1}{\lambda_D}, \tau_Z = \frac{1}{\lambda_Z}$. Найдем вероятности обеспечения ИБ данных D и задач Z:

$$Q_D = \left(Q_{цел}^{доп}\right)^{\alpha}, \quad Q_Z = \left(Q_{цел}^{доп}\right)^{\beta}, \ \alpha = \frac{\tau_D}{\tau_D + \tau_Z}, \beta = \frac{\tau_Z}{\tau_D + \tau_Z}.$$

Определим вероятность обеспечения целостности входных данных k-го уровня конфиденциальности:

$$Q_k^{ЦВ} = (Q_D)^{\alpha_k^{цел}}, \ \alpha_k^{цел} = \frac{\tau_k^{цел}}{\sum_{k=0}^{K} \tau_k^{цел}}.$$

По аналогии с рассуждениями, приведенными для компоненты конфиденциальности, получаем следующее решающее правило для формирования вероятностей обеспечения требуемого уровня целостности $Q_j^{ЦZB}$ задачи $z_j \in Z$ с использованием ее входных данных: $Q_j^{ЦZB} = \arg\max_{w\in W}\left\{Q_w^{ЦВ} \middle| D_{jw}^B \neq \varnothing\right\}, \quad j = \overline{(1,m)}$. Аналогичным образом назначаются уровни целостности выходным данным $d_r^V \in D^V$:

$$Q_j^{Цz} = \arg\max_{z_h \in Z_j}\{\max_{w\in W}\left(Q_w^{ЦZB}, Q\left(d_r \in D_h^V\right)\right) \middle| Z_j \neq \varnothing, D_h^V \neq \varnothing\}, j = \overline{(1,m)}, \quad h = \overline{(1,m)}.$$

Средние интервалы между попытками нарушения целостности задач $\tau_{z_j} = \frac{1}{\lambda_{z_j}}$.

Построим с помощью матрицы смежности задач $B_3 = [b_{hj}^{(3)}]_{m\times m}$, задающей частное бинарное отношение $R_3 \subseteq Z \times Z$, множество путей L. Рассмотрим некоторый путь $l_r\left(z_{p1}, z_{p2}, ..., z_{pi}, ..., z_{pr}\right) \in L$, где $r \in \overline{(1,R)}$, $R = |L|$, $z_{pi} \in Z$, $i = \overline{(1,r)}$, а $pr = |l_r|$, тогда $Q_{z_{pi}} = (Q_Z)^{\alpha_{z_{pi}}}$, где $\alpha_{z_{pi}} = \frac{\tau_{z_{pi}}}{\sum_{p1}^{pr} \tau_{z_{pi}}}$. Вероятности обеспечения требуемого уровня целостности каждой из задач, решаемой в АСУ, примут вид:

$$Q_{z_{pi}}^o = \max_{l\in L} Q_{z_{pi}}, \quad pi = \overline{(1,m)}, l = \overline{(1,L)}.$$

Назначим задаче $z_{pi} \in Z$ уровень целостности, равный $Q_{pi}^{ЦZ} = \max\left(Q_{pi}^{Цz}, Q_{z_{pi}}^{o}\right)$. Обозначим через $Q_{jf}^{цел}$ вероятности обеспечения требуемого уровня целостности ТС $t_f \in T$ при его использовании для решения задачи $z_j \in Z$. Вероятности обеспечения целостности ТС по аналогии с рассуждениями, приведенными выше, примут вид:

$$Q_{jf}^{цел} = b_{jf}^{(4)}\left(Q_j^{ЦZ}\right)^{\alpha_{jf}^{цел}},\ \alpha_{jf}^{цел} = \frac{b_{jf}^{(4)}\tau_{jf}^{цел}}{\sum_{f=1}^{t} b_{jf}^{(4)}\tau_{jf}^{цел}}.$$

С учетом принципа максимальной защищенности $Q_f^{цел} = \max\{Q_{f_1}^{цел}, Q_{f_2}^{цел}, ..., Q_{f_g}^{цел}\}$, где g – количество альтернативных уровней ИБ ТС $t_f \in T$.

Формирование допустимых значений вероятностей обеспечения доступности данных, технических средств и программного обеспечения АСУ аналогично методике, предложенной для компоненты целостности.

Модели и методы обеспечения информационной безопасности

Рассмотрим вероятностные модели и методы обеспечения информационной безопасности АСУ. Ниже приведена математическая модель выделения критических элементов АСУ. Обозначим $J(X, Д)$ – граф прикладных ИТ, где $X = П_1 \cup A \cup B \cup P \cup П_2 \cup \Re$ – множество вершин, $Д$ – множество дуг графа. Будем считать, что нарушение нормальной работы любой вершины $x \in X$ этого графа приводит к невозможности реализации соответствующей прикладной ИТ из их наличного множества. Обозначим через L полное множество маршрутов в графе $J(X, Д)$ и построим матрицу $\Omega = [\omega_{lk}]$ с элементами:

$$\omega_{lk} = \begin{cases} 1, & \textit{если вершина } x_k \in X \textit{ входит в маршрут } l_k \in L; \\ 0, & \textit{в противном случае}. \end{cases}$$

Выделим в графе J минимальное число вершин, через которые проходят все пути множества L. Введем вектор булевских переменных $\xi = (\xi_1, \xi_2, ... \xi_n)$, где $n = |x|$ – число вершин графа J. Любая переменная ξ_k может принимать следующие значения:

$$\xi_k = \begin{cases} 1, & \textit{если вершина } x_k \in X \textit{ включена в минимальное покрытие}; \\ 0, & \textit{в противном случае}. \end{cases}$$

Это требование можно записать в виде следующих ограничений:

$$\xi_k \in \{0;1\}, k = \overline{(1,n)}. \qquad (2)$$

Условия того, чтобы через каждую вершину, входящую в минимальное покрытие, проходил не менее чем один маршрут множества L, записывается как

$$\sum_{k=1}^{n} \omega_{lk}\xi_k \geq 1, \quad l \in L. \qquad (3)$$

С учетом ограничений (2) требование минимальности числа вершин, входящих в искомое покрытие, представляется целевой функцией вида

$$N = \sum_{k=1}^{n} \xi_k \to \min_{\xi_k}. \qquad (4)$$

Рассмотрим дополнительную целевую функцию вида:

$$P = \prod_{k=1}^{n} p_k^{\xi_k} \to \max_{\xi_k}. \qquad (5)$$

Здесь p_k – вероятность успеха возможных атак на k–ю компоненту множества прикладных ИТ. Решение задачи (5), (4), (3), (2) выделяет критические элементы АСУ.

Рассмотрим подход, позволяющий реализовать оптимальный выбор СЗИ АСУ. Зададим отношение W_1 представленное элементами:

$$w_{ij}^{(1)} = \begin{cases} 1, & \textit{если от } i-\textit{ой угрозы ИТ}-\textit{продукты системы защищает} \\ & j-\textit{ее средство}; \\ 0, & \textit{в противном случае.} \end{cases}$$

Пронумеруем элементы множества угроз и располагаемых средств защиты как $U = \{1,2,3,...,i,...,n\}, M = \{1,2,3,..., j,...,m\}$. Введем в рассмотрение булевские переменные:

$$x_j \in \{0;1\}, \quad j = \overline{(1,m)}, \qquad (6)$$

такие, что

$$x_j = \begin{cases} 1, & \textit{если для защиты системы выбрано } j-\textit{ое средство}; \\ 0, & \textit{в противном случае.} \end{cases}$$

Потребуем, чтобы каждая угроза системы была бы парирована не менее чем одним средством защиты. Это требование представим условием

$$\sum_{j=1}^{m} w_{ij}^{(1)} x_j \geq 1, \quad i = \overline{(1,n)}. \qquad (7)$$

Стоимость использования в ИС средств защиты определяется выражением

$$C = \sum_{j=1}^{m} c_j x_j \to \min_{x_j}, \qquad (8)$$

где c_j – стоимость j-го СЗИ. Пусть p_j – вероятность того, что противник сможет преодолеть j-е СЗИ. В этом случае стойкость системы защиты может быть оценена как вероятность ее преодоления противником. Эту вероятность можно представить как

$$\Theta = \prod_{j=1}^{m} p_j^{x_j} \to \min_{x_j}. \qquad (9)$$

Таким образом, выбор оптимальных СЗИ можно осуществить путем решения двухкритериальной задачи нелинейного булевского программирования (9), (8), (7), (6).

Рассмотрим теоретико-игровую модель размещения конфиденциальной информации на серверах АСУ. Пусть игрок A – это администратора СЗИ АСУ. В его распоряжении находится n стратегий $A_1, A_2, ..., A_n$, где A_i – стратегия игрока A, состоящая в том, что КД нужно расположить на i-м сервере. В распоряжении нарушителя также находится n стратегий $B_1, B_2, ..., B_n$, где B_j – стратегия игрока B, состоящая в том, что КД нужно искать на j-м сервере. Построим платежную матрицу Γ для игрока A:

$$\Gamma = \gamma_{ij} = \begin{bmatrix} P + c_1 & c_1 & \ldots & c_1 \\ c_2 & P + c_2 & \ldots & c_2 \\ \ldots & \ldots & \ldots & \ldots \\ c_n & c_n & \ldots & P + c_n \end{bmatrix},$$

где γ_{ij} – потери игрока A, если атаке подвергается j-й сервер, а КД находятся на i-м сервере. Здесь $P > 0$ – это материальный ущерб, а c_i – стоимость хранения КД на i-м

сервере. Гарантирующую смешанную стратегию игрока A примем за решение следующей задачи линейного программирования:

$$v \to \min, \tag{10}$$

где v – проигрыш игрока A, при выполнении условий

$$\sum_{i=1}^{n} p_i \gamma_{ij} - v \le 0, j = \overline{1,n}, \ \sum_{i=1}^{n} p_i = 1, \ p_i \ge 0, i = \overline{1,n}. \tag{11}$$

Будем использовать среднюю стоимость C размещения КД на серверах системы:

$$C = \sum_{i=1}^{n} c_i p_i \to \min. \tag{12}$$

Построим множество оптимальных по Парето решений с использованием линейной свертки критериев (10),(12) и при выполнении условий (11):

$$L(\alpha, p_1,.., p_i,.., p_n, v) = \alpha v + (1-\alpha)\sum_{i=1}^{n} c_i p_i \to \min.$$

Получим множество решений $\{(p_1, p_2,..., p_n)\}$, оптимальных по Парето.

Далее приведен метод, позволяющий использовать для повышения уровня безопасности АСУ мажоритарное резервирование критических компонент ПО. Будем считать, что имеются три программных продукта (ПП) $П_1, П_2, П_3$, выполняющих в составе АСУ одну и ту же функцию. Объединим введенные в рассмотрение ПП в мажоритарную программную систему (МПС), представленную на рисунке.

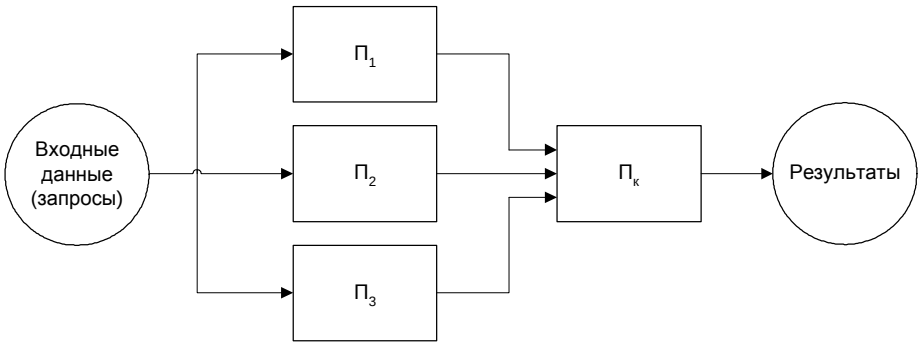


Рис. Мажоритарная программная система

Обозначим через p_i вероятность не нарушения безопасности *i*-ого ПП, p_k – вероятность ненарушения безопасного кворум-продукта. Вероятность получения функционально пригодных результатов с помощью МПС определим как

$$P = p_k(p_2 p_3 + p_1 p_3 + p_1 p_2 + 2 p_1 p_2 p_3). \tag{13}$$

Пусть $p_1 = p_2 = p_3 = p$, тогда выражение (13) принимает вид $P = p_k(3p^2 + 2p^3)$. Безопасность МПС будет выше стойкости любой ее компоненты из состава $П_1, П_2, П_3$.

Одним из перспективных способов повышения ИБ АСУ является маскировка конфиденциальных данных. Рассмотрим методы формирования дезинформирующих данных X_i, Y_j и Z_l, $i = \overline{1,n}$, $j = \overline{1,m}$, $l = \overline{1,k}$. Будем считать, что для каждой *i*-й записи ложного файла выполняется условие вида $X_i \in [a_i, b_i], i = \overline{1,n}$, где $[a_i, b_i]$ – интервал возможных значений реальной записи. Ложные данные $X_1, X_2,...X_n$ будем формировать в виде случайных чисел, каждое из которых должно удовлетворять условиям $X_i \in [a_i, b_i], i = \overline{1,n}$. Рассмотрим методы генерации дезинформирующих данных Y_j с дискретным множеством возможных значений, $j = \overline{1,m}$. Будем считать, что для каждой *j*-й записи ложного файла СПрО задано множество

$$Y_j \in \{k_1, k_2, ..., k_{r_j}\} \quad (14)$$

правдоподобных с точки зрения противника значений реального показателя y_j, $j = \overline{1,m}$. Ложные данные $Y_1, Y_2, ...Y_m$ будем формировать в виде совокупности случайных чисел, каждое из которых принимает значения из соответствующего множества (14). Ложные текстовые записи $Z_1, Z_2, ...Z_k$ будем формировать случайным образом путем выбора из состава соответствующих совокупностей «похожих» записей, сформированных СПрО.

Файл ложной конфиденциальной информации должен иметь максимальную степень дезинформации D и минимальным образом B быть близко к реальным данным. Эти требования можно записать как $D \to \max, B \to \min$. В работе [7] описан метод, с помощью которого можно выделить Парето-оптимальные варианты файлов ложных данных, удовлетворяющие критериям $D \to \max, B \to \min$.

Испытание на информационную безопасность средств защиты информации

В доступной литературе говорится о необходимости испытаний СЗИ, но не указываются пути реализации этого важного этапа в создании и эксплуатации СЗИ. Выделим две цели, которые должны быть достигнуты при решении задач АСИ СЗИ: 1) уменьшение трудоемкости проведения испытаний СЗИ АСУ на ИБ; 2) достоверная оценка защищенности СЗИ АСУ. Достижение этих целей, как и в любых других областях техники, возможно только путем автоматизации этих процессов, в частности применения специальных автоматизированных систем испытаний (АСИ).

Цель любого испытания – это получение достоверных результатов. Достижение этой цели возможно при использовании: 1) аппарата теории вероятностей и математической статистики, а именно, раздела обработки результатов испытаний; 2) количественных оценок, которые будут выступать в качестве критерия достоверности испытаний. В качестве такого количественного критерия может выступать оценка вероятности нарушения ИБ объекта испытания (ОИ).

Отметим, что количественная оценка достоверности проведения испытаний позволяет судить о достижении целей испытания при условии, что заданы или рассчитаны допустимые значения вероятностей нарушения ИБ ОИ. Использование подхода, учитывающего количественные оценки, позволяет конкретизировать и упростить методику принятия решения о соответствии ОИ требованиям, предъявляемым к нему с точки зрения ИБ.

В составе АСИ СЗИ можно выделить следующие основные компоненты: техническое и математическое обеспечения, программное обеспечение общего и специального типов, информационное обеспечение, лингвистическое обеспечение, персонал АСИ СЗИ.

Отметим, что комплекс программ организации атак можно организовать с использованием современных сканеров уязвимостей. Для сокращения трудоемкости разработки АСИ СЗИ предлагается использовать существующее программное обеспечение. В частности, для компоненты ОПО можно использовать любую UNIX-подобную ОС с набором всевозможных утилит и программ для разработки, доработки и настройки программных компонент СПО, а для компоненты СПО может быть применен сканер безопасности Nessus.

Предполагается, что АСИ СЗИ работает по схеме независимых испытаний. При этом число успешных атак распределено по биномиальному закону После тестирования на ИБ i-го СЗИ АСУ можно вычислить частоту нарушения ИБ $p^* = \frac{m}{n}$. Построив для нее доверительный интервал $I_\beta = (p_1, p_2)$, в который частота события нарушения

ИБ ОИ p^* попадает с доверительной вероятностью β, и учтя взаимное расположение расчетного (заданного) допустимого значения вероятности P_i относительно интервала $I_\beta = (p_1, p_2)$, для ОИ можно сформировать ряд правил, которые позволяют сделать вывод о качестве работы СЗИ ОИ.

Заключение

В статье приведены подходы для создания прикладной теории информационной безопасности, методики формирования требований по информационной безопасности АСУ, модели и методы повышения информационной безопасности АСУ и оценки их защищенности.

Литература

1. Моисеев Н.Н. Математические задачи системного анализа. М.: Наука, 1981. 488 с.
2. Моисеев В.С., Дятчин В.В, Тутубалин П.И. Оценка требуемых вероятностей обеспечения информационной безопасности // Вестник КГТУ им. А.Н. Туполева. №4. С. 36–39.
3. Романец Ю.В., Тимофеев П.А., Шаньгин В.Ф. Защита информации в компьютерных системах и сетях. М.: Радио и связь, 1999. 328 с.
4. Казарин О.В. Безопасность программного обеспечения компьютерных систем. М.: МГУЛ, 2003, 212 с.
5. Моисеев В.С. Анализ функционирования СУ предприятием на базе информационной модели. // ИВУЗ «Авиационная техника», №2. 1973. С. 15–22.
6. Кук Д., Бейз Г. Компьютерная математика. М.: Наука, 1990. 383 с.
7. Подников В.В., Ногин В.Д. Парето-оптимальные решения многокритериальных задач. М.: Наука, 1982. 254 с.

ПРОЕКТ УЧЕБНО-МЕТОДИЧЕСКОГО КОМПЛЕКСА ОЦЕНКИ РИСКОВ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

Ю.А. Торшенко
Научный руководитель – д.т.н., профессор Л.Г. Осовецкий

Рассмотрен проект учебно-методического комплекса, позволяющему решить проблему получения студентами практических навыков в области оценки рисков информационной безопасности.

Введение

На сегодняшний день автоматизированные системы (АС) играют ключевую роль в обеспечении эффективного выполнения бизнес-процессов как коммерческих, так и государственных предприятий. Вместе с тем повсеместное использование АС для хранения, обработки и передачи информации приводит к повышению актуальности проблем, связанных с их защитой. Подтверждением этому служит тот факт, что за последние несколько лет как в России, так и в ведущих зарубежных странах имеет место тенденция увеличения числа информационных атак, приводящих к значительным финансовым и материальным потерям. Для того чтобы гарантировать эффективную защиту от информационных атак злоумышленников, компаниям необходимо иметь грамотно построенную систему управления информационной безопасностью (СУИБ).

Оценка риска является первым шагом реализации программы обеспечения информационной безопасности. Обеспечение безопасности рассматривается как набор политик и соответствующих средств контроля, предназначенных для обеспечения бизнес-процессов и уменьшения соответствующих рисков [1]. Для приобретения студентами практических навыков в этой области был разработан проект учебно-методического комплекса оценки рисков информационной безопасности (УМК).

Нормативно-техническая база

Нормативно-техническая основа процесса оценки базируется на документах, которые выступают в качестве технического руководства, которому нужно следовать при оценках рисков ИБ для АС, информационных технологий (ИТ) или организации, для учета потребностей бизнеса организации и технических деталей оценки рисков.

В конце 2000 г. международный институт стандартов ISO (the International Organization for Standartization) и IEC (the International Electrotechnical Commission) на базе британского BS 7799 разработал и выпустил международный стандарт менеджмента безопасности ISO/IEC 17799 [2], который является официальным документом, регламентирующим все вопросы информационной безопасности [3]. Стандарт BS 7799 был создан на основе документа PD 0003 «Практические правила управления информационной безопасностью», разработанного Министерством торговли и промышленности Великобритании при участии специалистов ряда ведущих компаний и организаций, таких как Shell UK, Unilever, British Telecommunications, Британского компьютерного общества, Ассоциации британских страховщиков и др.

Согласно замыслу, основной задачей стандарта является разработка инструмента для создания эффективных систем информационной безопасности государственных и коммерческих организаций на основе современных методов менеджмента. В этом нормативном документе содержится исчерпывающий набор подходов к управлению безопасностью, которые включают в себя самые совершенные процедуры обеспечения информационной безопасности, используемые в разных странах.

Стандарт состоит из трех частей, две из которых, BS 7799-1 и BS 7799-2, были приняты в 2005 г. Международной организацией по стандартизации и Международной

электротехнической комиссией как ISO/IEC FDIS 17799:2005 Information technology – Security techniques – Code of practice for information security management и ISO/IEC FDIS 27001:2005 Information technology – Security techniques – Information security management systems – Requirements [4], соответственно.

ISO/IEC 17799 содержит свод правил по управлению информационной безопасностью и используется в качестве критерия для оценки механизмов безопасности организационного уровня, включая административные, процедурные и физические меры защиты. Этот стандарт определяет общую организацию, классификацию данных, системы доступа, направления планирования, ответственность сотрудников, использование оценки риска и т.д. в контексте информационной безопасности. Данный стандарт представлен в форме руководящих принципов и практических рекомендаций.

ISO/IEC 27001 рассматривает аспекты ИБ с точки зрения сертификации или аудита системы на соответствие требованиям этого стандарта. В этом документе указаны условия создания системы управления для проведения ее аудита или сертификации, а также перечислены требования, применительно к которым проводится проверка соответствия [5].

Одной из целей данного проекта было систематизировать материал описанных выше стандартов для более эффективного их изучения студентами. Для этого был разработан сводный глоссарий информационной безопасности, включающий в себя термины, используемые в стандартах ISO/IEC 17799 и ISO/IEC 27001, в него также были введены определения, описанные в государственном стандарте РФ ГОСТ Р 51897-2002 «Менеджмент риска. Термины и определения» [6]. На основе глоссария и материалов использованных в нем стандартов был создан и учебно-методического комплекс оценки рисков информационной безопасности, включающий в себя методические указания для проведения лабораторной работы по изучению международного стандарта ISO/IEC 27001:2005.

Учебно-методический комплекс

Для приобретения студентами практических навыков по оценке рисков информационной безопасности и построению СУИБ был разработан учебно-методический комплекс оценки рисков информационной безопасности. УМК состоит из нескольких документов, написанных на языках HTML и Java Script. Для запуска УМК необходимо наличие Web-браузера, поддерживающего Java Script (например, Internet Explorer 4.0 и выше). УМК представляет собой набор учебных материалов, объединенных в комплекс, обладающей общей панелью навигации.

Все части УМК, написанные на языке гипертекстовой разметки, осуществляют только вывод справочной информации и не содержат сложных алгоритмических структур. Гипертекстовый файл test.htm также включает в себя программную вставку на языке Java Script, реализующую механизм оценки результатов тестирования.

Тестирование

В состав УМК входит система тестирования (рис. 1), состоящая из 21 вопроса по материалам комплекса. Система автоматически контролирует повторное прохождение теста, а также осуществляет оценку результатов работы (рис. 2) в соответствии с количеством правильных ответов (таблица).

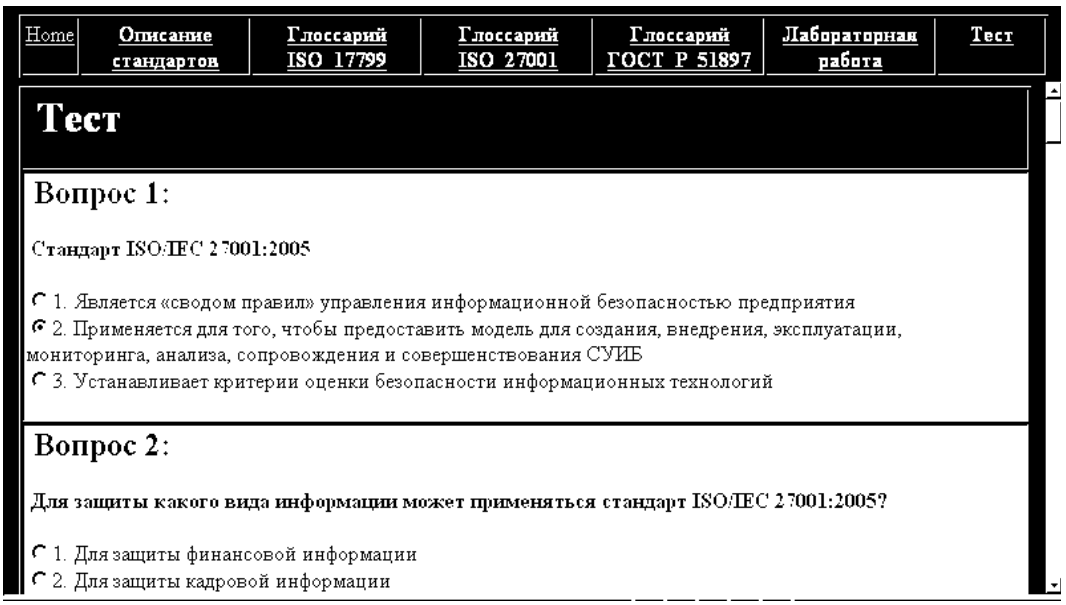


Рис. 1. Система тестирования

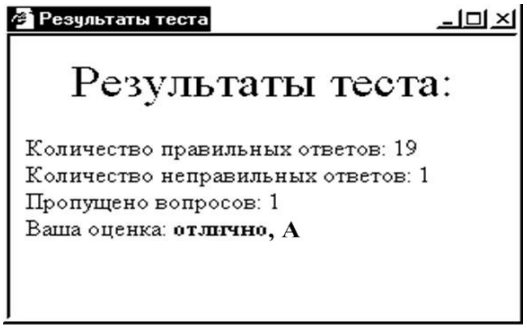


Рис. 2. Окно вывода результатов

Количество правильных ответов	Оценка	
19-21	отлично	A
16-18	хорошо	B
13-15	хорошо	C
9-12	удовлетворительно	D
7-9	удовлетворительно	E
0-6	неудовлетворительно	FX

Таблица. Оценка результатов тестирования

Лабораторная работа

В дополнение к УМК была разработана лабораторная работа по изучению стандарта ISO 27001. Лабораторная работа рассчитана на 4–6 академических часов и состоит из четырех этапов, каждый из которых оценивается отдельно, а затем на основе полученных результатов преподавателем ставится общая оценка.

Группа (1–3 чел.) выполняет 1 и 2 этапы цикла PDCA относительно построения СУИБ, а именно планирование (создает политику СУИБ, описывает цели, процессы и процедуры, необходимые для управления риском и улучшения информационной безопасности, приводит политику СУИБ в соответствие с общей политикой безопасности организации и ее целями) и осуществление (описывает мероприятия по внедрению и управлению политикой СУИБ, менеджментом, процессами и процедурами).

Две группы студентов меняются своими результатами и осуществляют проверку (производят оценку работы предпринятых мер против угроз информационной безопасности). Затем происходит обратный обмен результатами, группа осуществляет корректирующие действия (предпринимает корректирующие и профилактические меры, необходимость применения которых основана на результатах проверки, планирует мероприятия по достижению непрерывного усовершенствования СУИБ).

Последним этапом лабораторной работы является индивидуальный тест на знание стандарта ISO 27001 и других справочных материалов, приведенных в учебно-методическом комплексе. Тест проводится средствами программы test.htm, входящей в состав УМК.

В случае возникновения спорной ситуации или по желанию преподавателя могут быть заданы контрольные вопросы.

Заключение

В процессе разработки данного проекта были изучены стандарты ISO/IEC 17799 и ISO/IEC 27001, затрагивающие проблемы оценки рисков в области информационной безопасности и построения комплексной системы управления информационной безопасностью (СУИБ).

Для приобретения студентами практических навыков по оценке рисков информационной безопасности и построению СУИБ был разработан учебно-методический комплекс оценки рисков информационной безопасности (УМК), состоящий из нескольких документов, написанных на языках HTML и Java Script, и включающий в себя сводный глоссарий стандартов информационной безопасности, тексты стандартов, тестирование и методические указания по проведению лабораторной работа по изучению стандарта ISO 27001.

УМК отвечает всем поставленным задачам: содержит справочные материалы по стандартам ISO/IEC 17799, ISO/IEC 27001 и ГОСТ Р 51897, включает в себя инструментарий, необходимый для получения практических навыков в рассматриваемой области, и систему оценки теоретических знаний.

Литература

1. Носаков В. Создание комплексной системы управления информационной безопасностью // Jet Info. 2006. №7. (http://www.jetinfo.ru).
2. Международный стандарт ISO/IEC FDIS 17799:2005 Information technology – Security techniques – Code of practice for information security management
3. Медведовский И. ISO 17799: Эволюция стандарта в период 2002–2005, 2005 (http://www.dsec.ru/).
4. Международный стандарт ISO/IEC FDIS 27001:2005 Information technology - Security techniques – Information security management systems – Requirements.
5. Буянова И.С. Современная система управления информационной безопасностью с новым стандартом ISO/IEC 27001:2005, 2006 (http://www.traectoria.ru).
6. Государственный стандарт РФ ГОСТ Р 51897-2002 Менеджмент риска. Термины и определения.

МОДИФИЦИРОВАННАЯ МОДЕЛЬ ОБЩЕГО КОНТЕКСТА БЕЗОПАСНОСТИ НА ОСНОВЕ ОБЩИХ КРИТЕРИЕВ В КОНТЕКСТЕ КОНКУРЕНТНОГО ПАРНО-СУБЪЕКТНОГО ВЗАИМОДЕЙСТВИЯ

А.С. Пастухов
Научный руководитель – д.т.н., профессор Л.Г. Осовецкий

Статья затрагивает некоторые проблемы, связанные с фундаментальным подходом стандарта «Общие критерии» и его аутентичного перевода РД БИТ к Общему контексту безопасности (ОКБ). Приводится расширенное представление о характере взаимодействия элементов ОКБ, вводится конкурентный характер противостояния субъектов ОКБ и указывается дуальная направленность данного противостояния. В статье так же приводится математический аппарат, реализующий данное взаимодействие.

Введение

Информационные технологии играют все большую роль в современном мире. Совершающийся на наших глазах переход человеческого общества от постиндустриального к информационному влечет за собой как новые преимущества, так и новые проблемы. Информация, являющаяся нематериальным ресурсом, постепенно превращается в главнейшую составляющую человеческой деятельности. Тот, кто владеет информацией, правит современным миром. Поэтому современные высокие технологии немыслимы без защиты информации.

Вопросы обеспечения информационной безопасности исследуются в разных странах достаточно давно. Со временем люди, занимающиеся данной проблемой, пришли к выводу, что существование серьезного и комплексного подхода к защите информации не возможно без закрепления всех разработанных методов в специальных документах – стандартах. Одним из таких стандартов являются «Общие критерии оценки безопасности информационных технологий» (Common Criteria for IT Security Evaluation). За этим документом исторически закрепилось более короткое название – «Общие критерии», ОК (Common Criteria, CC).

В Российской Федерации ОК получили распространение в виде своего аутентичного перевода – Руководящего документа Гостехкомиссии РФ по безопасности информационных технологий (РД БИТ).

На наш взгляд, одной из недоработок ОК (РД БИТ) является концентрация стандарта на оценке ИБ как таковой и упрощение представления о взаимодействии субъектов в контексте модели отношений высокоуровневых понятий безопасности.

В работе поставлены следующие задачи:

1. поэлементно описать модель взаимодействия высокоуровневых понятий безопасности в нотации РД БИТ. Провести формальный анализ структур субъектов, взаимодействующих в данной модели, и расширить характер их взаимодействия с однополярного до дуального (парно-субъектного);
2. модифицировать модель с учетом расширенного представления о взаимодействии ее элементов. Проанализировать конкурентную борьбу вышеупомянутых субъектов, выявить ее причины и цели. Учесть важную роль СЗИ (СУИБ) в структуре каждого из конкурирующих субъектов (К-субъектов);
3. на основе эмпирических данных разработать математический аппарат, реализующий конкурентное взаимодействие субъектов.

Системотехническая модель конкурирующего взаимодействия

В [1] приведена схема, иллюстрирующая высокоуровневые понятия безопасности и их взаимосвязь. Данная схема представляет собой системотехническую модель, до-

полненную текстовыми комментариями. Рассмотрим основные элементы модели и связи между ними подробнее.

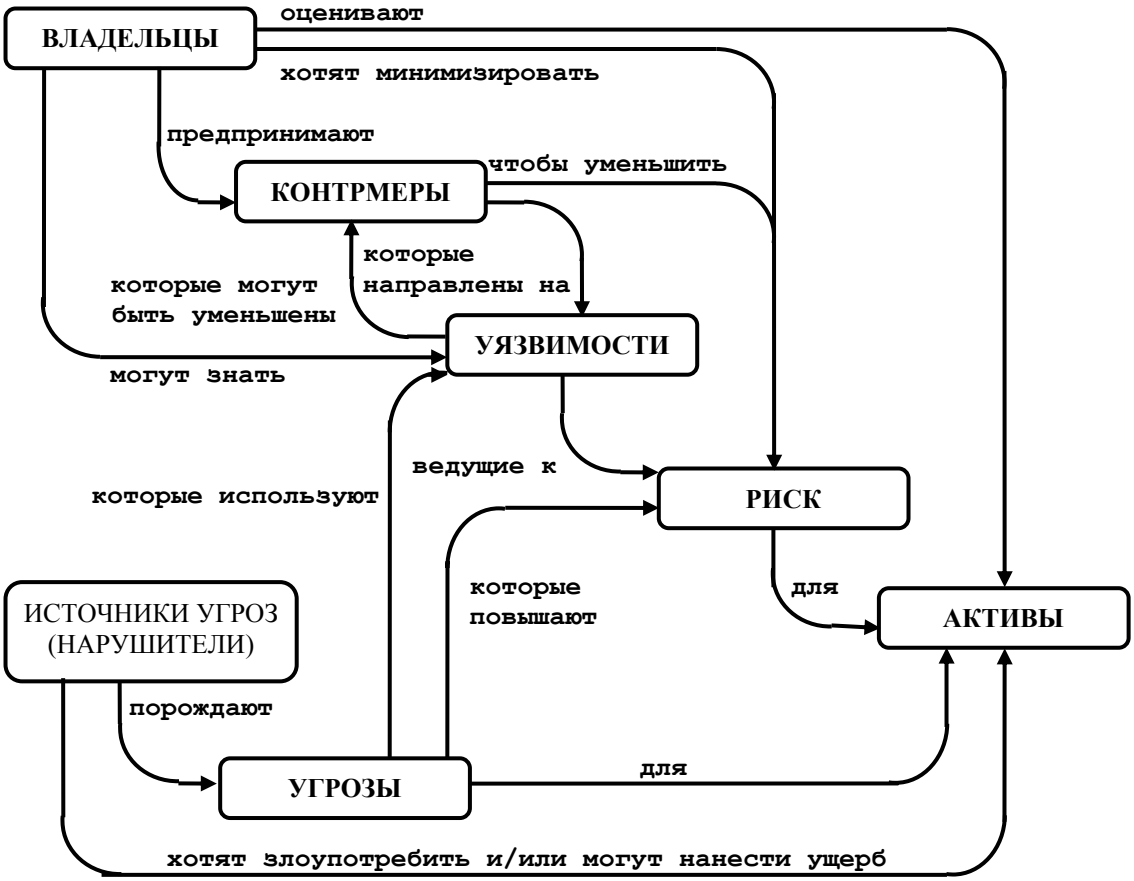


Рис. 1. Понятия безопасности и их взаимосвязь

За сохранность рассматриваемых активов отвечают их владельцы, для которых эти активы имеют ценность. Существующие или предполагаемые нарушители также могут придавать значение этим активам и стремиться использовать их вопреки интересам их владельца. Владельцы будут воспринимать подобные угрозы как потенциал воздействия на активы, приводящего к понижению их ценности для владельца. К специфическим нарушениям безопасности обычно относят (но не обязательно ими ограничиваются): наносящее ущерб раскрытие актива несанкционированным получателем (потеря конфиденциальности), ущерб активу вследствие несанкционированной модификации (потеря целостности) или несанкционированное лишение доступа к активу (потеря доступности).

Владельцы активов будут анализировать возможные угрозы, чтобы решить, какие из них действительно присущи их среде. В результате анализа определяются риски. Анализ может помочь при выборе контрмер для противостояния угрозам и уменьшения рисков до приемлемого уровня.

Контрмеры предпринимают для уменьшения уязвимостей и выполнения политики безопасности владельцев активов (прямо или косвенно распределяя между этими составляющими). Но и после введения этих контрмер могут сохраняться остаточные уязвимости. Такие уязвимости могут использоваться нарушителями, представляя уровень остаточного риска для активов. Владельцы будут стремиться минимизировать этот риск, задавая дополнительные ограничения.

Таким образом, данная модель является моделью конкурентного взаимодействия двух субъектов – Владельца и Нарушителя, борющихся за обладание активом. При

этом Владелец использует исключительно защитные методы – политику информационной безопасности, анализ рисков и другие контрмеры. Нарушитель, соответственно, использует исключительно методы нападения – порождает различные угрозы, анализирует защитные методы Владельца, ищет «дыры» в политике информационной безопасности. Полученную модель, поэтому можно назвать «парно-субъектной» по характеру взаимодействующих элементов.

Так что же происходит в реальном мире в контексте безопасности информационных технологий? Здесь каждый из элементов нашей модели перестает быть абстрактным субъектом и обретает плоть.

Например, Владелец – это некая фирма A, которая обладает важным информационным активом V (ноу-хау). Данный актив V позволяет фирме A занимать лидирующее положение на рынке и получать прибыль.

Нарушитель – это некая фирма B, которая хочет злоупотребить и\или нанести ущерб активу V, принадлежащему фирме A. С этой целью фирма B порождает различные угрозы в отношении фирмы A с целью преодолеть защитные контрмеры, создаваемые политикой информационной безопасности фирмы A, и похитить либо нанести ущерб активу V.

Какие же нарушения безопасности угрожают активу V?

- Нанесение ущерба путем раскрытия актива несанкционированным получателем (потеря конфиденциальности);
- ущерб активу вследствие несанкционированной модификации (потеря целостности);
- несанкционированное лишение доступа к активу (потеря доступности).

Чем данные нарушения могут быть выгодны руководству компании B?

- Потеря конфиденциальности активом V может нивелировать лидирующее положение фирмы A на рынке, вследствие потенциальной возможности других компаний использовать актив V в своих собственных целях.
- Потеря целостности или доступности активом V также может привести к потере фирмой A лидирующего положения на рынке, вследствие невозможности фирмой A в дальнейшем пользоваться данным активом V.

В каноническом представлении РД БИТ рассмотренные субъекты выглядят следующим образом.

Фирма A (Владелец), как уже было сказано, обладает активом V, которому угрожают риски R, которые образуются вследствие наличия у фирмы A уязвимостей H. Для защиты актива V фирма A применяет различные контрмеры G, призванные снизить риски R для актива V, которые направлены на уязвимости H.

Фирма B (Нарушитель) порождает угрозы U и хочет злоупотребить и\или причинить ущерб активу V, принадлежащему фирме A.

В подобном случае каждый из субъектов – фирма A и фирма B – выполняют свою собственную функцию – защитную и атакующую. Но фирма сама по себе не может выполнять подобную функцию. Этим должен заниматься один из ее элементов. В случае фирмы A таким элементом является система защиты информации (СЗИ). Данная система создается исключительно для защиты актива V от угроз U, т.е. для реализации контрмер G. У фирмы B в ее каноническом виде по РД БИТ должен быть исключительно один элемент, отвечающий за порождение угроз U.

Возникает закономерный вопрос – какая же польза от этого фирме B? Почему фирма B хочет завладеть либо причинить ущерб активу V? Подобное желание легко объяснить – фирма B на самом деле является конкурентом фирмы A. В таком случае фирме B выгодно злоупотребить и\или причинить ущерб активу V, принадлежащему фирме A, поскольку это позволит фирме B упрочить свое положение на рынке и победить в конкурентной борьбе.

Логично предположить, что, раз фирма А и фирма В функционируют на одном рынке и являются конкурентами, структура фирм должна быть похожа. Единственное структурное различие будет заключаться в различных функциях отдела информационной безопасности. У фирмы А этот отдел выполняет защитную функцию, а у фирмы В – функцию нападения. По аналогии с фирмой А представим фирму В в виде модели.

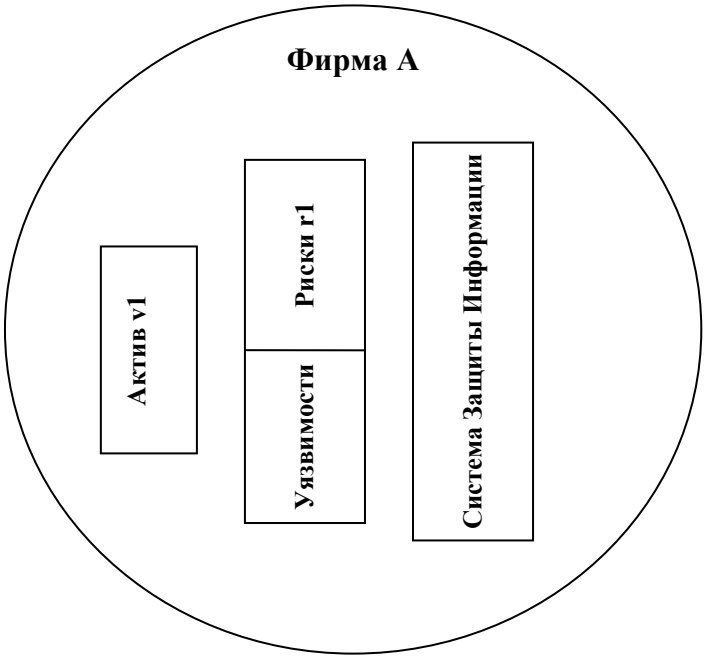


Рис. 2. Структура субъекта «Владелец» (Фирма А)

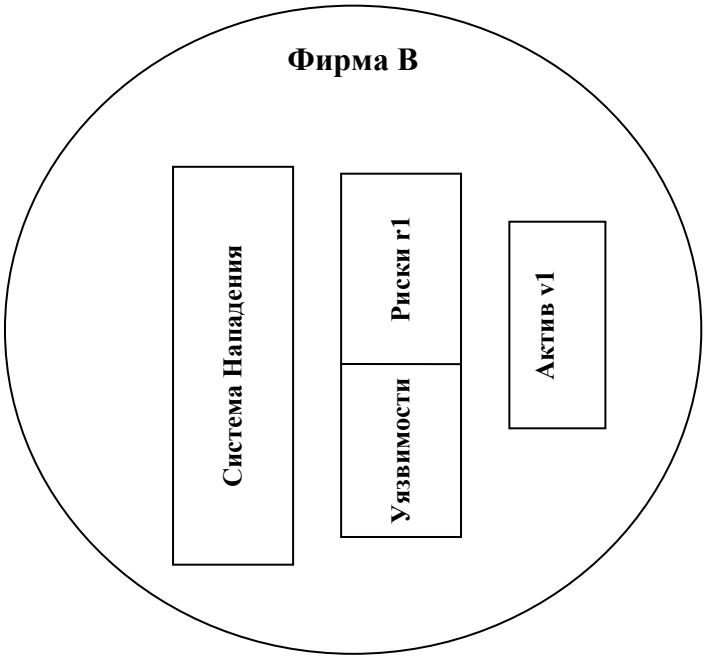


Рис. 3. Структура субъекта «Нарушитель» (Фирма В)

Таким образом, фирма В, так же как и фирма А, обладает активом V2, который она ценит и хочет сохранить. Этим активам v2 угрожают риски r2, которые могут использовать уязвимости u2. Но о каких уязвимостях может идти речь, если мы предполагаем наличие у фирмы В только системы нападения на актив v1? При такой структуре фирма В очень быстро прекратила бы свое существование вследствие отсутствия СЗИ. Однако фирма В надеется занять лидирующее положение на рынке, для чего порождает различные угрозы активу v1. Следовательно, у фирмы В тоже есть СЗИ, кото-

рая защищает актив v2 от угроз u2. Но кто же в таком случае порождает угрозы U2? В нашей ситуации, если на рынке нет больше никаких участников, кроме фирм А и В, таким источником угроз U2 может служить только система нападения фирмы А. Логично также предположить, что эти функции в обоих системах (фирмах) исполняет один и тот же элемент – СЗИ.

Таким образом, в результате вышеприведенных умозаключений мы приходим к иной трактовке взаимодействия наших субъектов А и В. Не существует никаких «хороших» владельцев и «плохих» нарушителей. На рынке идет конкурентная борьба между двумя СЗИ, каждая из которых сформирована в собственной фирме. Основными задачами таких СЗИ являются одновременно попытки завладеть чужим активом и не допустить захвата собственного актива.

В результате подобного подхода однополярная модель РД БИТ может быть преобразована в парно-субъектную модель взаимодействия двух конкурирующих субъектов (К-субъектов), а точнее, модель взаимодействия СЗИ этих субъектов (рис. 4).

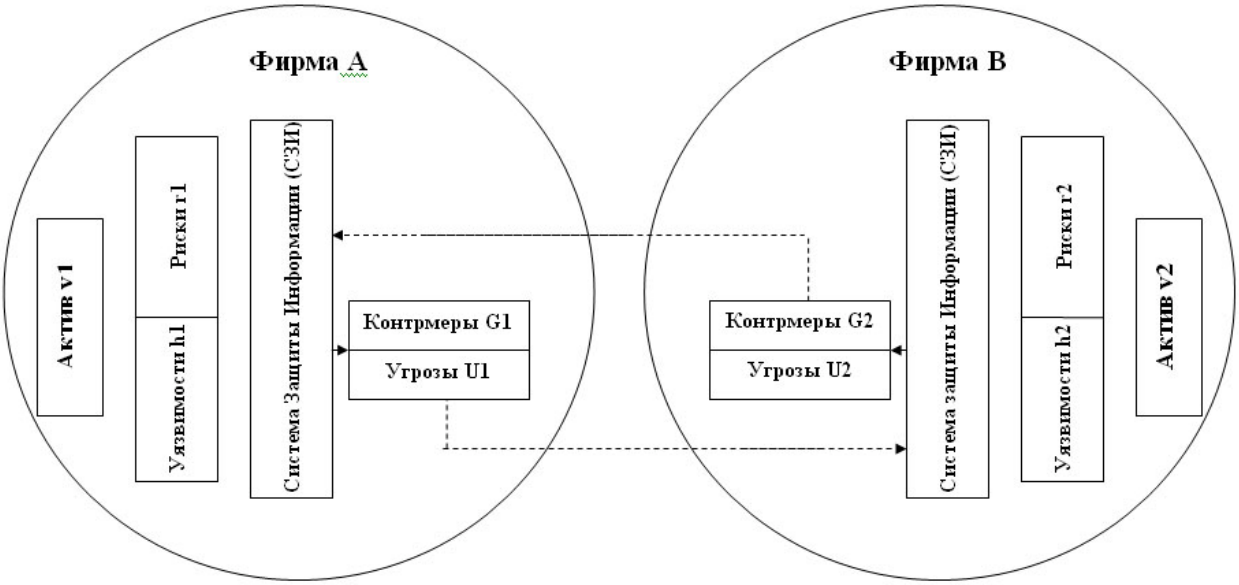


Рис. 4. Парно-субъектная модель конкурирующего взаимодействия

Математический аппарат

Опишем элементы математической модели.

Главные элементы: A1, A2 – активы, принадлежащие соответственно фирме А и фирме В; Y1, Y2 – уязвимости в СЗИ фирмы А и фирмы В, соответственно; J1, J2 – угрозы, порождаемое в отношении друг друга фирмами А и В.

Побочные элементы: Jот1, Jот2 – все угрозы, отраженные СЗИ фирм А и В, соответственно; Jота1, Jота2 – угрозы, активно отраженные СЗИ фирм А и В.

Коэффициенты:

$Кз1=\frac{Jот1}{J1}$, $Кз2=\frac{Jот2}{J2}$ – коэффициенты успешности выполнения защитных функций СЗИ фирм А и В;

$Ка1=\frac{Jота1}{Jот1}$, $Ка2=\frac{Jота2}{Jот2}$ – коэффициенты, показывающие успешность атак, порождаемых СЗИ фирм А и В.

Кот1, Кот2 – коэффициенты, показывающие успешность отражения угроз СЗИ фирм А и В.

Рассмотрим частный случай, в котором

Y1=Y2, A1=A2, Кот1 = Кот2.

Подобные условия необходимы для упрощения вычислений и большей точности при выражении конкурентных отношений наших субъектов.

Зададим дополнительное ограничение:

$Кз1 \geq Кз2$,

откуда следует, что

$$\frac{Кз1}{Кз2} \geq 1.$$

Запишем данное неравенство, но за основу возьмем Ка1 и Ка2. Получим

$$\frac{Ка2}{Ка1} \bullet \frac{J2^2}{J1^2} \geq 1.$$

Данное соотношение наглядно демонстрирует решающее влияние угроз J1, J2 на нашу модель.

Введем ***функцию защищенности*** $F\left(\frac{Ka1}{Ka2}\right)$, показывающую зависимость коэффициента защищенности Кз1 от отношения коэффициентов атаки.Ка1 и Ка2. Получим следующие отношения:

$$Кз1 = \frac{1}{Ka2}; Кз2 = \frac{1}{Ka1},$$

Условием ***баланса защищенности*** в конкурентной борьбе фирм А и В является обратная пропорциональность отношения коэффициентов защищенности к отношению коэффициентов успешности атак:

$$\frac{Кз1}{Кз2} = \frac{Ka2}{Ka1}.$$

Заключение

В статье были решены следующие задачи.

1. Поэлементно описана модель взаимодействия высокоуровневых понятий безопасности в нотации РД БИТ. Проведен формальный анализ структур субъектов, взаимодействующих в данной модели, расширен характер их взаимодействия с однополярного до дуального (парно-субъектного).

2. Модель модифицирована с учетом расширенного представления о взаимодействии ее элементов. Проанализирована конкурентная борьба вышеупомянутых субъектов, выявлены ее причины и цели. Учтена роль СЗИ (СУИБ) в структуре каждого из конкурирующих субъектов (К-субъектов).

3. На основе эмпирических данных разработан математический аппарат, реализующий конкурентное взаимодействие субъектов.

Литература

1. Руководящий документ Гостехкомиссии РФ «Безопасность информационных технологий», 2002. С. 8.

К ОБНАРУЖЕНИЮ АТАК В КОМПЬЮТЕРНЫХ СИСТЕМАХ НЕЙРОСЕТЕВЫМИ СРЕДСТВАМИ

А.А. Марченко, С.В. Матвиенко, Ф.Г. Нестерук
Научный руководитель – д.т.н., доцент Г.Ф. Нестерук

Анализ публикаций о применении интеллектуальных средств для решения задач защиты информации, в частности, посвященных системам обнаружения компьютерных атак (СОА), показывает, что основным механизмом обнаружения компьютерных атак является интеллектуальный анализ данных, а основными средствами для реализации интеллектуального анализа данных – экспертные системы, нейронные сети, системы нечеткой логики и гибридные нейро-нечеткие и нейро-экспертные системы.

Введение

К задачам, решаемым методами интеллектуального анализа данных, относятся [1]:

1) классификация;
2) прогнозирование;
3) кластеризация;
4) ассоциация;
5) последовательность;
6) *визуализация данных*.

Первые две задачи решаются с использованием *предсказательных* моделей: по набору обучающих данных с заранее известными результатами создаются модели, которые с большой долей уверенности предсказывают результаты для реальных данных. Последующие три задачи представляются *описательными* моделями: по зависимостям, выявленным в известных данных, создаются модели, которые используются в системах принятия решений. Визуализация информации состоит в отображении многомерного пространства параметров задачи в пространства меньшей размерности (2-, 3-мерные изображения), удобные для представления средствами компьютерной графики.

Перечисленные задачи с успехом могут быть использованы для обнаружения компьютерных атак и соответствуют традиционным областям применения экспертных систем, нейронных сетей, систем нечеткой логики и гибридных интеллектуальных систем.

Классификация систем обнаружения атак

Распространенная классификация СОА – по принципу реализации: host-based, т.е. обнаружение атаки, направленной на конкретный узел сети, и network-based – направленной на сегмент сети или сеть в целом. Системы класса host-based подразделяют на три подуровня (рис. 1) [2]:

- Application IDS, обнаруживающие атаки на конкретные приложения;
- OS IDS, обнаруживающие атаки на операционные системы;
- DBMS IDS, обнаруживающие атаки на системы управления базами данных.

Другим классификационным признаком может служить технология обнаружения атак (рис. 2):

- системы, обнаруживающие аномальное поведение (anomaly detection);
- системы, обнаруживающие злоупотребления (misuse detection).

Первые основаны на профилях нормального поведения пользователя, системной деятельности или сетевого трафика. Данные наблюдения сравниваются с ожидаемыми профилями нормального поведения, которые строится в период обучения СОА. Системы второго классы основаны на описании атаки в виде сигнатуры (signature) или правила и поиска данной сигнатуры в контролируемом пространстве (сетевом трафике, журнале регистрации и т.д.). В качестве сигнатуры атаки может выступать шаблон дей-

ствий или строка символов, характеризующие злоупотребления. Сигнатуры хранятся в базе данных, аналогичной той, которая используется в антивирусных системах. Антивирусные резидентные мониторы являются частным случаем СОА. База сигнатур (правил), как правило, создается с помощью экспертного метода.

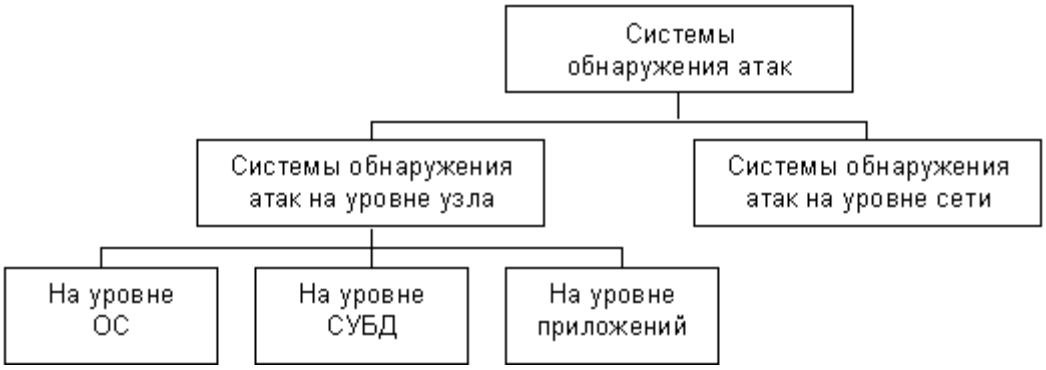


Рис. 1. Классификация систем обнаружения атак по принципу реализации

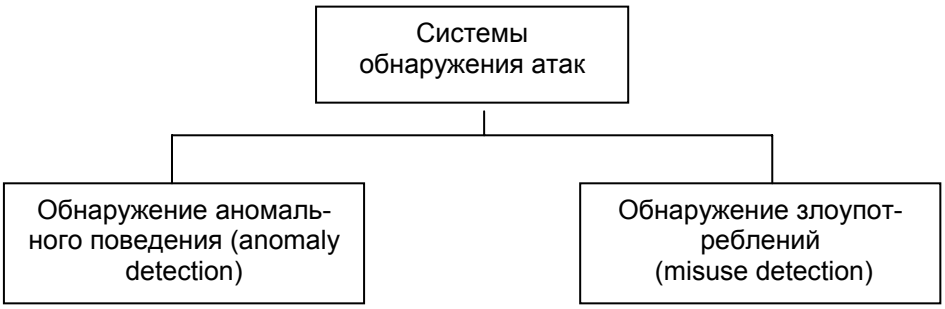


Рис. 2. Классификация систем обнаружения атак по технологии обнаружения атак

Рассмотрим нейросетевой подход к организации СОА, классифицируемой по принципу реализации как host-based система, а по технологии обнаружения атак – как система обнаружения аномального поведения.

Искусственные нейронные сети в системах обнаружения атак

Большинство подходов к обнаружению атак используют форму анализа контролируемого пространства на основе правил или статистических методов. В качестве контролируемого пространства могут выступать журналы регистрации или сетевой трафик. Анализ опирается на набор заранее определенных правил, которые создаются администратором или самой СОА. Экспертные системы (ЭС) обнаружению атак строятся на основе правил. ЭС состоит из набора правил, которые соответствуют знаниям эксперта. ЭС требуют постоянного обновления для того, чтобы оставаться актуальными. Отсутствие обновления ЭС снижает защищенность сети, вводя пользователей в заблуждение относительно действительного уровня защищенности.

Сетевые атаки постоянно изменяются, поскольку злоумышленники используют индивидуальные подходы, а также в связи с регулярными изменениями в программном обеспечении и аппаратных средствах компьютерных систем. Развертывание атаки во времени либо одновременные действия нескольких злоумышленников затрудняют обнаружение атаки с помощью ЭС. Из-за разнообразия атак даже постоянные обновления правил ЭС не дают гарантии точной идентификации атак.

Одним из путей устранения названных проблем является использование нейронных сетей (НС). В отличие от ЭС, которые могут дать пользователю определенный ответ, соответствуют или нет рассматриваемые характеристики характеристикам, заложенным в базе правил, НС проводит анализ информации и предоставляет возможность

оценить, согласуются ли данные с характеристиками, которые она научена распознавать. Степень соответствия нейросетевого предсказания может достигать 100% и зависит от качества обучения.

Вначале НС обучается правильной идентификации предварительно выбранных примеров предметной области. Реакция НС анализируется, и система настраивается таким образом, чтобы достичь заданной точности. В дополнение к первоначальному периоду обучения НС также набирается опыта по мере того, как она проводит анализ данных, связанных с предметной областью. Важное преимущество НС при обнаружении злоупотреблений заключается в способности изучать характеристики атак и идентифицировать элементы, которые не похожи на те, что наблюдались в сети прежде.

Принципы нейросетевого мониторинга безопасности

Совокупность значений параметров (показатели сетевых пакетов или состояния системных ресурсов, данные о последовательностях системных вызовов или разбора лог-файлов) может характеризовать состояние контролируемого узла компьютерной системы (КС) в каждый момент времени. При этом дискретизация времени может зависеть от конкретных настроек подсистемы сбора данных. Таким образом, некоторую комбинацию значений X_i, полученную в очередной момент времени, можно считать входным воздействием (вектором) СОА. Тогда результатом анализа такой комбинации будет некоторая величина h_i, представляющая ответ на вопрос, является ли представляемое ею состояние нормальным поведением или проявлением конкретной атаки.

Предположим, что в функционировании КС имеются не известные априори закономерности, которые характеризуются группой сходных векторов – скоплением точек в многомерном пространстве. Описание их называют профилем штатного поведения узла. Тогда, если очередной вектор попадает в одну из описанных областей, фиксируется один вид поведения (нормальное или аномальное), если не попадает, то противоположный. Задача выявления аномалий и атак, таким образом, заключается в составлении профиля поведения путем наблюдения за заведомо исправным узлом или узлом, подвергающимся конкретной атаке, и в последующем поиске отклонений от сформированного профиля во время эксплуатации системы.

Решение задачи формирования профиля может быть получено двумя путями – как распознавание образов и как кластеризация. В обоих случаях профиль будет представлять собой разбиение пространства признаков на непересекающиеся области.

В случае *распознавания* образов необходимо обеспечить отнесение каждого входного вектора к одному из нескольких классов, представляющих конкретные компьютерные атаки или нормальное поведение. В случае *кластеризации* необходимо выявить группы похожих векторов, которые все относятся либо к аномальному поведению, либо к нормальному. Тогда остальные вектора, не попавшие в указанные группы, можно отнести к противоположному поведению.

Создание профиля предполагает предварительное формирование множества примеров. Так, чтобы обучить некую систему распознаванию нескольких классов воздействий, необходимо иметь примеры (векторы), относящиеся ко всем этим классам. Для получения «нормальных» векторов следует зафиксировать ряд состояний КС в течение определенного времени. Для получения же «аномальных» векторов потребуется воздействовать на КС искусственным образом, чтобы зафиксировать нужные комбинации параметров. Причем такую работу необходимо периодически повторять, чтобы учитывать постепенное изменение профиля поведения контролируемой КС.

НС применяются для решения вышеописанных задач. Для распознавания образов часто применяются НС типа многослойного персептрона, но наиболее распространен-

ными являются самоорганизующиеся карты Кохонена (SOM – Self-Organizing Map) и сети адаптивного резонанса ART [3].

Сети теории адаптивного резонанса

Сети адаптивного резонанса или классификаторы Карпентера-Гроссберга (ART – Adaptive Resonance Theory Network) применяются для кластеризации многомерных векторов. Главной особенностью ART является то, что процесс обучения НС и ее эксплуатации не разделяются. ART представляют собой потоковые алгоритмы кластеризации (без предварительного накопления обучающей выборки) с изменяемым количеством кластеров, что значительно уменьшает время обучения, хотя и снижает скорость работы [4–6].

Сети ART имеют несколько модификаций, основные из которых рассмотрены ниже.

ART1. Сети ART1 [4] предназначены для обработки бинарных векторов. НС состоит из двух слоев нейронов (рис. 3): входного (сравнительного), число нейронов в котором фиксировано и равно размерности входных векторов, и выходного (соревновательного) с переменным количеством нейронов, где каждому нейрону соответствует класс объектов.

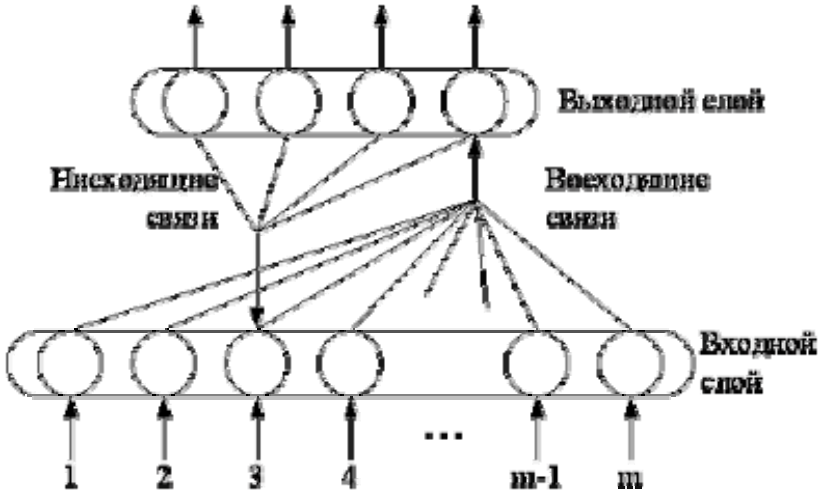


Рис. 3. Нейронная сеть ART1

Отметим особенности ART1.

1. Количество входов и весов входов каждого нейрона определяется следующим правилом. Каждый нейрон входного слоя соединен с каждым нейроном выходного слоя и, наоборот, каждый нейрон выходного слоя связан с каждым нейроном входного. Каждой связи поставлен в соответствие вес. Связи от входных нейронов к выходным называют восходящими, а от выходных к входным – нисходящими. В качестве функции состояния каждого нейрона рассматривается расстояние в некоторой метрике между входным вектором X и вектором весов нейрона W.

2. Функция активации нейронов выходного слоя вычисляется как взвешенная весами сумма координат входного вектора.

3. Обучение НС является неуправляемым (без учителя) и заключается в коррекции весов связей так, чтобы каждый нейрон был победителем для максимального количества векторов из обучающего множества и при этом выполнялся некоторый критерий близости.

Алгоритм обучения ART1 можно описать следующим образом.

Первый входной вектор считается эталоном первого кластера. Следующий входной вектор сравнивается с эталоном первого кластера. Считается, что он принадлежит этому кластеру, если расстояние до эталона в некоторой метрике меньше заданного порога. При этом координаты эталона корректируются по определенному правилу. В противном случае второй вектор становится эталоном второго кластера. Данный процесс повторяется для всех последующих входных векторов. Число кластеров, таким образом, постепенно растет в зависимости от заданной величины порога и используемой метрики.

ART2. Сети теории адаптивного резонанса ART2 [4] по своим классификационным признакам, особенностям обучения, структуры и функционирования схожи с сетями ART1. Отличие заключается лишь в том, что обрабатываемые многомерные вектора являются не бинарными, а вещественными. Вследствие этого входной слой сети представляет собой совокупность шести подслоев, каждый из которых включает по m нейронов, где m – размерность входного вектора. Назначение указанных подслоев заключается в нормализации входного вектора и приведения его к виду, близкому к векторам, обрабатываемым сетью ART1.

Fuzzy ART [5]**.** Данная НС является объединением сети ART1 и теории нечетких множеств. Сеть Fuzzy ART обрабатывает многомерные вещественные вектора. Алгоритм работы Fuzzy ART рассматривается ниже.

Векторы полей. ART система включает поле F_0 нейронов, представляющих текущий входной вектор; поле F_1, получающее информацию по восходящим связям из поля F_0 и по нисходящим связям из поля F_2, представляющее активную категорию. Вектор поля F_0 обозначается $I = (I_1,\ldots, I_M)$, каждый элемент которого I_i лежит в интервале [0,1], i=1,..., M. Вектор поля F_1 обозначается $x=(x_1,\ldots, x_M)$, вектор поля F_2 обозначается $y=(y_1,\ldots, y_N)$.

Весовой вектор. С каждой категорией j (j=1,…, N) в поле F_2 связан вектор $w_j \equiv (w_{j1},..., w_{jM})$. Первоначально

$$w_{j1}(0)=\ldots=w_{jM}(0)=1; \quad (1)$$

тогда каждая категория называется *незадействованной*. После выбора категории для классификации она становится *задействованной.* Каждый элемент весового вектора монотонно не убывает во времени и стремится к некоторому пределу.

Параметры. Динамика Fuzzy ART определяется параметром выбора $\alpha > 0;$ параметром скорости обучения $\beta \in [0, 1]$; параметром близости $\rho \in [0,1]$.

Выбор категории. Для каждого входа I и элемента j поля F_2 *функция выбора* T_j определяется как

$$T_j(I) = \frac{|I \wedge w_j|}{\alpha + |w_j|}, \quad (2)$$

где нечеткий оператор $\wedge$ определяется как

$$(p \wedge q)_i \equiv \min(p_i, q_i) \quad (3)$$

и норма вектора $|\cdot|$ определяется как

$$|p| \equiv \sum_{i=1}^{M} |p_i| \quad (4)$$

для любых M-размерных векторов p и q.

ART система выполняет выбор категории, если в каждый момент времени активным является не более одного элемента в поле F_2. Выбранная категория индексируется как J,

$$T_J = \max\{T_j: j=1\ldots N\}. \quad (5)$$

Если более одного T_j максимально, то выбирается категория j с наименьшим индексом. В частности, элементы становятся задействованными в порядке j=1, 2, 3, ….

Когда *J*-ая категория выбрана, $y_J = 1$ и $y_j = 0$ для $j \neq J$. Вектор поля F_1 удовлетворяет уравнению

$$x = \begin{cases} I, \text{ если } F_2 \text{ неактивно} \\ I \wedge w_j, \text{ если J-ая категория } F_2 \text{ выбрана.} \end{cases} \quad (6)$$

Резонанс или сброс. Резонанс происходит, если *функция совпадения* $|I \wedge w_j| / |I|$ выбранной категории удовлетворяет критерию близости:

$$\frac{|I \wedge w_j|}{|I|} \geq \rho \quad ; \quad (7)$$

т.е., в соответствии с (6), когда J-ая категория выбрана, резонанс происходит при

$$|x| = |I \wedge w_j| \geq \rho |I| \quad . \quad (8)$$

Затем следует обучение, как описано ниже. *Сброс* происходит, если

$$\frac{|I \wedge w_j|}{|I|} < \rho \quad (9)$$

или

$$|x| = |I \wedge w_j| < \rho |I| \quad . \quad (10)$$

Тогда значение функции выбора T_j устанавливается в 0 на время обработки текущего входного вектора для предотвращения постоянного выбора одной и той же категории в процессе поиска. Новый индекс *J* выбирается в соответствии с (5). Процесс поиска продолжается, пока выбранная категория *J* не будет удовлетворять (7).

Обучение. После окончания поиска весовой вектор w_J обновляется в соответствии с уравнением

$$w_J^{(new)} = \beta (I \wedge w_J^{(old)}) + (1 - \beta) w_j^{(old)} \quad . \quad (11)$$

Правило *быстрого обучения* соответствует установке $\beta = 1$.

Нормализация входного вектора/Комплементарное кодирование. При исследовании ряда аналоговых ART систем была выявлена проблема быстрого разрастания числа категорий, когда большое число входных векторов изменяло норму весовых векторов. Разрастание числа категорий в Fuzzy ART не происходит, если входные вектора нормализованы, т.е. для некоторого $\gamma > 0$

$$|I| \equiv \gamma \quad (12)$$

для всех входных векторов *I*. Нормализация может быть достигнута путем предобработки каждого входного вектора *a*, например:

$$I = \frac{a}{|a|} \quad . \quad (13)$$

Комплементарное кодирование – правило нормализации, сохраняющее информацию об амплитуде. Входной вектор для поля F_1 в комплементарном коде представляет собой 2M-размерный вектор

$$I = (a, a^c) \equiv (a_1, ..., a_M, a_1^c, ..., a_M^c) \quad , \quad (14)$$

где

$$a_i^c \equiv 1 - a_i \quad . \quad (15)$$

Заметим, что

$$|I| = |(a, a^c)| = \sum_{i=1}^{M} a_i + (M - \sum_{i=1}^{M} a_i) = M \quad , \quad (16)$$

т. е. входные вектора, представленные в комплементарном коде, автоматически нормализуются. При использовании комплементарного кодирования первоначальное условие (1) заменяется на

$$w_{j1}(0) = ... = w_{j,2M}(0) = 1 \quad (17)$$

Fuzzy ARTMAP [6]**.** ARTMAP – класс НС, характеризующихся инкрементным обучением с учителем для решения классификационных задач. Fuzzy ARTMAP обучается классификации входных данных по нечеткому набору признаков – вектору, элементами которого являются значений от 0 до 1, представляющие степень наличия того или иного признака. ARTMAP система включает два модуля ART (ART_a и ART_b), которые создают классификационные категории в ответ на последовательность входных векторов (рис. 4).

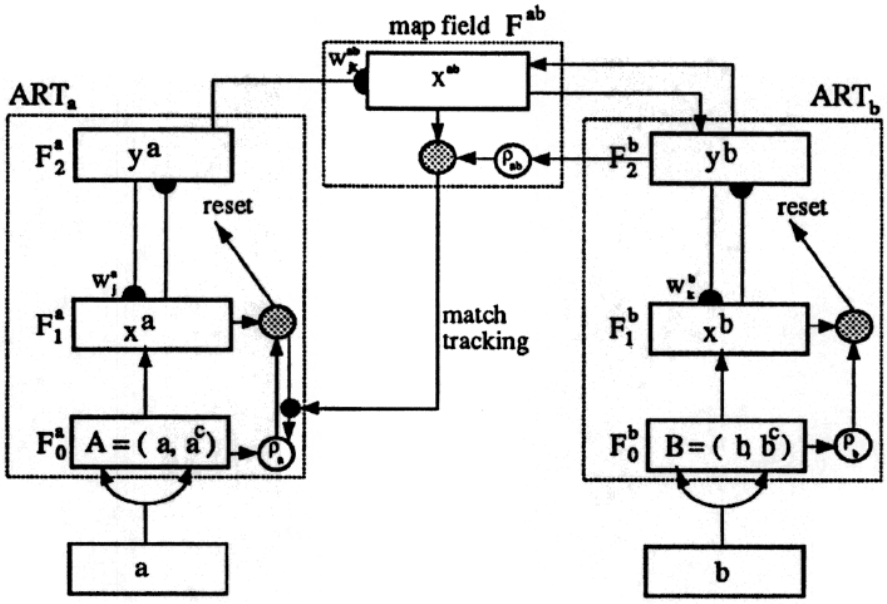


Рис. 4. Архитектура Fuzzy ARTMAP

Во время обучения ART_a получает поток $\{a^{(p)}\}$ входных векторов, а ART_b – поток $\{b^{(p)}\}$, где $b^{(p)}$ – правильное предсказание класса для $a^{(p)}$. Эти модули связаны сетью ассоциативного обучения и внутренним контроллером. Контроллер служит для создания минимального числа ART_a категорий, необходимых для достижения критерия точности. Это достигается минимаксным правилом обучения, которое позволяет ARTMAP системе минимизировать ошибку предсказания и максимизировать обобщение предсказаний.

Fuzzy ARTMAP система включает два Fuzzy ART модуля, ART_a и ART_b, которые соединены вместе посредством меж-ART модуля F^{ab}, называемым *полем преобразования*. Поле преобразования служит для формирования ассоциаций прогнозирования между категориями и для реализации *правила отслеживания совпадений*, согласно которому параметр близости ART_a увеличивается в ответ на ошибку прогнозирования в ART_b. Отслеживание совпадений реорганизует структуру категорий таким образом, что ошибка прогнозирования не повторяется при последующих представлениях входного вектора. Алгоритм работы поля преобразования F^{ab} описывается следующим образом.

ART_a и ART_b. Входные вектора для ART_a и ART_b в комплементарном коде обозначим $I = A = (a, a^c)$ для ART_a; $I = B = (b, b^c)$ для ART_b. Переменные в ART_a и ART_b обозначаются индексами a и b, соответственно. Для ART_a выходной вектор поля F_1^a обозначим $x^a \equiv (x_1^a ... x_{2Ma}^a)$; выходной вектор поля F_2^a обозначим $y^a \equiv (y_1^a ... y_{Na}^a)$; j-ый ве-

совой вектор ART_a обозначим $w_j^a \equiv (w_{j1}^a, w_{j2}^a, ..., w_{j,2Ma}^a)$. Аналогично, для ART_b выходной вектор поля F_1^b обозначим $x^b \equiv (x_1^b ... x_{2Mb}^b)$; выходной вектор поля F_2^b обозначим $y^b \equiv (y_1^b ... y_{Nb}^b)$; k-ый весовой вектор ART_b обозначим $w_k^b \equiv (w_{k1}^b, w_{k2}^b, ..., w_{k,2Mb}^b)$. Для поля преобразования обозначим выходной вектор $x^{ab} \equiv (x_1^{ab} ... x_{Nb}^{ab})$; весовой вектор от j-ого элемента поля F_2^a к полю F^{ab} обозначим $w_j^{ab} \equiv (w_{j1}^{ab}, ..., w_{jNb}^{ab})$. Вектора x^a, y^a, x^b, y^b и x^{ab} устанавливаются в 0 между представления входных векторов.

Активация поля преобразования. Поле преобразования F^{ab} становится активным, когда одна из категорий, ART_a или ART_b, активна. Если выбрана категория J поля F_2^a, тогда ее весовой вектор w_J^{ab} активирует F^{ab}. Если активна категория K поля F_2^b, тогда элемент активируется элементом K в F^{ab} с помощью связей типа один-к-одному между F_2^b и F^{ab}. Если оба модуля, ART_a и ART_b, активны, то F^{ab} активируется только в том случае, если ART_a предсказывает ту же категорию, что и ART_b через весовой вектор w_J^{ab}. Выходной вектор поля F^{ab} x^{ab} равен

$$x^{ab} = \begin{cases} y^b \wedge w_J^{ab}, & \text{если J-ая категория } F_2^a \text{ активна и } F_2^b \text{ активно} \\ w_J^{ab}, & \text{если J-ая категория } F_2^a \text{ активна и } F_2^b \text{ неактивно} \\ y^b, & \text{если } F_2^a \text{ неактивно и } F_2^b \text{ активно} \\ 0, & \text{если } F_2^a \text{ неактивно и } F_2^b \text{ неактивно} \end{cases} . \quad (18)$$

Согласно (18), x^{ab}=0, если предсказание w_J^{ab} не совпадает с y^b. Такое несовпадение активизирует поиск в ART_a более подходящей категории как описано ниже.

Отслеживание совпадений. В начале представления каждого входного вектора параметр близости ρ_a ART_a равен исходному значению $\overline{\rho_a}$. Параметр близости поля преобразования равен ρ_{ab}. Если

$$\left|x^{ab}\right| < \rho_{ab}\left|y^b\right|, \quad (19)$$

тогда ρ_a увеличивается, пока не будет больше $\left|A \wedge w_J^a\right|\left|A\right|^{-1}$, где A – входной вектор F_1^a в комплементарном коде. Тогда

$$\left|x^a\right| = \left|A \wedge w_J^a\right| < \rho_a\left|A\right|, \quad (20)$$

где J – индекс активной категории F_2^a. После увеличения ρ_a происходит поиск другой категории в ART_a, который приводит к активации другой категории J поля F_2^a, удовлетворяющей

$$\left|x^a\right| = \left|A \wedge w_J^a\right| \geq \rho_a\left|A\right| \quad (21)$$

и

$$\left|x^{ab}\right| = \left|y^b \wedge w_J^a\right| \geq \rho_{ab}\left|y^b\right|. \quad (22)$$

Обучение поля преобразования. Правило обучения определяет, каким образом весовые вектора w_{jk}^{ab} поля преобразования изменяются с течением времени. Первоначально весовые вектора w_{jk}^{ab} $F_2^a \rightarrow F^{ab}$ равны

$$w_{jk}^{ab}(0) = 1. \qquad (23)$$

При резонансе при активной категории J модуля ART_a w_J^{ab} приближается к вектору поля преобразования x^{ab}. При быстром обучении, когда J обучается предсказывать категорию K ART_b, эта ассоциация остается постоянной, т.е. $w_{JK}^{ab} = 1$.

Fast Adaptive Neural Network Classifier (FANNC) [7]. FANNC является развитием теории адаптивного резонанса и теории поля. FANNC предназначена для решения классификационных задач. Особенностями этой НС являются однопроходное обучение, высокая точность и быстрая скорость обучения. Процесс обучения НС – инкрементный. При появлении новых данных НС «дообучается» за счет небольшой модификации топологии, т.е. добавлением одного или двух нейронов и связей к уже существующей НС.

Сеть FANNC состоит из четырех слоев нейронов (рис. 5). Функцией активации нейронов скрытых слоев является сигмоидная функция. Гауссовы веса соединяют нейроны входного слоя с нейронами второго слоя. FANNC использует нейроны второго слоя для внутренней классификации входного вектора, а нейроны третьего слоя – для внутренней классификации выходного вектора. Модификация связей между этими слоями осуществляется путем обучения с учителем. За исключением связей между первым и вторым слоем, все связи двунаправленные. Обратные связи передают ответные сигналы для реализации резонанса и соревнования нейронов и всегда равны 1.0.

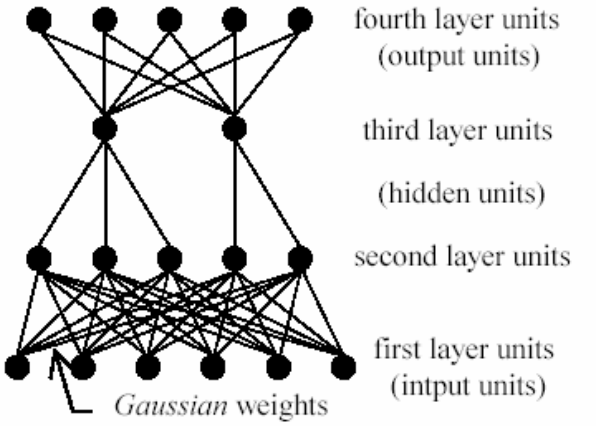


Рис. 5. Архитектура FANNC

В начальный момент времени НС состоит только из входного и выходного слоев. Число элементов в этих слоях равно, соответственно, числу компонентов входного и выходного векторов. Число элементов в скрытых слоях равно нулю. При представлении новых данных FANNC добавляет скрытые нейроны. При представлении первого вектора FANNC добавляет два скрытых нейрона – один во второй слой и один в третий слой. Эти нейроны соединены между собой. Прямая и обратная связь равна 1.0. Нейрон третьего слоя соединен со всеми выходными нейронами, прямые связи равны соответствующим компонентам выходного вектора. Обратные связи равны 1.0. Нейрон второго слоя соединен со всеми нейронами входного слоя с помощью гауссовых весов. Центры чувствительности равны соответствующим значениям компонентов входного вектора, ширины характеристик чувствительности устанавливаются в значение по умолчанию.

Обозначим входной вектор как $A_k = (a_k^1, a_k^2, ..., a_k^n)$ (k=1, 2, …, m), где k – индекс входного вектора, n – число компонентов входного вектора. Входное значение для нейрона j второго слоя от нейрона i первого слоя равно

$$bIn_{ij} = e^{-(\frac{a_i^k - \theta_{ij}}{\alpha_{ij}})^2},$$

где θ_{ij} и α_{ij} – соответственно, центр чувствительности и ширина характеристики чувствительности гауссова веса, соединяющего нейрон *i* с нейроном *j*. Так как динамические свойства гауссова веса полностью определяются центром чувствительности и шириной характеристики чувствительности, то знания, получаемые во время обучения, могут быть запомнены путем модификации этих двух параметров.

Алгоритм обучения сети FANNC представлен на рис. 6 [7].

Заключение

Рассмотрев существующие подходы к классификации систем обнаружения атак и общие принципы мониторинга безопасности с помощью нейронных сетей, комплекс задач дальнейшего исследования можно определить следующим образом.

1. В связи с многообразием нейросетевых алгоритмов необходимо провести их исследование применительно к задаче анализа данных аудита КС с целью обнаружения аномального поведения пользователей и выбрать наиболее подходящий нейросетевой алгоритм.

2. Необходимо определить совокупность контролируемых событий аудита, которые характеризовали бы поведение пользователя.

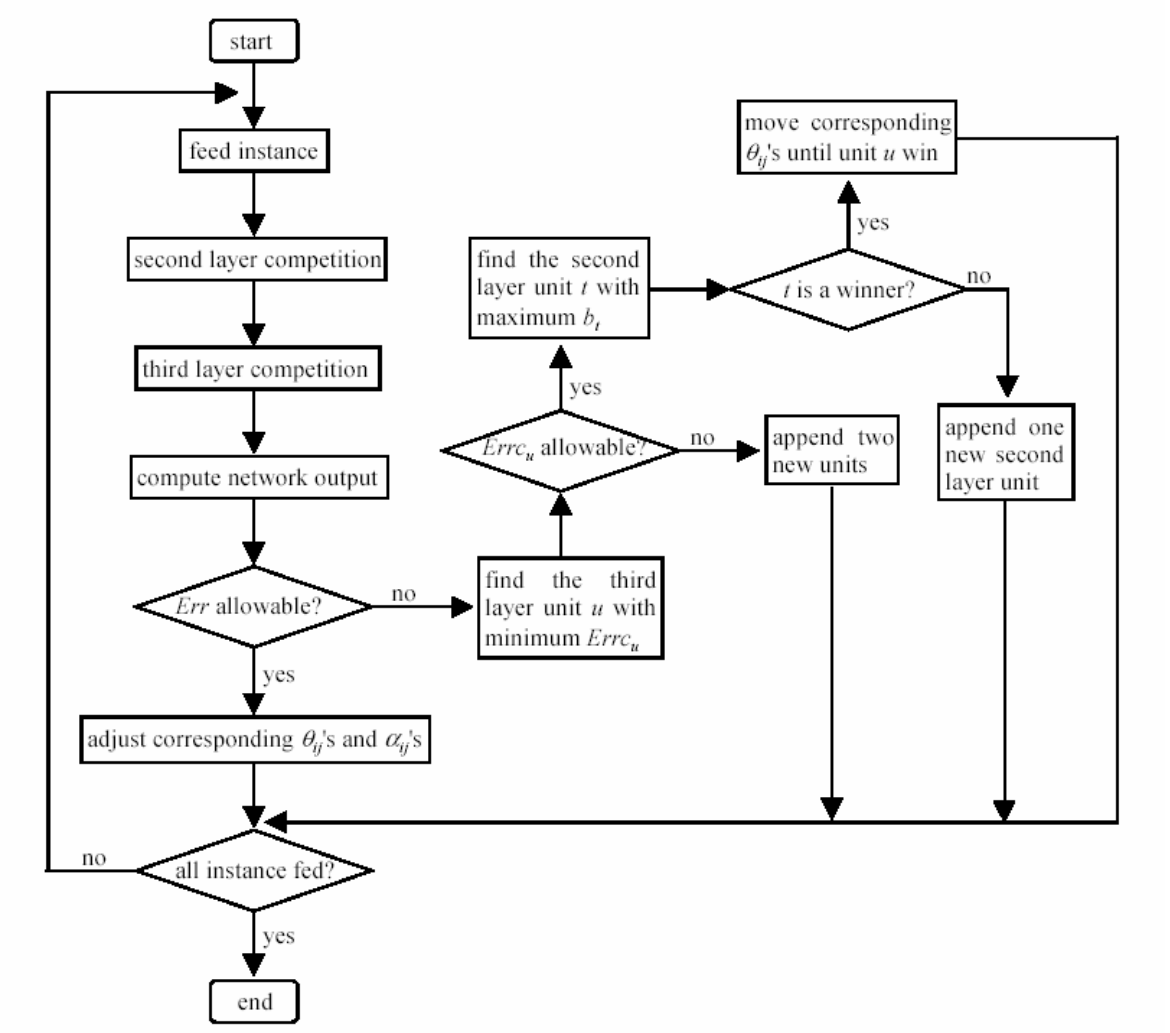


Рис. 6. Алгоритм обучения сети FANNC

3. Необходимо определить способ преобразования данных аудита во входной вектор нейронной сети и, при необходимости, способ нормализации входных векторов для соответствия требованиям к входным векторам выбранной топологии НС.

Литература

1. Дюк. В., Самойленко А. Data Mining: учебный курс (+CD). СПб: Питер, 2001.
2. Лукацкий А.В. Обнаружение атак. 2-е изд., перераб. и доп. СПб.: БХВ-Петербург, 2003.
3. Уоссермен Ф. Нейрокомпьютерная техника: Теория и практика. М.: Мир, 1992.
4. Carpenter G.A., Grossberg S. A self-organizing neural network for supervised learning, recognition, and prediction. // IEEE Communications Magazine. 1992. 30. P. 38–49.
5. Carpenter G.A., Grossberg S., Rosen D.B. Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system. // Neural Networks. 1991. 4. P. 759–771.
6. Carpenter G.A., Grossberg S., Markuzon N., Reynolds J.H., Rosen D.B. Fuzzy ARTMAP: An adaptive resonance architecture for incremental learning of analog maps. // Proc. of the International Joint Conference on Neural Network. 1992.
7. Zhou Z., Chen S., Chen Z. FANNC: A Fast Adaptive Neural Network Classifier. // Knowledge and Information Systems. 2000. 2. P. 115–129.

АНАЛИЗ И ПРОГНОЗИРОВАНИЕ ДИНАМИКИ УЯЗВИМОСТЕЙ

Е.О. Калашник
Научный руководитель – к.т.н., доцент А.В. Любимов

Проблема уязвимостей объектов информатизации и их обнаружения исследуется очень давно, и за время ее существования предпринимались различные попытки анализа и прогнозирования, но практически все они носят аналитический характер. Целью данной работы является реализация математических методов обработки временных рядов на основе американской национальной базы данных уязвимостей (National Vulnerability Database, NVD) для анализа и прогнозирования динамики уязвимостей объектов информатизации.

Состояние проблемы

Информатизация любой организационной системы несет с собой не только известные преимущества, которые стимулируют весь процесс информатизации, но и появление новых угроз, связанных с использованием информационных автоматизированных систем. Однако угрозы сами по себе не проявляются, все угрозы могут быть реализованы при наличии активных субъектов – источников угроз»и слабых мест – уязвимостей, присущих отдельным компонентам информационных автоматизированных систем – защищаемым объектам [1].

В связи с этим необходимо решить ряд задач, одной из которых является анализ защищенности отдельных компонентов информационных автоматизированных систем. Устранение обнаруженных уязвимостей нейтрализует связанные с ними угрозы до того времени, пока не появятся благоприятные условия – новые уязвимости. Таким образом, возникает необходимость анализа и прогнозирования динамики уязвимостей объектов информатизации.

Проблема уязвимостей изучается давно и открыто. Разработаны методики и подходы к анализу защищенности. Описано использование методов теории вероятности и математической статистики для оценки вероятностей обнаружения уязвимостей в информационных автоматизированных системах [2]. Изданы серии книг, посвященных системам обнаружения атак [3].

По мнению одних авторов, статистика показывает, что количество уязвимостей растет год от года. С одной стороны, это связано с тем, что год от года растет количество ПО, а с другой – с тем, что сейчас уязвимости ищутся намеренно как хакерами, так и компаниями-производителями ПО и ОС [4]. По мнению других, кризис на ИТ-рынке, вызванный ростом взлома компьютерных систем и отдельного программного обеспечения (а как следствие – экономический ущерб от невнимания к вопросам информационной безопасности рос космическими темпами), привел к тому, что заказчики перестали постоянно «обновлять» ПО, поскольку, кроме косметических новшеств, в продуктах не было реальных улучшений. Сокращение заказов вынудило разработчиков переориентироваться с косметической на более детальную доработку своих продуктов. Следовательно, говорят они, число уязвимостей в ПО резко сокращается [5].

Исходя из этого, предлагается произвести самостоятельный математический расчет динамики уязвимостей.

Объект исследования

В качестве объекта исследования использована база данных уязвимостей NVD (проект Американского национального института стандартов и технологий – National Institute of Standards and Technology, NIST)). Это обширная база данных уязвимостей информационной безопасности, которая объединяет все доступные американскому правительству ресурсы по уязвимостям и предоставляет ссылки на отраслевые ресурсы.

NVD является единственной базой данных, основанной на стандарте CVE и полностью с ним согласованной. На сайте NVD предоставляются следующие бесплатные и не требующие регистрации услуги: поисковая машина с высокой степенью детализации; XML выгрузки; служба статистики [6].

Однако сайт NVD имеет недостатки: невозможен просмотр самой базы в целом, служба статистики не позволяет делать статистику по конкретным датам появления уязвимостей, а, следовательно, невозможно произвести сравнения и дать прогноз. Эти недостатки послужили предпосылкой для создания собственной клиент-серверной базы данных на основе XML выгрузок NVD и построения собственного метода анализа и прогнозирования уязвимостей содержащихся в NVD.

Данная работа поможет специалистам в области информационной безопасности делать запросы по своим программным продуктам или техническим средствам, смотреть статистику уязвимостей по времени для групп или классов объектов информатизации, а также самостоятельно делать прогнозы и принимать соответствующие меры для предотвращения угроз своей информационной безопасности.

Описание проделанной работы

На сервер выкладывается база данных NVD, к нему пишется клиентское приложение, которое позволяет не только просматривать саму базу, но и выдавать статистические данные по таким параметрам, как, например: разработчик, продукт, версия, категория уязвимости, тип воздействия и т.д. Полученные результаты анализируются, строятся графики, а уже на основе самих графиков выбирается и применяются конкретные математические методы обработки временных рядов, что позволяет строить прогноз относительно динамики появления новых уязвимостей.

Хотелось бы отметить, что для прогнозирования динамики уязвимостей одной базы данных NVD недостаточно, так как при исследовании графиков возникает ряд вопросов относительно динамики роста самих объектов информатизации.

Применяя данный метод на практике, можно сделать прогноз динамики уязвимостей с целью предотвращения различного рода угроз информационной безопасности, что актуально на сегодняшний день, учитывая рост объектов информатизации.

Литература

1. Лукацкий А.В. Выявление уязвимостей компьютерных сетей. (Источник: http://www.citforum.ru/security/internet/vulnerability/).
2. Кучер В.А., Агранович В.С. Использование методов теории вероятностей и математической статистики для оценки вероятностей обнаружения уязвимостей в информационных автоматизированных системах (Источник: www.contrterror.tsure.ru/site/magazine5/pdf/06-38-Kucher-Agranovich.pdf).
3. Бейс Ребекка. ICSA. Введение в обнаружение атак и анализ защищенности (Источник: http://bugtraq.ru/library/books/icsa/index.html).
4. Щеглов А. Уязвимости современных ОС (Источник: http://www.cnews.ru/reviews/free/security2005/articles/vulnerability.shtml).
5. Боровко Р. / CNews Analytics. Число уязвимостей в ПО резко сокращается (Источник: http://www.cnews.ru/reviews/free/security2004/globalthreats/index2.shtml).
6. Официальный сайт NVD: http://nvd.nist.gov/

ИССЛЕДОВАНИЕ СОСТОЯНИЯ БЕЗОПАСНОСТИ ВЕБ-САЙТОВ БАНКОВСКОЙ СФЕРЫ СЕВЕРО-ЗАПАДНОГО ФЕДЕРАЛЬНОГО ОКРУГА НА ОСНОВЕ «ЛЕГАЛЬНЫХ» МЕТОДОВ АНАЛИЗА ЗАЩИЩЕННОСТИ

А.В. Захаров
Научный руководитель – д.т.н., профессор Л.Г. Осовецкий

Статья описывает результаты разработки методики «легальных» методов анализа защищенности веб-сайтов банковской сферы и апробации методики на веб-сайтах банков Северо-Западного федерального округа без нарушения прав владельцев. По результатам исследования разработаны рекомендации, предназначенные для банков, имеющих собственные веб-сайты или планирующих их создание. Данные рекомендации получены из анализа изъянов в безопасности, имеющих место у некоторых рассмотренных в исследовании банков, а также путем обобщения лучшей практики обеспечения информационной безопасности веб-сайтов и направлены на усиление безопасности.

Введение

Актуальность предлагаемого исследования обусловлена недостаточной изученностью российского сегмента сети Интернет в целом и состояния его региональных фрагментов в аспекте информационной безопасности. В последние 2–3 года и в России, и особенно за рубежом наблюдается быстрый рост числа Интернет-услуг, предоставляемых банками. Как следствие, незамедлительно появились новые виды угроз, и значительно увеличилось количество реализаций уже известных угроз, направленных на кражу информации о кредитных картах и счете клиента. Новостные ленты, посвященные компьютерной безопасности, постоянно извещают о новых инцидентах, связанных с онлайновыми аферами, приведшими к финансовому ущербу. Самой опасной угрозой последних лет, появившейся на фоне роста объема выполняемых через Интернет платежей, стал «фишинг» (англ. phishing – произносится как английское слово, означающее «рыбную ловлю»). По данным Gartner Group только за 2006 год общий ущерб от фишинга в США составил порядка 2 миллиардов долларов США.

Таким образом, можно утверждать, что именно банковско-финансовая сфера, представляющая возможности финансового обогащения, становится основным объектом атак интернет-злоумышленников. Современный веб-сайт представляет собой часть корпоративной инфраструктуры. Взлом корпоративного сайта – это удар по репутации и имиджу компании. Очень неприятным в подобных событиях становится огласка происшествия. Но потеря данных с сайта, информации о клиентах – это уже прямые убытки. И огласка таких происшествий происходит далеко не всегда. Чем серьезнее компания и известнее ее имя и продукты, тем существеннее бывают риски и убытки от взлома корпоративного сайта. В этой связи вопросы оценки состояния безопасности и степени уязвимости банковской сферы в Санкт-Петербурге и регионе представляют актуальную задачу.

Разработка методики проведения анализа защищенности веб-сайтов

Первый из рассматриваемых методов заключается в получении общедоступной информации о технической поддержке веб-сайта. Как известно, регистрация IP-адресов и доменных имен в Глобальной сети осуществляется централизованно на международном уровне. Информация о зарегистрированных адресах и их владельцах является общедоступной. В России эти сведения можно получить, в частности, на сайте Российского НИИ развития сетей. Данный сетевой сервис, известный под названием WHOIS, позволяет собрать детальную информацию о сетевой инфраструктуре банка (компании).

Также полезным инструментом для получения информации является сведения, предоставляемые компанией NetCraft, занимающейся мониторингом информационной безопасности в Интернете. Особое внимание уделяется компаниям из сферы электронной коммерции. Если веб-сайт, указанный в запросе пользователя на выдачу информации, отсутствует в базе веб-сайтов компании Netcraft, он будет добавлен в список веб-сайтов, мониторинг безопасности которых осуществляет служба Netcraft.

В целом в аспекте «пассивного сбора информации» весь Интернет можно представить одним распределенным инструментом для сбора информации – в данном случае информации, раскрытие которой может оказаться критическим для безопасности системы.

Информация, которая может быть получена подобным образом, включает:

- имя домена или доменов банка;
- адреса подсетей, которыми владеет банк;
- точные адреса узлов, находящихся на периметре сети банка;
- сервисы, запущенные на определенных выше узлах;
- типы операционных систем, запущенных на определенных выше узлах;
- роли узлов, находящихся на периметре;
- механизмы сетевой безопасности, используемые компанией-жертвой (межсетевые экраны и его правила, фильтрующие маршрутизаторы и списки контроля доступа, системы обнаружения атак и т.д.);
- информация о пользователях и группах;
- дополнительная информация (сведения о SNMP, таблицы маршрутизации и т.п.).

Второй метод включает в себя анализ веб-сайта исследуемого банка на предмет информации, которая может помочь злоумышленнику. Так как каждая организация по-своему реализует сайты – со своими разделами и подразделами, то существует несколько наиболее типичных ошибок, которые облегчают злоумышленникам сбор информации о своей жертве и, в то же время, позволяют провести такой сбор информации совершенно легально.

На сервере практически любой компании (в том числе и банка) существует раздел «О компании», в котором рассказывается об истории создания компании, вехах в ее развитии, руководстве и т.д. Нередко можно встретить раздел, посвященный корпоративной политике, который облегчает хакерам реализацию атак social engineering. Также в данном разделе могут быть указаны рабочие часы, которые позволяют злоумышленнику реализовывать свои несанкционированные действия в нерабочие часы без боязни быть обнаруженными.

Наименее целесообразным является размещение на сайте подробного описания системы защиты корпоративной сети. В исходном коде страниц Web-сайта также может содержаться важная, а зачастую конфиденциальная информация. Например, на некоторых сайтах, построенных на основе программного обеспечения MS Internet Information Server и интегрированных с базой данных, в ASP-страницах могут содержаться пароли на доступ к БД или иным важным ресурсам.

Третий метод включает анализ:

1. наличия на веб-сайте механизмов идентификации и аутентификации пользователей для доступа к определенным веб-страницам данного сайта (FIA_UID и FIA_UAU);
2. использования протоколов SSL или TLS для защиты протокола HTTP, реализующего функциональные требования безопасности FIA_UID, FIA_UAU, FDP_UCT и FDP_UIT.

Суть метода заключается в использовании специального вида запросов к поисковым интернет-сервисам для получения конфиденциальной информации. Одним из средств автоматизированного анализа с применением данной технологии является про-

граммный комплекс Foundstone SiteDigger (дословно «сайтокопатель»). В комплексе реализованы методы поиска разнообразной конфиденциальной информации на веб-сайте, а также критичной информации о веб-сервере, оставленной администратором веб-сайта по недосмотру или вследствие ошибок конфигурирования. В основе работы комплекса лежит использование API-функций крупнейшей поисковой системы в мире – Google. Методы сбора информации об уязвимостях веб-сайтов с использованием поисковика Google в западной литературу даже получило специальное название – Google Hacks.

Методика анализа защищенности веб-сайтов

Таким образом, сформулируем общую методику проведения исследования. Данная методика позволяет осуществить моделирование типовых угроз для веб-сайтов и включает следующие шаги.

- На первом этапе осуществляется сбор информации об исследуемом объекте с использованием общедоступной информации о сетевой инфраструктуре объекта анализа – трассировку маршрутов к веб-сайту, анализ информации, выдаваемой службой whois, анализ программного обеспечения веб-серверов с использованием службы NetCraft и т.д.
- Анализ размещенной на веб-сайте информации позволяет выявить сведения, которые станут для злоумышленника основой для дальнейших действий. Выявление сведений о сотрудниках – для использования методов социальной психологии, номеров телефонов компании – для поиска модемов, изучение вакансий – для сбора информации об используемом в организации программном обеспечении (ПО) и т.д.
- Анализ утечек конфиденциальной информации с использованием программного комплекса Foundstone SiteDigger.
- Анализ защищенности информационного обмена и доступа к конфиденциальной информации на веб-сайте с использованием программного инструментально-моделирующего комплекса анализа защищенности, разработанного в диссертационном исследовании автора «Модель и методы мониторинга и оценки защищенности веб-сайтов сети Интернет».

На основе собранной информации проводится анализ и оценка текущего состояния защищенности веб-сайтов банковской сферы СЗФО. По результатам исследования приводятся рекомендации по устранению уязвимостей и повышению уровня защищенности.

Анализ общедоступной информации легальными методами

С использованием службы NetCraft была собрана статистика по программному обеспечению веб-сайта банков СЗФО. Собранные данные, включающие программное обеспечение веб-сервера и операционную систему, представлены в табл. 1, 2. С использованием программного комплекса анализа защищенности хостов в Интернете были получены результаты применения комплекса средств защиты информационного обмена между веб-сайтов банка и его посетителями.

Программное обеспечение веб-сервера	Доля (%)
Apache	79
Microsoft IIS	16
Другие	5

Таблица 1. Программное обеспечение веб-серверов

Операционная система на веб-сервере	Доля (%)
FreeBSD	62
Linux/Unix-based	19
Windows	17
Другие	2

Таблица 2. Операционные системы, применяемые на веб-серверах банковской сферы СЗФО

Анализ результатов свидетельствует о недостаточной распространенности использования протокола SSL/TLS для безопасного доступа к веб-сайту – поддержку этого протокола осуществляет только 38,3 % рассмотренных веб-сайтов (рис. 1). При этом отметим, что в данном испытании исследовался только лишь возможность доступа ко всему веб-сайту по защищенным протоколам. Безопасность доступа к услугам дистанционного банковского обслуживания исследована отдельно в следующем подразделе работы.

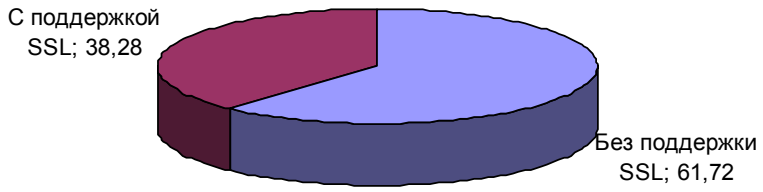


Рис. 1. Доли веб-сайтов (в %) с поддержкой и без поддержки протоколов SSL/TLS

Анализ сферы предоставления услуг дистанционного банковского обслуживания через Интернет

В ряде случаев исследованию мешал факт сокрытия на веб-сайте подробной информации о предоставляемых сервисах. В этих случаях более детальную информацию предлагалось получить у менеджеров банка при личном посещении. Отметим, что подобные меры приводят к неудобству для клиентов, однако с точки зрения безопасности их можно назвать оправданными – применение различных ограничений на распространение информации о системах защиты способствует укреплению безопасности этих сервисов. Логичным является выбор банков о предоставлении детальной информации не всем (как в данном случае – посетителям веб-сайта банка), а только его клиентам.

Безопасность услуги дистанционного обслуживания клиентов через Интернет рассмотрим на примере типового механизма предоставления подобной услуги.

В 60 % рассмотренных процедур подключения услуги дистанционного банковского обслуживания требуется оформить договор и получить необходимые реквизиты в офисе банка. Однако личное присутствие требуется не всегда. Например, в ряде банков можно просто позвонить по указанному телефону, сообщить оператору свое имя, номер карты, кодовое слово и номер мобильного телефона. Оператор сообщит логин и вышлет SMS с временным паролем, который Вам необходимо будет сменить при первом входе в систему «Интернет-Банк».

Отметим также, что услуга «Интернет-банк» клиентам предоставляется как бесплатно, так и за дополнительную плату. Какой-либо корреляции между требованием оплаты услуги и предоставлением повышенного уровня безопасности мы не отметили.

Как правило, информация о предоставлении данных услуг размещается на главной странице. В единичных случаях нам пришлось внимательно изучить веб-сайт для обнаружения подобной информации.

По результатам исследования можно отметить, что на данный момент нет общего устоявшегося и закрепившегося названия для услуг, оказываемых банком дистанционно через сеть Интернет. На наш взгляд, этот факт свидетельствует о неразвитости данной услуги, о том, что предложение такого рода услуг находится на этапе становления в СЗФО. Так, в процессе исследования нам встречались следующие названия.

1. Интернет-банкинг.
2. Банк ON-LINE.
3. Интернет-банк.
4. iBank.
5. Телебанк.
6. Интернет-клиент.
7. Онлайн-банкинг.
8. Электронная коммерция.
9. Интернет-сервис БАНК.
10. Электронный банкинг.
11. Система дистанционного обслуживания Enter.IMB™.
12. Электронное банковское обслуживание.

Перейдем к анализу механизмов реализации услуги дистанционного банковского обслуживания и мер по их защите. В табл. 3 представлены средства безопасности, применяемые для защиты услуги дистанционного обслуживания клиентов через сеть Интернет.

Мера безопасности	**Количество применений**
SSL/TLS	115
USB Token/Touch Memory/Ключевая дискета	11
Специализированное программное обеспечение	52
JAVA-клиенты	12

Таблица 3. Механизмы обеспечения безопасности услуг вида «Банк-Клиент»

Отметим, что в ряде случаев веб-сайтом применяется комбинирование разных механизмов защиты, поэтому сумма применяемых механизмов защиты в таблице больше количества рассмотренных веб-сайтов. Как видно из таблицы, наиболее распространенным механизмом обеспечения безопасности информационного обмена является применение одного из протоколов TLS/SSL. Описание этого протокола уже приведено в первом разделе работы. Подавляющее большинство сертификатов банковских сайтов в случае применения протокола SSL выданы организацией Thawte.

Также популярным средством защиты является использование специального программного обеспечения при предоставлении услуги дистанционного клиентского обслуживания. В этом случае программа представляет собой приложение под определенную операционную систему (мы обнаружили только программное обеспечение под платформу Windows). Это приложение необходимо скачать и установить на пользовательский компьютер. Использование специализированного программного обеспечения повышает уровень безопасности информационного обмена, однако сужает универсальность использования услуги – доступ к ней возможен только с компьютера пользователя, а не из компьютерного клуба, интернет-кафе и т.д.

Следующей по популярности системой добавочной защиты является использование JAVA-клиентов. Мы специально выделили данный вид программной защиты в отдельный класс, так как JAVA-апплеты более универсальны вследствие кроссплатформенности языка JAVA. Применение JAVA-апплета для доступа к услугам вида

«банк-клиент» позволяет воспользоваться услугой с любого компьютера, имеющего интернет-браузер с поддержкой JAVA.

Для обеспечения высокого уровня безопасности ведущие банки предоставляют добавочный уровень защиты в виде использования ключевой дискеты, элемента Touch memory или USB-token. Touch Memory – это миниатюрное электронное устройство, выполненное на базе специальной микросхемы с автономным литиевым источником питания. Оно имеет энергонезависимую память и уникальный идентификационный номер. Герметичный стальной корпус, выполненный в виде таблетки диаметром несколько миллиметров, предохраняет устройство от вредных атмосферных воздействий, а также механических повреждений и электромагнитных полей. Touch Memory предназначен для осуществления защищенного доступа к специальным серверам банка и для заверения электронных документов, направляемых пользователем в банк, электронной подписью.

USB ключи представляют собой микроэлектронные устройства, позволяющие безопасно хранить разнообразную информацию, в том числе электронные сертификаты, персональные ключи и т.д. и подключаются к компьютеру через USB-порт. USB-ключ предназначен для осуществления защищенного доступа к специальным серверам банка и для заверения электронных документов, направляемых пользователем в банк, электронной подписью.

В банке осуществляется проверка электронных цифровых подписей на каждом документе, принятом от клиента. Встроенные средства шифрования обеспечивают передачу информации между клиентом и банком в зашифрованном виде. При этом используются сертифицированные криптографические средства защиты информации.

Средства информационной безопасности системы вида «банк-клиент» при корректном использовании в сочетании с необходимой организационной и правовой поддержкой гарантируют надежную защиту системы от несанкционированного доступа и модификации информации независимо от используемых для ее передачи каналов связи. Для обеспечения информационной безопасности в системе банк-клиент применяются следующие механизмы:

- разграничение прав подписи документов на клиентском месте;
- конфиденциальность доступа на веб-сервер банка, обеспечивающая гарантированную защиту банковской тайны;
- сертифицированная защита «активных элементов» системы на клиентском месте;
- сертифицированные средства криптозащиты информации, включая шифрование передаваемых данных и электронную цифровую подпись;
- защита паролем электронной цифровой подписи.

Регион СЗФО	**% доля банков, предоставляющих услуги дистанционного банковского обслуживания**
Санкт-Петербург	78,5 %
Мурманская область	100 %
Вологодская область	100 %
Новгородская область	33,3 %
Республика Коми	100 %
Псковская область	0 %
Калининградская область	100 %

Таблица 4. Распространенность услуги дистанционного банковского обслуживания по регионам СЗФО

Защита от фишинга

В данном случае об угрозе фишинга клиентов стали предупреждать уже после инцидента. Характерная ситуация – когда меры защиты принимаются не заблаговременно, а только после атаки. Тем не менее, на сайте двух банков имеются предупреждения для клиентов, которые должна снизить успешность фишинг-атак злоумышленников. Типовые рекомендации выглядят следующим образом.

- Банк **НИКОГДА НЕ ЗАПРАШИВАЕТ** Вашу персональную информацию по электронной почте и не просит Вас зайти в систему интернет-банкинга **по ссылке в письме**, так как это противоречит соображениям безопасности.
- Пожалуйста, обращайте внимание на сообщения о подозрительных операциях. Так, если Вам сообщают, что на Ваш счет поступили деньги, которых Вы не ожидали, этого достаточно, чтобы заподозрить попытку мошенничества.
- Самый надежный и безопасный способ воспользоваться нашей системой интернет-банкинга – набрать адрес сайта **WWW.***.RU** в адресной строке Вашего браузера (чаще всего это MS Internet Explorer) и пройти в систему по ссылке, которая есть на всех страницах сайта.
- Если Вы все же успели стать жертвой мошенников или подозреваете, что Ваши данные попали в чужие руки, рекомендуем Вам незамедлительно сообщить об этом в банк по телефонам…
- Даже в случае попадания Ваших данных к мошенникам, Вам совсем не обязательно закрывать счет в банке или блокировать банковскую карту. Достаточно связаться с круглосуточной службой поддержки банка по телефону …, и наши специалисты помогут Вам решить возникнувшую проблему.
- Не отвечайте на полученные по электронной почте подозрительные сообщения от банка, не связавшись предварительно с банком по известным Вам телефонам.
- Не сообщайте посторонним пароли системы "Интернет Банк-Клиент", PIN-код карты, номер карты, дату истечения срока действия карты, код безопасности карты (три цифры, напечатанные на полосе для подписи после номера карты), размер кредитной линии по карте, информацию об операциях по карте, персональную информацию, информацию о паролях и данных документов. Помните, что ответственным сотрудникам банка известна данная информация (кроме PIN-кода карты и пароля для входа в систему "Интернет Банк-Клиент"), и у них нет необходимости запрашивать ее у клиента. В случае, если у Вас возникли сомнения, попросите у сотрудника назвать свою фамилию и имя и свяжитесь с ним самостоятельно через справочную службу банка.
- Для приобретения товаров через Интернет, по почте или телефону пользуйтесь услугами известных и надежных компаний и магазинов.

Заключение

Сегодня в России около 15 млн. интернет-пользователей [10]. По оценкам специалистов, ориентирующихся в своих прогнозах на опыт ведущих зарубежных стран, атаки в киберпространстве в России, будут представлять угрозу национальной безопасности, когда это число возрастет примерно в 5 раз, хотя и в нынешней ситуации возникает опасность серьезного экономического ущерба. В первую очередь, учитывая глобальную тенденцию криминализации Интернета [11, 12], эти атаки будут направлены на те сферы информационной инфраструктуры, которые могут принести коммерческую выгоду злоумышленникам. В этой связи наиболее целесообразным в настоящее время является анализ и применение превентивных мер защиты, учитывающих текущую ситуацию и прогнозирование поля угроз.

В данной работе проведено одно из первых публичных исследований защищенности веб-сайтов одной из наиболее привлекательных для злоумышленников – банковской сферы Северо-Западного Федерального округа. Результаты исследования представляют оценку текущего состояния информационной безопасности региона в области защиты банками веб-ресурсов от внешних злоумышленников, состояние развития и сферы услуг, предоставляемых петербургскими банками и банками региона через Интернет, анализ применяемых средств и технологий защиты.

Методика исследования существенно отличается от распространенных методик, используемых в настоящее время, легальностью используемых методов анализа. Легальные методы исследования, разработанные, обобщенные и усовершенствованные в работе, находятся в рамках российского законодательства, так как не приводят к вторжению/нарушению работы сетей и отдельных ЭВМ. Все действия, выполненные в процессе анализа, лежат в рамках действий легальных пользователей веб-сайта. Целью методов является поиск таких изъянов или недостатков в защите веб-сайтов, которые позволяют злоумышленникам преодолеть систему защиты или получить критически важную информацию для последующего вторжения.

По результатам исследования разработаны рекомендации, предназначенные для банков, имеющих собственные веб-сайты или планирующих их создание. Данные рекомендации получены из анализа изъянов в безопасности, имеющих место у некоторых рассмотренных банков в исследовании, а также путем обобщения лучшей практики обеспечения информационной безопасности веб-сайтов и направлены на усиление безопасности банковских веб-сайтов. Важным аспектом является стратегическая направленность некоторых рекомендаций, которая позволяет уже сейчас подготовиться к защите от очень вероятных будущих угроз.

Литература

1. Вирусная энциклопедия Лаборатории Касперского. Американцы потеряли 1 миллиард долларов из-за фишинга. Internet URL: http://www.viruslist.ru
2. Биячуев Т.А. Модель и методы мониторинга и оценки защищенности веб-сайтов сети Интернет: Дис. на соиск. к.т.н.: 05.13.19. СПб, 2005. 208 с.
3. Леонов А. Интернет-разведка глазами хакера. Internet URL http://www.it2b.ru/it2b2.view2.page57.html
4. Ollmann G. Passive Information Gathering. The Analysis of Leaked Network Security Information», 2004. 42 p. Internet URL: http://www.technicalinfo.net/papers/PassiveInfoPart1.html
5. Сбор критических данных без вторжения. Internet URL: http://www.bugtraq.ru/library/security/019.html
6. Tara Calishaen, Rael Dornfest Goolge Hacks.100 Industrial Strength Tips and Tools, O'Reilly, 287 p.
7. Желтые страницы Санкт-Петербург 2006. СПб: РУЗко, 2006. 1575 с.
8. Банки Северо-Запада России 2005. Телефонный справочник. СПб: Изд-во Ассоциации банков Северо-Запада России, 2005. 235 с.
9. Ушаков К. Особенности виртуальной рыбалки. Москва, CIO World №3, 20 марта 2006 г. – С. 51–54.
10. Фонд общественного мнения. Опросы «Интернет в России / Россия в Интернете». Выпуск 13. Осень 2005. Internet URL: http://bd.fom.ru/report/map/d051060
11. Лукацкий А. Криминализация Интернета и защита информационных систем. // Промышленные ведомости. 2005. №10. С. 32–34.
12. Касперский Е. Прогнозы изменений в антивирусной индустрии. Internet URL: http://www.kaspersky.ru/threats?chapter=188269734

2 ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

АЛГОРИТМ ДИФФЕРЕНЦИАЛЬНОЙ КОДОВО-ИМПУЛЬСНОЙ МОДУЛЯЦИИ В ЗАДАЧАХ КОМПРЕССИИ ЦИФРОВОГО ПОТОКА ДАННЫХ, ОПИСЫВАЮЩИХ ДВИЖЕНИЕ 3D-МОДЕЛИ

Е.П. Мироненко (Санкт-Петербургский государственный университет аэрокосмического приборостроения)
Научный руководитель – д.т.н., профессор Н.Н. Красильников (Санкт-Петербургский государственный университет аэрокосмического приборостроения)

Цель работы заключалась в экспериментальном исследовании величины компрессии данных, описывающих движение 3D модели человеческой головы в пространстве, при использовании в качестве алгоритма компрессии дифференциальной кодово-импульсной модуляции (ДКИМ).

Введение

Наиболее распространенным способом представления трехмерного объекта в компьютере является полигональная сетка [1]. Полигональные модели просто создавать и легко анимировать. Анимация моделей осуществляется в покадровом режиме, при котором форма и положение модели в каждом кадре описываются координатами вершин полигональной сетки.

Для передачи движения полигональной 3D-модели, состоящей из набора вершин, на сторону приемника необходимо с заранее установленной частотой кадров передавать пространственные координаты X,Y и Z каждой вершины в новом кадре [2]. В случае глобального движения всех вершин модели одновременно (в случаях поворота или перемещения модели) вместо передачи координат всех вершин по отдельности достаточно передавать углы поворота модели и величину смещения ее центра вдоль осей координат. Однако в случае локального движения отдельных вершин относительно центра модели приходится передавать полностью координаты нового местоположения каждой сдвинувшейся вершины. Наличие подобных движений наиболее характерно при моделировании тела и головы человека, в частности, при описании мимики лица. В большинстве современных форматов хранения и передачи информации о 3D-моделях подобное движение передается в покадровом режиме, т.е. передаются (сохраняются) координаты всех вершин модели в каждом новом кадре. К таким форматам относятся:

1) формат OBJ – разработан фирмой Alias;
2) формат VRML – предназначен для передачи данных о 3D-объектах в сетях.

Кодирование движения 3D-модели по методу ДКИМ

Метод дифференциальной кодово-импульсной модуляции (ДКИМ) [3] предполагает передачу не самих новых координат вершин, а передачу разности между их положением в предыдущем и последующем кадрах. Уменьшение размера посылки в этом случае достигается за счет того, что локальное движение имеет место лишь для части вершин, большинство же из них в случае моделирования мимики лица остаются неподвижными относительно центра головы. Процесс передачи (сохранения) движения модели в кадрах видеопоследовательности можно реализовать двумя способами:

1) передача данных о перемещении только тех вершин, которые изменили свое положение в новом кадре;

2) передача данных о перемещении всех вершин в новом кадре.

Рассмотрим первый метод передачи. В этом случае требуется передать только информацию о движении тех вершин, которые изменили свое положение относительно предыдущего кадра. Передаются данные о перемещении вершины вдоль осей X, Y, Z. Если смещение вдоль оси не происходило, то информация об этом не передается. Но наличия данных только о величине перемещения недостаточно для восстановления закодированного движения. Для правильного декодирования необходимо для каждой передвинувшейся в новом кадре вершины передавать ее номер и названия (номера) осей, движение вдоль которых описывают передаваемые числовые значения сдвига. Таким образом, посылка о перемещении одной вершины (в случае, если она переместилась вдоль всех осей) в данном методе имеет вид, показанный в табл. 1:

Метка номера вершины	Номер вершины	Метка оси X	Величина смещения вдоль оси X	Метка оси Y	Величина смещения вдоль оси Y	Метка оси Z	Величина смещения вдоль оси Z

Таблица 1. Последовательность данных при использовании первого метода передачи

Рассмотрим второй метод передачи. В этом методе последовательно передаются (сохраняются) данные о перемещении всех вершин вдоль всех осей в новом кадре относительно предыдущего. Передача данных осуществляется в фиксированном порядке как для номеров вершин (по возрастанию), так и для осей (Первыми передаются данные о перемещении вдоль оси X, затем вдоль Y и затем вдоль Z). Если вершина не перемещалась вдоль какой-либо из осей, то в этом случае вместо значения величины перемещения передается ноль. Если вершина не перемещалась совсем, то передаются три нуля (по одному для каждой оси). Для передачи величины смещения вершины вдоль одной оси используется фиксированное количество байт, поэтому в данном методе не требуется передавать дополнительную информацию о номере вершины и оси, перемещение вдоль которой описывает данное значение. Зная порядковый номер вершины, легко найти информацию о ее перемещении в последовательности данных. Посылка о перемещении одной вершины в данном методе имеет вид, показанный в табл. 2:

Величина смещения вдоль оси X	Величина смещения вдоль оси Y	Величина смещения вдоль оси Z

Таблица 2. Последовательность данных при использовании второго метода передачи

При использовании второго метода передачи (сохранения) информация о движении содержит большое количество избыточных данных (последовательности нулей для вершин, не меняющих своего положения от кадра к кадру). В этом случае компрессия данных достигается за счет удаления избыточности путем применения к этим данным арифметического кодирования, кодирования по Хаффману или алгоритма RLE.

Уменьшение разрядности данных

Для уменьшения объема передаваемых данных можно уменьшать разрядность, с которой в компьютере представляются данные о перемещении модели. В этом случае появляется вероятность передачи движения некоторых вершин с ошибкой. Так, если вершина от кадра к кадру движется медленно, то вся информация, описывающая ее движение, хранится в младших разрядах. Если понижать разрядность данных, то эта информация может потеряться, вершина будет оставаться неподвижной, и ошибка о ее

положении будет накапливаться с каждым кадром. Для компенсации этой ошибки можно использовать алгоритм, представленный на рисунке.

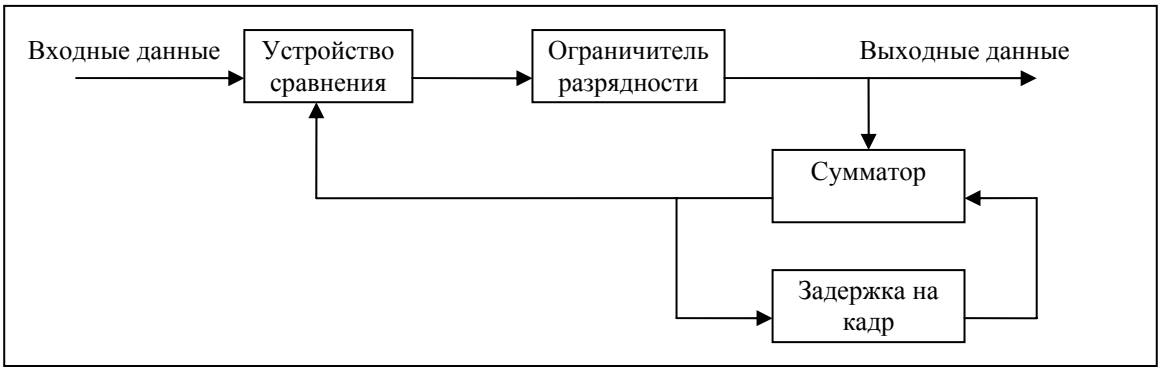


Рисунок. Схема кодера, реализующего уменьшение разрядности данных и компенсацию ошибок

Входные данные (информация о перемещении вершин) поступают на ограничитель разрядности, откуда уходят на выход и также поступают на сумматор, где суммируются с данными из предыдущего кадра. Далее эта информация сравнивается с входными данными нового кадра, и, если суммарное расхождение становится больше, чем величина младших (отбрасываемых) разрядов, то оно добавляется к данным о перемещении вершины в новом кадре. Для увеличения точности представления модели задержку информации о перемещении вершин можно сделать несколько кадров.

Методика измерений

Для исследования использовались несколько моделей человеческой головы, имеющих различную плотность полигональной сетки, чтобы можно было установить зависимость величины компрессии от сложности используемой модели. В табл. 3 представлены модели, на которых проводились исследования, в начальной и конечной фазах движения.

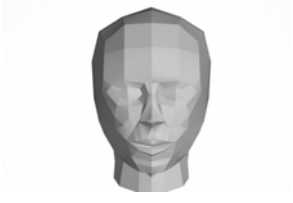
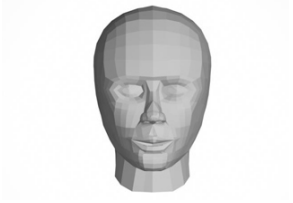
Количество вершин	215	779	2831
Вид модели (начальная фаза)			
Вид модели (движение 1 – улыбка) (конечная фаза)			

Таблица 3. Используемые в исследовании модели

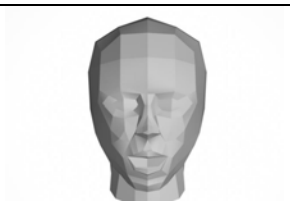
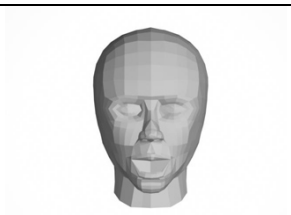
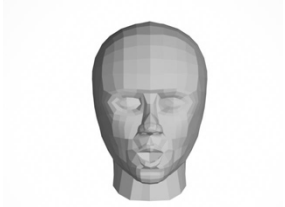
Вид модели (движение 2 – открытый рот) (конечная фаза)			
Вид модели (движение 3 – звук «О») (конечная фаза)			
Вид модели (движение 4 – грусть) (конечная фаза)			

Таблица 3. Используемые в исследовании модели (продолжение)

Результаты экспериментальных исследований

В табл. 4–9 представлены объемы данных, полученные при использовании двух вариантов алгоритма ДКИМ, исследуемых в данной работе. Также в этих таблицах представлены результаты, полученные при использовании наиболее распространенных современных форматов записи информации о 3D-моделях (OBJ, DXF, VRML). В случае ДКИМ в первом кадре передавалась вся информация о модели по стандарту OBJ (текстурные координаты, координаты нормалей и описание граней), для следующих кадров передавались величины перемещений (вектора движения). Для описания перемещения одной вершины вдоль одной оси используется два байта. В скобках указан объем данных, без первого кадра. Данная информация наиболее правильно отражает разницу в объеме данных, получаемых при использовании алгоритма ДКИМ относительно других форматов.

	ДКИМ 1	ДКИМ 2	OBJ	VRML
Объем данных (кБ)	549(51,7)	580(82,9)	2910(2420)	5190(4330)
Объем данных после арифметического кодирования (кБ)	164(17,7)	156(10,3)	902(755)	807(672)

Таблица 4. Сравнение форматов, модель из 2831 вершины, движение 4, 6 кадров

	ДКИМ 1	ДКИМ 2	OBJ	VRML
Объем данных (кБ)	193(1,6)	214(22,8)	1120(959)	1370(1140)
Объем данных после арифметического кодирования (кБ)	52,8(1,23)	52,7(1,09)	313(262)	272(228)

Таблица 5. Сравнение форматов, модель из 799 вершин, движение 4, 6 кадров

	ДКИМ 1	ДКИМ 2	OBJ	VRML
Объем данных (кБ)	51,2(1,11)	56,4(6,29)	301(250)	376(313)
Объем данных после арифметического кодирования (кБ)	14,7(0,945)	14,6(0,844)	82,6(68,8)	82,3(68,6)

Таблица 6. Сравнение форматов, модель из 215 вершин, движение 4, 6 кадров

	ДКИМ 1	ДКИМ 2	OBJ	VRML
Объем данных (кБ)	541(43,2)	581(82,9)	2920(2430)	5190(4330)
Объем данных после арифметического кодирования (кБ)	163(17,1)	157(11,4)	898(751)	775(649)

Таблица 7. Сравнение форматов, модель из 2831 вершины, движение 2, 6 кадров

	ДКИМ 1	ДКИМ 2	OBJ	VRML
Объем данных (кБ)	198(6,89)	214(22,8)	1120(959)	1370(1140)
Объем данных после арифметического кодирования (кБ)	55(3,44)	54,3(2,72)	309(258)	271(227)

Таблица 8. Сравнение форматов, модель из 799 вершин, движение 2, 6 кадров

	ДКИМ 1	ДКИМ 2	OBJ	VRML
Объем данных (кБ)	50,8(1,56)	55,6(6,29)	295(246)	376(313)
Объем данных после арифметического кодирования (кБ)	14,8(1,18)	14,7(1,08)	82(68,3)	82(68,3)

Таблица 9. Сравнение форматов, модель из 215 вершин, движение 2, 6 кадров

Обсуждение полученных результатов

В работе выполнены исследования величины компрессии данных, описывающих локальное движение отдельных вершин полигональной 3D-модели в пространстве (на примере мимики человеческого лица) при использовании в качестве алгоритма компрессии ДКИМ и проведено сравнение исследуемого алгоритма с наиболее распространенными форматами хранения данных о трехмерных моделях.

При анализе результатов исследований обращает на себя внимание тот факт, что с увеличением длительности последовательности кадров величина компрессии, получаемая при использовании алгоритма ДКИМ, относительно других форматов возрастает. Большая часть информации о модели (координаты вершин, вектора нормалей текстурные координаты и информация о гранях (полигонах)) в варианте с использованием ДКИМ передается в первом кадре. Таким образом, при короткой последовательности данные из первого кадра составляют большую часть информации посылки (файла), и общий объем данных мало отличается от объема данных, получаемых при использовании других форматов. Однако с увеличением длительности последовательности кадров доля первого кадра в общем объеме данных уменьшается, и величина компрессии возрастает.

Эффективное использование полученных результатов возможно в области передачи данных, описывающих движение 3D-модели, таких как системы передачи, хранения и воспроизведения трехмерных изображений, синтезированных на компьютере [4], видеоконференции, в которых вместо реального изображения диктора передается его трехмерная модель, анимированная в соответствии с движением говорящего, а также для разработки алгоритмов сжатия мультимедиа данных.

Литература

1. Евсеев Г. Maya3: Трехмерная графика и визуализация. М.: Десс Ком, 2001. 448 с.
2. Мараффи К. Создание персонажей в Maya. Моделирование и анимация. М.: Вильямс, 2004. 448 с.
3. Ковалгин Ю.А., Вологдин Э.И. Цифровое кодирование звуковых сигналов. М.: Корона-принт, 2004. 240 с.
4. Красильников Н.Н. Цифровая обработка изображений. М.: Вузовская книга, 2001. 320 с.

ПРОБЛЕМА ВЫЯВЛЕНИЯ ФУНКЦИОНАЛЬНЫХ ЗАВИСИМОСТЕЙ В РЕЛЯЦИОННОЙ БД

Е.Ю. Симоненко (Université de la Méditerranée Aix-Marseille II)
Научный руководитель – доктор информатики Ноэль Новелли, Docteur en Informatique, Maître de conférences N. Novelli (Université de la Méditerranée Aix-Marseille II)

Алгоритмы TANE, DEPMINER, FUN, FastFDs и FD_MINE решили проблему выявления функциональных зависимостей (ФЗ) в базе данных. Публикации по проблеме прекратились. Рост объемов БД, передача данных с помощью потоков стали причиной невозможности их обработки по известным алгоритмам. Проблема вновь стала актуальной и требует нового решения. Это отразилось в росте публикаций. Инкрементальный алгоритм позволит эффективно рассчитывать ФЗ, а также приближенные ФЗ.

Введение

С проблемой выявления ФЗ в реляционной БД связано множество публикаций, самые важные из них описывают алгоритмы TANE [1], DEPMINER [2], FUN [3], FastFDs [4] и FD_MINE [5]. С 2002 года публикации прекратились. Это может свидетельствовать, в том числе, и о прекращении исследований в данном направлении. Однако в 2006 году интерес к проблеме, видимо, появился снова. Проблема стала одной из основных в направлениях работы конференций по БД. С целью изучения проблемы и причин обновления интереса к ней, совместно с Российской национальной библиотекой им. М.Е. Салтыкова-Щедрина был выполнен информационный поиск публикаций на английском, французском и русском языках. Оказалось, что на русском языке они практически отсутствуют. В 11 публикациях периода 1995–2002 гг. предлагаются наиболее эффективные алгоритмы извлечения ФЗ. Рассматривались теоретические основы каждой публикации, предложенный подход к проблеме и его ограничения.

Выяснилось:

1. большинство существующих алгоритмов неэффективны при современных объемах данных и передаче данных потоками (streaming);
2. необходим алгоритм, обрабатывающий данные инкрементально, т.е. по мере их добавление в БД, без пересчета всего множества уже найденных ФЗ. Такой алгоритм, INC_MINER, был предложен в 2002 году [6], но он имеет множество ограничений, в том числе не работает с приближенными ФЗ.

Подобные обзоры проводились и ранее [1, 7], для описания состояния проблемы на момент предложения нового алгоритма. Данный же обзор проводился с целью выявления перспективного направления исследований в области БД.

Поиск информации по проблеме

Для анализа состояния проблемы исследования был проведен информационный поиск публикаций по теме выявления функциональных зависимостей в базах данных. Особое внимание было уделено информации, появившейся с 1995 года. До 1995 года в публикациях преобладал классический подход к решению проблемы, и не было предложено практической реализации описанных алгоритмов. Кроме того, эти алгоритмы были неэффективны в реальных условиях. Поэтому, за исключением [8] и [9], предлагающих классические алгоритмы, мы выбрали публикации, начиная с создания алгоритма TANE [1], авторы которого впервые использовали подход к проблеме с точки зрения data mining.

Поиск осуществлялся как в российских, так и в зарубежных источниках с помощью информационно-сервисного центра Российской национальной библиотеки.

Публикаций, вышедших после 2002–2003 года, найдено не было. Для выяснения причин и был проведен данный обзор. Из журнальных публикаций, материалов конференций, диссертаций и дипломных работ были отобраны лишь предлагающие новый подход к решению проблемы, либо отличающиеся наибольшим числом ссылок на них.

Итак, рассмотрим важнейшие подходы в порядке их значимости для решения проблемы, обозначив заголовком алгоритм.

TANE и его предшественники

TANE и его предшественники содержат классическое определение функциональной зависимости (*F-зависимость, ФЗ*): если значения кортежа на некотором множестве атрибутов единственным образом определяют значения на другом множестве атрибутов, говорят, что имеет место *функциональная зависимость*.

Дадим формальное определение.

Пусть r – отношение со схемой R, X и A – подмножества R. Отношение r удовлетворяет функциональной зависимости $X \to A$, если **(** r **WHERE (** $X = x$ **))[** A **]** имеет не более одного кортежа для каждого X-значения x. Другими словами, возьмем любые два кортежа $t1$ и $t2$. Если $X \to A$ и $t1(X) = t2(X)$, то $t1(A) = t2(A)$.

ФЗ $X \to A$ *тривиальна*, если $A \in X$.

ФЗ $X \to A$ *минимальна*, если для любого $Y \subset X$ не выполняется зависимость $Y \to A$.

ФЗ называется *приближенной* (approximate), если она *почти* выполнена [1]. *Почти* означает, что для ее выполнения достаточно удалить некоторое количество кортежей. Для измерения степени приближения авторами вводится понятие *погрешности* e. Авторы [1] определяют e как отношение минимального числа кортежей, которые достаточно удалить из отношения для выполнения зависимости $X \to A$, к общему числу кортежей. Авторы [8] и [9] вводят несколько определений погрешностей, причем e соответствует $g3$.

Все рассмотренные алгоритмы направлены на сокращение пространства, на котором ведется поиск, для этого авторами алгоритмов предлагаются новые характеристики, позволяющие отсеивать кандидатов в ФЗ.

В классических подходах к извлечению ФЗ [8, 9], предшествующих TANE, алгоритмы основываются на сравнении кортежей [8] и на операции сортировки [9]. Для каждого атрибута A алгоритм ищет комбинацию атрибутов X, для которой выполняется ФЗ $X \to A$, последовательно сравнивая все кортежи между собой. Это делает невозможным использование классических алгоритмов в реальной жизни.

Рассматривая проблему извлечения ФЗ с точки зрения data mining, TANE является новаторским алгоритмом. TANE опирается на концепт разбиения отношения на классы эквивалентности (partition), изначально определенный в [12, 13], и предлагает усовершенствование этого понятия (refinement), что позволяет по-другому охарактеризовать ФЗ. Отношение разбивается на классы в соответствии со значением на выбранном атрибуте. Определяется произведение классов, позволяющее легко получить ФЗ. Кроме того, адаптируя определение погрешности e к классам эквивалентности, TANE извлекает приближенные ФЗ. Это позволяет учитывать гипотезу об ошибочных данных, так как приближенные ФЗ выполняются только частью кортежей. Используя принципы алгоритмов извлечения ассоциативных правил [14, 15], TANE находит множество всех минимальных нетривиальных ФЗ (минимальное покрытие ФЗ).

Таким образом, адаптируя принципы data mining для извлечения ФЗ, TANE доказал свою эффективность по сравнению с классическими алгоритмами и возможность применения на отношениях больших размеров. Извлечение же им приближенных ФЗ позволяет его применять к данным, содержащим ошибки (как всегда бывает в операционных БД).

Dep-Miner

Этот подход [2] реализует извлечение минимальных нетривиальных ФЗ, а также автоматический вывод отношений Армстронга, что облегчает задачу администрирования при обслуживании и анализе существующих БД. Dep-Miner также использует принцип алгоритмов [14, 15].

Применяя оптимизированное представление отношения в виде классов эквивалентности (используемое в TANE), Dep-Miner реализует один из алгоритмов, описанных в [8], и предлагает новую характеристику согласованных множеств (agree sets). Для каждой пары кортежей согласованное множество объединяет атрибуты, имеющие одинаковое значение. Согласованные множества рассчитываются исходя из классов эквивалентности. Из согласованных множеств выводятся максимальные множества, их гиперграф, затем множество поперечных (трансверсальных) минимумов гиперграфа, и, наконец, сами минимальные ФЗ. Подробно этот алгоритм описан в [2] и [7].

С точки зрения авторов [2], Dep-Miner превосходит TANE по нескольким параметрам, благодаря лучшей характеристике ФЗ. В то же время, Dep-Miner не позволяет работать с приближенными ФЗ.

FUN

Алгоритм FUN [7] также рассматривает проблему извлечения ФЗ с точки зрения data mining и использует представление отношения в виде классов эквивалентности. FUN опирается на новую, по сравнению с предыдущими публикациями, характеристику минимальных ФЗ, что позволяет эффективнее сокращать область поиска. FUN позволяет находить и приближенные ФЗ, используя, как и TANE, погрешность $g3$, определенную в [8, 9]. Кроме того, авторы FUN предлагают метод визуализации ФЗ, предназначенный для конечного пользователя и позволяющий ему контролировать верность любой ФЗ.

Таким образом, FUN соединяет лучшие свойства TANE и Dep-Miner, вводит новую характеристику ФЗ – проекцию минимальных ФЗ – и предлагает пользователю визуализацию ФЗ. По результатам экспериментов FUN превосходит Dep-Miner.

FastFDs

FastFDs – один из четырех наиболее эффективных алгоритмов извлечения ФЗ [4]. Как и Dep-Miner, FastFDs рассчитывает минимальные покрытия гиперграфа, однако вместо алгоритма «уровней» (level-wise), FastFDs использует обход в глубину и эвристический подход, что делает его эффективнее предыдущих алгоритмов.

Интересно, что все 4 публикации [1-4] появились с 1999 по 2002 год, причем последние 3 – с незначительной разницей во времени. При этом TANE стоит особняком как первый алгоритм, решивший проблему выявления ФЗ с помощью data mining, а FastFDs завершает этот цикл публикаций как алгоритм, отличный от TANE и не являющийся алгоритмом «уровней», но также подходящий к проблеме с точки зрения data mining.

FD_MINE

Алгоритм, описанный в [4], в свою очередь сокращает область поиска ФЗ за счет эксплуатации их свойств, это тоже алгоритм «уровней». В своей статье авторы сравнивают FD_MINE с TANE, и преимущество оказывается на стороне FD_MINE. [4] сравнительно редко цитируется другими авторами. В этот обзор он попал по настоянию научного руководителя.

INC-MINER

Все описанные выше алгоритмы рассматривают статичное представление БД (как моментальный снимок)**.** Если в базу добавляются строки, множество ФЗ, найденных с помощью алгоритма, становится недействительным.

В 2002 г. студентом факультета информатики марсельского Университета, Ксавье Рувьером (Xavier Rouviere) под руководством Розин Сиккетти (Rosine Cicchetti) и Лотфи Лакхаль (Lotfi Lakhal) была выполнена дипломная работа [6]. В ней был предложен алгоритм INC-MINER. Эта работа из 14 страниц, вклад которой в решение проблемы извлечения ФЗ, по признанию самих авторов, составляет полстраницы. В работе повторяются определения, данные в [2], и вводится новая характеристика ФЗ, основанная на свойстве согласованных множеств, обусловленном добавлением нового кортежа. Алгоритм позволяет, после добавления строки, рассчитать новое множество ФЗ с наименьшими затратами ресурсов. Однако не рассмотрено добавление нескольких строк одновременно, изменение, удаление строк. Алгоритм не рассчитывает приближенные ФЗ.

Возобновление интереса к проблеме

Отсутствие публикаций в период с 2002 по 2006 годы может свидетельствовать о достаточном соответствии уровня решения проблемы потребностям практики. С 2006 году проблема извлечения ФЗ снова активно обсуждается на конференциях. Это связано, во-первых, с тем, что эффективности всех существующих алгоритмов уже недостаточно для обработки возросшего объема данных. Во-вторых, передача данных потоками (streaming) делает объем и содержание базы данных постоянно изменяющимися. При этом сложно предугадать количество данных, которые будут получены в следующий момент. Поэтому подход к базе данных как к ее моментальному снимку устарел, а с ним и использующие его алгоритмы.

Единственный алгоритм, предлагающий инкрементальное решение проблемы извлечения ФЗ, мало известен и нигде не применяется. Основной причиной этого, вероятно, является его недостаточная эффективность для реальных данных.

Заключение

Проблема извлечения ФЗ из реляционных отношений была актуальна и обсуждалась в научной печати и на конференциях на протяжении пятнадцати лет, с 1987 по 2002 годы. Пик публикаций приходится на 1999–2001 гг., в это время были созданы наиболее эффективные из существующих алгоритмы: TANE, DEPMINER, FUN, FastFDs. Проблема была решена, необходимость извлечения ФЗ исчезла.

Однако любую научную проблему могут сделать вновь актуальной потребности практики. Так и произошло с рассматриваемой проблемой. Можно поэтому предполагать, что появление инкрементального алгоритма, извлекающего, кроме точных ФЗ, приближенные, вновь переведет проблему в число решенных. Она снова потеряет свою актуальность. И это отразится в числе публикаций.

Но бурный рост объемов и способ передачи данных, потребности их обработки позволяют предположить, что эта проблема потребует все новых решений. Поэтому мониторинг публикаций является актуальной задачей исследования не только самого существа проблемы, но и ее связи с практикой. В данной работе было рассмотрено, в основном, изменение актуальности проблемы, но это и является самым главным для ее понимания.

Литература

1. Huhtala Y., Karkkainen J., Porkka P., Toivonen H. TANE: An Efficient Algorithm for Discovering Functional and Approximate Dependencies // The Computer Journal, 1999. P. 100–111.
2. Lopes S., Petit J.M., Lakhal L. Efficient Discovery of Functional Dependencies and Armstrong Relations // Proceedings of the International Conference on Extended Database and Technology (EDBT'00), volume 1777 of Lecture Notes in Computer Science. P. 350–364.
3. Novelli N., Cicchetti R. Functional and Embedded Dependency Inference: A Data Mining Point of View // Information Systems, 2001, 26(7). P. 477–506.
4. Wyss C., Giannella C., Robertson E. FastFDs: A Heuristic-Driven, Depth-First Algorithm for Mining Functional Dependencies from Relation Instances // Proceedings of Data Warehousing and Knowledge Discovery (DaWaK 2001). Munich. 2001.
5. Yao H., Hamilton H.J., Butz C.J. FD_MINE: Discovering Functional Dependencies in a Database Using Equivalences // University of Regina. Computer Science Department. Technical Report CS-02-04. August, 2002. ISBN 0-7731-0441-0.
6. Rouviere X., Inference Incrementale des Dependances Fonctionnelles dans une Base de Donnees Relationnelle // Universite de la Mediterranee. Aix-Marseille II. Memoire de DEA 2002.
7. Novelli N. Extraction de Dependances Fonctionnelles dans les Bases de Donnees : une Approche Data Mining // These de l' Universite de la Mediterranee. Aix-Marseille II. 2000.
8. Mannila H., Raiha K.-J. Algorithms for Inferring Functional Dependencies // Data & Knowledge Engineering, 1994, 12. P. 83–99.
9. Mannila H., Raiha K.-J. The Design of Relational databases // Addison Wesley, 1994.
10. Dalkilik M.M., Gucht D.V., Robertson E.L. CE: the classifier-estimator framework for data mining // Proceedings of 7thIFIP 2.6 Working Conference on Database Semantics (DS-7). Leysin, Switzerland. 1997.
11. Kivinen J., Mannila H. Approximate Dependency Inference from Relations. // Theoretical Computer Science, 1995. 149. P. 129–149.
12. Spyratos N. The Partition Model : A Deductive Database Model // Technical Report Rapports de recherché n.286, Institut National de Recherche en Informatique et en Automatique (INRIA). 1984.
13. Spyratos N. The Partition Model : A Functional Approach // Technical Report Rapports de recherché n.430, Institut National de Recherche en Informatique et en Automatique (INRIA). 1985.
14. Agrawal R., Srikant R. Fast Algorithms for Mining Association Rules // Proceedings of the 0^{th} International Conference on Very Large Data Base (VLDB'94). Santiago, Chili. September 1994. P. 487–499.
15. Mannila H., Toivonen H. Levelwise search and borders of theories in knowledge discovery // Data Mining and Knowledge Discovery, 1(3). 1997. P. 241–258

ПАТТЕРНЫ ПРОЕКТИРОВАНИЯ ВСТРОЕННЫХ СИСТЕМ

А.Н. Лукичев
Научный руководитель – к.т.н., доцент А.Е. Платунов

В статье рассматриваются современные методики проектирования встроенных систем и понятие *паттерна проектирования* в их контексте. Анализируются проблемы применения паттернов для проектирования встроенных систем и предлагаются пути их решения.

Введение

За последние несколько лет в области встроенных систем (ВсС) существенно возрос интерес к различным подходам и методам высокоуровневого проектирования. Предлагаемые в результате исследований методики призваны решить такие проблемы, как стремительно возрастающая сложность ВсС, их распределенность, расширение номенклатуры элементной базы (в самом широком смысле – от проводников и дискретных электронных компонент печатных плат до технологий программирования и отладки и блоков IP/VC систем на кристалле), требования к надежности и работоспособности [4]. Такие методики [1, 2, 3, 6], в основном, имеют следующие тенденции [6]:

- повышение уровня абстракции, на котором осуществляется основной объем проектирования системы;
- повторное использование компонент;
- унификация внешних интерфейсов компонент, разработка масштабируемых архитектур;
- использование различных абстракций, таких как «модель вычислений» [2], позволяющих формализовать не только разработку отдельных компонент ВсС, но и способы их организации в систему.

Высокоуровневое проектирование требует наличия средств, позволяющих адекватно отражать и фиксировать определенные свойства системы на верхних уровнях абстракции. Одним из таких средств является паттерн проектирования [8] как способ решения некоторой технической задачи. Паттерны проектирования успешно применяются в объектно-ориентированном подходе к созданию прикладного ПО в течение как минимум последних 10 лет. Существуют попытки применения паттернов в области встроенных систем.

До сих пор, несмотря на все более частые заявления о необходимости абстрагироваться в проектировании ВсС от закрепления за компонентом системы статуса «программный» или «аппаратный», существует во многом искусственное разделение на «отрасли» разработки аппаратуры и программного обеспечения встроенных систем. В работе указывается на принципиальную близость аппаратных и программных решений и наличие общих механизмов, лежащих в их основе.

В рамках аспектной технологии проектирования ВсС [6, 16, 17] предлагается абстракция «вычислительный механизм». Понятие вычислительного механизма четко до сих пор, однако, не определено. В работе делается попытка конкретизировать это понятие. Указывается на сходство понятий «паттерн проектирования» и «вычислительный механизм» в контексте поведенческого аспекта ВсС.

Формализация паттернов проектирования в вычислительной технике в настоящее время в области ООП ограничена применением нотации UML, а в области ВсС зачастую просто отсутствует. В лучшем случае применяются схематические структурные изображения (такие как принципиальная электрическая схема для иллюстрации аппаратного паттерна [5, 12]). В работе обсуждается способ решения этой проблемы для паттернов проектирования ВсС.

Паттерны проектирования

В области ВсС, как и в любой другой области вычислительной техники, не обходится без программирования интерактивных приложений, базирующихся на платформе общего назначения (ПК, пользовательские терминалы, PDA и т.п.). В рамках объектно-ориентированного подхода (ООП) к проектированию такого рода прикладного ПО в последние несколько лет распространилась технология применения так называемых *паттернов проектирования* [8]. С точки зрения ООП, паттерн представляет собой «описание взаимодействия объектов и классов, адаптированных для решения общей задачи проектирования в конкретном контексте» [8]. Таким образом, паттерн представляет собой некий шаблон, по которому строится решение задачи. Задача проектирования при этом обладает в достаточной степени типичностью, т.е. может возникнуть при проектировании разных систем и в разное время. В то же время эта задача существует в контексте, ограничивающем возможность ее возникновения и, следовательно, возможность применения паттерна для ее решения. Согласно К. Александру, впервые употребившему понятие «паттерн» применительно к строительству зданий, «любой паттерн описывает задачу, которая снова и снова возникает в нашей работе, а также принцип ее решения, причем таким образом, что это решение можно потом использовать миллион раз, ничего не изобретая заново» [8]. Паттерн, таким образом, заключает в себе положительный опыт решения какой-либо задачи в процессе проектирования, которая потенциально может возникнуть при проектировании других систем.

В области ООП паттерны получили широкое распространение. В настоящее время существуют обширные каталоги паттернов («языки паттернов», например [9]), знание и умение применять наиболее часто используемые из них является требованием, характеризующим базовую подготовку специалиста при приеме на работу, ежегодно проводятся международные конференции PLoP (Pattern Languages of Programs) [10].

Несмотря на большую популярность паттернов проектирования в ООП, а может быть, и вследствие нее, работы по созданию паттернов в проектировании встроенного программного обеспечения ведутся многими группами исследователей, причем не только в рамках объектно-ориентированного подхода (см., например, [11, 12]). Во многих случаях, однако, это попытки адаптации существующих паттернов ООП.

Такие попытки также применяются и по отношению к проектированию аппаратуры, в частности, компонентов «систем на кристалле» (SoC). Дамашевичус и коллеги [7] дают хороший обзор попыткам применения объектно-ориентированных методик к разработке аппаратного обеспечения.

Многие работы в области разработки аппаратных компонент описывают способы решения задач в стиле, сходном с описанием паттерна проектирования [8]: описывается задача или класс задач, контекст, в котором она существует, способ ее решения, а также примеры применения описываемой технологии или методики. Например, Кишиневский и коллеги в [13] приводят методику решения задачи взаимодействия компонент в схеме с потенциально изменяемыми задержками передачи сигналов. Описывается контекст, задача («мотивация»), способ ее решения, а также обсуждаются достоинства, недостатки и перспективы, приводятся примеры применения.

Кроме того, существуют попытки систематизации паттернов проектирования встроенных систем [4] и разработки каталогов («языков паттернов» [8]). В таких каталогах (см., например, [5, 14]) собраны паттерны, решающие задачи разработки как программного, так и аппаратного обеспечения ВсС.

Сложность современных ВсС, их гетерогенность не позволяют проектировщикам создавать систему «с нуля», требуя от них применения ранее накопленного опыта в виде успешных в прошлом технических решений. Эти же особенности заставляют, по мнению многих исследователей [1, 2, 6, 7], повышать уровень абстракции, осуществляя

проектирование путем правильной композиции уже готовых (или в достаточной мере описанных на данном уровне абстракции) компонент. По такому принципу, например, в настоящее время строится проектирование SoC: основная масса компонент системы представляет собой готовые блоки IP, которые объединяются в разработанную проектировщиками структуру. Сами IP при этом могут подвергаться незначительной «адаптации» под систему. Очевидно, применение паттернов как элементов накопленного успешного опыта должно осуществляться на всех уровнях абстракции проектируемой системы, и в особенности там, где осуществляется основной объем проектирования – на самых верхних уровнях (системном [7], архитектурном [6]). Несмотря на это, практически все попытки разработки, спецификации и систематизации паттернов проектирования ВсС в настоящее время предполагают узкую специализацию под средства разработки (языки программирования, аппаратные платформы, элементную базу). Многими (в том числе и Дамашевичусом [7]), тем не менее, подчеркивается необходимость описания и применения паттернов высокоуровневого проектирования.

Программные и аппаратные решения

Встроенные системы в большинстве своем характеризуются тесным сопряжением *программных* и *аппаратных* решений. Проектирование ВсС практически всегда включает реализацию как программной, так и аппаратной части. Программное обеспечение встроенных систем взаимодействует с аппаратурой: датчиками и исполнительными устройствами. Встроенные системы создаются с целью взаимодействия с физическим миром – сбора информации и управления объектами посредством исполнительных устройств. Часть этого взаимодействия (отдельные его элементы) реализуется при проектировании аппаратным, а часть – программным обеспечением. Это разделение функциональности обычно может иметь при проектировании аппаратуры несколько вариантов, так что выбор конкретного из них зависит от решений проектировщика и доступной ему элементной базы. Современные методы (например, [1, 2, 6]) предлагают осуществлять основной объем проектирования на уровнях, не зависящих от программно-аппаратной дихотомии, проясняя во многих случаях вопрос разделения HW/SW и «отодвигая» это разделение на более поздние этапы проектирования, когда система почти готова в виде спецификаций.

Зачастую поведение компонента на системном уровне вообще не зависит от того, как он будет реализован – программно или аппаратно. Спецификацию его поведения можно отобразить как на «программную», так и на «аппаратную» платформу [2, 6], причем с одинаково устраивающими проектировщика выходными характеристиками. Кроме того, в последнее время становится все сложнее отличить программное обеспечение от аппаратного, особенно с развитием технологий реализации аппаратных проектов на ПЛИС. Применение языков описания аппаратуры, таких как VHDL, Verilog (в особенности SystemVerilog), SystemC, JHDL, фактически стирает четкую грань между «программированием» поведения и созданием аппаратного блока из имеющихся в ПЛИС ресурсов. В случае применения средств высокоуровневого описания (например, при проектировании на Esterel или в САПР SCADE [15]), спецификация компонента может быть автоматически синтезирована как в ПО, так и в аппаратуру. С точки зрения автора, при проектировании встроенных систем правильнее говорить не о «программности» или «аппаратности» компонента, а о степени последовательности либо параллельности реализуемых им вычислений.

Между тем большинство исследованных автором паттернов, относящихся к проектированию ВсС, четко формулируют свою принадлежность к «программному» или «аппаратному» контексту. Очевидно, такие паттерны найдут ограниченное применение

в методиках проектирования, делающих акцент на уровни абстракции, инвариантные к HW/SW дихотомии.

Аспектная технология. Вычислительный механизм

В рамках аспектной технологии проектирования BcC [6, 16, 17] предложен взгляд на проектируемую систему с разных точек зрения (в рамках различных *аспектов*). Каждый такой аспект учитывает лишь определенный ряд взаимосвязанных свойств системы. Примерами аспектов могут служить: вычислительный или поведенческий [6] аспект, структурный, функциональный, конструктивный, надежностный, инструментальный, экономический аспекты. В каждом конкретном случае для проектировщиков может оказаться важным произвольный набор таких аспектов, причем суть любого из них (состав свойств системы в рамках аспекта) также может быть произвольной. Принципиальным является лишь «полнота» аспекта, т.е. самодостаточность получаемых в его рамках моделей системы [16]. При таком подходе возможна независимая разработка системы целиком в рамках одного из ее аспектов [16], например, полная спецификация поведения системы в терминах некоторой вычислительной платформы, т.е. ее *реализация* на выбранной платформе.

Процесс проектирования BcC в рамках этой технологии, таким образом, охватывает и рассматривает различные сферы деятельности коллектива проектировщиков: не только разработку непосредственно вычислительной ее части, но и применение определенных инструментальных технологий, «процессуальное» обеспечение проектирования (спецификация требований, контроль версий, учет дефектов и изменений), другие виды деятельности. Зачастую такие виды деятельности, остающиеся как бы на втором плане по отношению к разработке поведения системы, являются важными не менее самой разработки и во многом определяют ее успех. В рамках этих «сопряженных» с проектированием поведения видов деятельности у каждого коллектива, очевидно, есть некоторый накопленный опыт, в том числе и положительный: применяемые технологии внутрисхемного программирования, отладки аппаратных проектов, удачные конструкторские решения, «любимая» система контроля версий, стратегия развития и продвижения на рынок продукта. Паттерны как средство спецификации и сохранения положительного опыта могли бы выступать еще одним инструментом проектирования (в дополнение к абстракциям, перечисленным в [17]) в рамках соответствующего аспекта. Применяя паттерны проектирования в рамках некоторого аспекта системы (например, удачный вариант размещения компонентов на печатной плате, который был применен ранее для платы с аналогичным составом компонентов), можно добиться эффекта, аналогичного эффекту от паттернов в области ООП [8, 9, 10]. Грамотное применение собственных паттернов для различных аспектов одной и той же системы позволит значительно повысить эффективность проектирования.

Основное внимание в проектировании все же уделяется поведенческому аспекту BcC. В рамках этого аспекта решаются проблемы организации вычислительного процесса, взаимодействия различных компонент системы в части обмена инструкциями и данными. Аспектная технология проектирования BcC предлагает термин *вычислительный механизм*, который, хотя и употребляется в [6, 16, 17], но не имеет четкого определения обозначаемого им понятия. В [18] вычислительные механизмы понимаются как «технические решения из различных областей в абстрактном представлении». Далее автор статьи проясняет смысл сказанного: «основываясь на приведенном выше определении механизма, данная абстракция играет роль основного «кирпичика» в организации вычислительного процесса. <…> вычислительный механизм решает частную задачу, например, буферизация данных, декодирование управляющего слова, выборка и

анализ команды, кэширование информационного потока». Вычислительный механизм, таким образом:

- решает некоторую частную техническую задачу;
- специфицирован в абстрактном виде, т.е. предназначен для отображения на конкретные случаи решения задач;
- является элементом организации вычислительного процесса в системе.

Исходя из вышесказанного, очевидна близость, если не идентичность, понятий *паттерна проектирования* и *вычислительного механизма* в контексте поведенческого аспекта BcC. Предлагается рассматривать эти два термина в данном контексте как взаимозаменяемые.

Спецификация вычислительного механизма

Множество предлагаемых паттернов проектирования BcC, в особенности попытки применения паттернов к проектированию аппаратуры, используют для спецификации задачи и ее решения нотацию UML, которая широко используется при спецификации паттернов в ООП. Очевидно, семантика языка заставляет разработчика подстраивать элементы HW проекта под понятия ООП: класс, объект, наследование, интерфейс и т.д. Во многих случаях наблюдаются попытки расширить нотацию UML там, где невозможно отождествить понятия аппаратного обеспечения и ООП (например, [7]). Существуют, однако, паттерны, в которых используются более адекватные языки [5]. Аспектная технология и некоторые другие методы проектирования [2] используют понятие *модели вычислений* [17] для отражения системы правил, понятий и средств, в рамках которых организуется вычислительный процесс. В рамках модели вычислений для формальной спецификации системы выбирается наиболее адекватный язык, в терминах которого отображается вычислительный процесс (например, известные графические диаграммы для представления конечных автоматов Мили или Мура или текстовый язык Esterel для спецификации синхронно-реактивной системы). Очевидно, и спецификация паттерна должна содержать описание технического решения в терминах выбранной модели вычислений. Это, в свою очередь, позволит напрямую применять паттерны в конкретных моделях, подвергая их формальное описание лишь незначительной адаптации под конкретную задачу. Язык, используемый для спецификации, при этом может быть любым подходящим для применения в рамках модели вычислений: преобразования описаний в другой синтаксис в этом случае не должны быть сложными и могут осуществляться как вручную проектировщиком, так и автоматически трансляторами (при условии формальности синтаксиса).

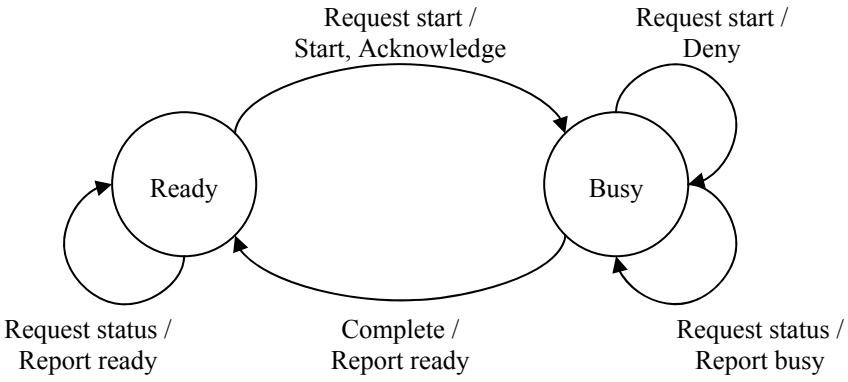


Рис. Асинхронный интерфейс компонента, выполняющего максимум одну операцию одновременно

На рисунке приведен пример спецификации паттерна, отражающего внешний интерфейс некоторого компонента системы, который способен выполнять максимум одну операцию. Интерфейс позволяет компоненту выполнять операции асинхронно по отношению к использующим его компонентам-клиентам, а им, в свою очередь, не дожидаться завершения выполнения операции после ее запроса. Для спецификации используется модель вычислений конечного автомата Мили, сам автомат представлен диаграммой переходов между состояниями.

В соответствии с механизмом, компонент может находиться в двух состояниях: готов к выполнению операции («Ready») и выполняет операцию («Busy»). Переходы между этими состояниями осуществляются по следующим условиям:

- Request start – запрос начала операции от клиента;
- Request status – запрос состояния выполнения операции от клиента;
- Complete – завершение выполнения операции.

При выполнении переходов компонент производит следующие действия:

- Start – начать выполнение запрошенной операции;
- Acknowledge/Deny – подтвердить/отклонить запрос;
- Report ready/busy – сообщить о готовности либо неготовности к выполнению.

Очевидно, паттерн не оговаривает особенности и количество выполняемых операций, способ передачи данных о запросе и о результате, состав этих данных, количество клиентов и возможность одновременного поступления запросов, а также многие другие особенности. Однако это и не требуется. Паттерн описывает абстрактный принцип построения внешнего интерфейса компонента, конкретное наполнение этого интерфейса осуществляется в результате применения паттерна при проектировании той или иной системы. Паттерн в достаточной степени абстрактен, так, что он может описывать интерфейс как компонента, потенциально реализованного программно, так и потенциально аппаратного компонента. В самом деле, *Request start* и *Request status* могут быть программными вызовами некоторого модуля, результатами которых могут являться коды возврата вызовов соответственно *Acknowledge*/*Deny* и *Ready*/*Busy*. Действие *Report ready* в результате окончания выполнения операции может являться программным callback-вызовом клиента, запросившего выполнение операции. В другом же случае действия *Request start* и *Request status* могут быть соответственно записью определенного значения и чтением из порта периферийного блока микропроцессора. При этом компонентом, к внешнему интерфейсу которого применяется паттерн, является периферийный блок, а клиентом является программный модуль-драйвер блока. Действие *Report ready* после завершения операции может быть запросом на прерывание к микропроцессору.

С другой стороны, паттерн в достаточной степени конкретен, чтобы зафиксировать отношения, которые могут возникнуть между компонентом и его клиентом. В частности, компонент не может выполнять одновременно более одной операции: после перехода в состояние «Busy» на любой новый запрос компонент выдает отказ. Также паттерн предусматривает применение в тех случаях, когда не предполагается возникновение ошибки при начале или в ходе выполнения операции: на запрос о начале в состоянии «Ready» компонент всегда подтверждает начало операции, а после начала операция всегда завершается. Кроме того, паттерн не предусматривает прерывание операции после начала ее выполнения. Таким образом, спецификация паттерна на рисунке накладывает некоторые довольно конкретные ограничения на отношения компонента и его клиентов, фиксируя способы взаимодействия с ним в системе. Применение паттерна, следовательно, представляет собой процесс фиксации определенных свойств вычислительного процесса, что, в свою очередь, является элементарным действием при проектировании поведенческого аспекта ВсС [16].

Заключение

Хотя аспектная технология проектирования BcC находится в настоящее время на этапе своего формирования [16], она уже может быть признана, по крайней мере, перспективной [6]. Многие вопросы, обсуждаемые в рамках данного метода проектирования, до сих пор не решены. В частности, нет четкого определения понятию «вычислительный механизм». В данной работе предлагается отождествить это понятие с паттерном проектирования в рамках поведенческого аспекта BcC.

Применение паттернов для проектирования встроенных систем в настоящее время не так широко распространено, как в области объектно-ориентированного проектирования прикладного программного обеспечения. В работе дается обзор паттернов проектирования BcC. В настоящее время применение паттернов в области встроенных систем довольно ограничено. Среди причин, ограничивающих применение и развитие этого довольно мощного инструмента проектирования, можно назвать, по крайней мере, две:

- четкое разделение задач (и, соответственно, решений) на «программные» и «аппаратные» на фоне развития методов проектирования BcC, стирающих эту границу. Отсутствие спецификаций паттернов на системном, не зависящем от этого разделения, уровне, а потому, с развитием упомянутых методов, бесперспективность существующих паттернов;
- отсутствие способа формализации паттернов проектирования, обеспечивающего их применение в формальных спецификациях проектируемых систем. Существующие попытки адаптировать для этих нужд язык UML (объектно-ориентированный по природе) вынуждают разработчиков прибегать к различным ухищрениям: расширять язык, находить соответствие понятий из разных моделей вычислений и т.п. По мнению автора, для спецификации решений в области BcC, пусть и абстрактных, доступно достаточное количество более адекватных формальных языков (см., например, [19]).

Предлагается повысить уровень абстракции задач в рамках проектирования, решения которых могут быть описаны паттернами. Накопление положительного опыта в области высокоуровневого проектирования с помощью паттернов позволит существенно увеличить его эффективность. Это также создаст механизм для сохранения, формализации, обмена и улучшения опыта решения традиционно возникающих в проектировании BcC задач.

Кроме того, предлагается использовать для формализации вычислительных механизмов (паттернов проектирования поведения BcC) существующие формальные языки в рамках выбранной в конкретном случае модели вычислений.

Литература

1. Alberto Sangiovanni-Vincentelli. Defining Platform-Based Design // EEDesign of EETimes, February, 2002, http://www.eedesign.com/features/exclusive/OEG20020204S0062 (электронное издание).
2. Edward A. Lee, Stephen Neuendorffer, Michael J. Wirthlin. Actor-Oriented Design of Embedded Hardware and Software Systems // Journal of Circuits, Systems and Computers, No. 3, 2002, pp. 231–260.
3. G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. V. Lopes, J.-M. Loingtier, J. Irwin. Aspect-Oriented Programming // Proceedings of the 15th European Conference on Object-Oriented Programming (ECOOP 1997), 1997, pp. 327–353.

4. David Kalinsky. Design Patterns for High-Availability Embedded Systems // Embedded.com, 2002, http://www.embedded.com/story/OEG20020729S0030 (электронное издание).
5. Michael J. Pont, Royan H.L. Ong, Chinmay R. Parikh, Ridwan Kureemun, Chen Pang Wong, William Peasgood, Yuhua Li. A Selection of Patterns for Reliable Embedded Systems // Proceedings of EuroPlop '99, Kloster Irsee, 1999, http://www.le.ac.uk/engineering/mjp9/europlop99mjp.pdf
6. Постников Н.П. Поведенческий и инструментальный аспекты проектирования встроенных вычислительных систем; дисс. канд. технических наук – СПбГИТМО (ТУ), 2004.
7. Robertas Damaševičius, Giedrius Majauskas, Vytautas Štuikys. Application of Design Patterns for Hardware Design // Proceedings of 40th Design Automation Conference, Anaheim, ACM Press, 2003, pp. 48 –53.
8. E. Gamma, R. Helm, R. Johnson, J. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software // Addison Wesley, 1995, 395 pages.
9. Ward Cunningham et al. Portland Pattern Repository, http://c2.com/cgi/wiki?PortlandPatternRepository
10. Hillside Group, http://hillside.net
11. A. Corsaro, D. Schmidt, R. Klefstad, C. O'Ryan. Virtual Component: A Design Pattern for Memory-Constrained Embedded Applications // Proceedings of 9th Conference on the Pattern Languages of Programs, 2002, http://doc.ece.uci.edu/publications/virtual_component.pdf
12. S. Key, M. Pont, S. Edwards. Implementing Low-Cost TTCS Systems Using Assembly Language // Proceedings of EuroPLoP, 2003, http://www.le.ac.uk/eg/mjp9/sakep03.pdf
13. Jordi Cortadella, Mike Kishinevsky, Bill Grundmann. Synthesis of Synchronous Elastic Architectures // Proceedings of 43rd Design Automation Conference, ACM Press, 2006, pp. 657 – 662.
14. Embedded Design Pattern Catalog, http://www.eventhelix.com/RealtimeMantra/PatternCatalog/
15. G. Berry. The Foundations of Esterel // Proof, Language and Interaction: Essays in Honour of Robin Milner, MIT Press, 2000, pp. 425–454.
16. Платунов А.Е., Постников Н.П. Единое проектное пространство плюс аспектная технология – перспективная парадигма проектирования встраиваемых систем // Научно-технический вестник СПбГУ ИТМО. Выпуск 11. Актуальные проблемы анализа и синтеза сложных технических систем. СПб: СПбГУ ИТМО, 2003ю
17. Платунов А.Е. Архитектурные абстракции в технологии сквозного проектирования встроенных вычислительных систем // Научно-технический вестник СПбГИТМО (ТУ). Выпуск 6. Информационные, вычислительные и управляющие системы. – СПб: СПбГИТМО (ТУ), 2002.
18. Платунов А.Е. Роль вычислительных моделей и механизмов в проектировании встроенных систем // Научно-технический вестник СПбГУ ИТМО. Выпуск 14. Информационные технологии, вычислительные и управляющие системы. СПб: СПбГУ ИТМО, 2004.
19. Stephen A. Edwards. Languages for Digital Embedded Systems – Kluwer Academic Publishers, Boston, 2000, 306 p.

ИСПОЛЬЗОВАНИЕ ORACLE HTML DB ДЛЯ МОДЕРНИЗАЦИИ ПРИЛОЖЕНИЯ «ДЕКАНАТ» ИНФОРМАЦИОННОЙ СИСТЕМЫ УНИВЕРСИТЕТА

М.А. Гланштейн
Научный руководитель – к.т.н., профессор В.В. Кириллов

Данная статья посвящена обоснованию необходимости, а также оценке возможности и эффективности перевода приложений информационной системы университета на Intranet-среду Oracle HTML DB. Также в работе рассмотрен пример построения приложения «Деканат» с использованием Oracle HTML DB.

Введение

В настоящее время основным средством построения приложений информационной системы университета является Oracle Developer. Данный инструмент относится к модели сетевого взаимодействия «клиент-сервер», характеризуемой тем, что локальная машина пользователя является «толстым клиентом», так как программное обеспечение приложений располагается на локальных машинах пользователей.

Вследствие того, что приложения располагаются локально, возникает множество проблем при администрировании информационной системы, так как при добавлении новых или модификации существующих приложений администратору базы данных необходимо развернуть и настроить приложение на рабочей машине каждого пользователя. Данная проблема усугубляется, если пользователей приложений много и они распределены по разным помещениям (или даже городам и странам). Также проблемой в данном случае является неоднородность программного и аппаратного обеспечения рабочих компьютеров пользователей.

Еще одной проблемой, возникающей в случае расположения форм приложений на рабочих машинах пользователей, является то, что в подобной ситуации система очень уязвима от непродуманных действий пользователей, что добавляет еще больше сложностей администратору базы данных. Также недостатком Oracle Developer являются очень ограниченные возможности в дизайне интерфейса.

Эти проблемы сейчас остро стоят для информационной системы нашего университета. Oracle Developer, используемый для построения приложений, требует очень больших затрат на администрирование.

Решением описанных проблем может служить отказ от модели «клиент-сервер» с использованием «толстого» клиента и переход на Intranet-технологию («клиент-сервер» с «тонким» клиентом).

Характеристика продукта Oracle HTML DB

Новый продукт корпорации Oracle – Oracle HTML DB – является централизованной средой разработки, предназначенной для создания и развертывания Web-приложений для баз данных Oracle.

Oracle HTML DB относится к Intranet-технологии и позволяет обеспечить доступ к приложениям без необходимости установки на локальных машинах пользователей специального программного обеспечения. Как видно из рис. 1, все специальное программное обеспечение располагается не на рабочих машинах пользователей, а на отдельной машине (машинах). Таким образом, пользователю для доступа к приложениям, построенным с помощью Oracle HTML DB, или разработчику, проектирующему приложение в этой среде разработки, достаточно на своей локальной машине иметь только Web-браузер.

Oracle HTML DB не требует трудозатрат на администрирование рабочих мест пользователей. Другими словами, добавление новых приложений или модификация

существующих не требует от администратора баз данных никаких действий по настройке приложений для каждого пользователя, так как все пользователи осуществляют взаимодействие с Oracle HTML DB через Web-браузер.

Oracle HTML DB представляет собой довольно простой и интуитивно понятный инструмент для освоения основных принципов разработки приложений. Данный продукт позволяет без специальных знаний Web-технологий и создания большого объема программного кода создавать Web-приложения для баз данных Oracle.

Что касается стоимости, Oracle HTML DB входит в стандартную поставку Oracle 10, т.е. нет необходимости приобретать его отдельно.

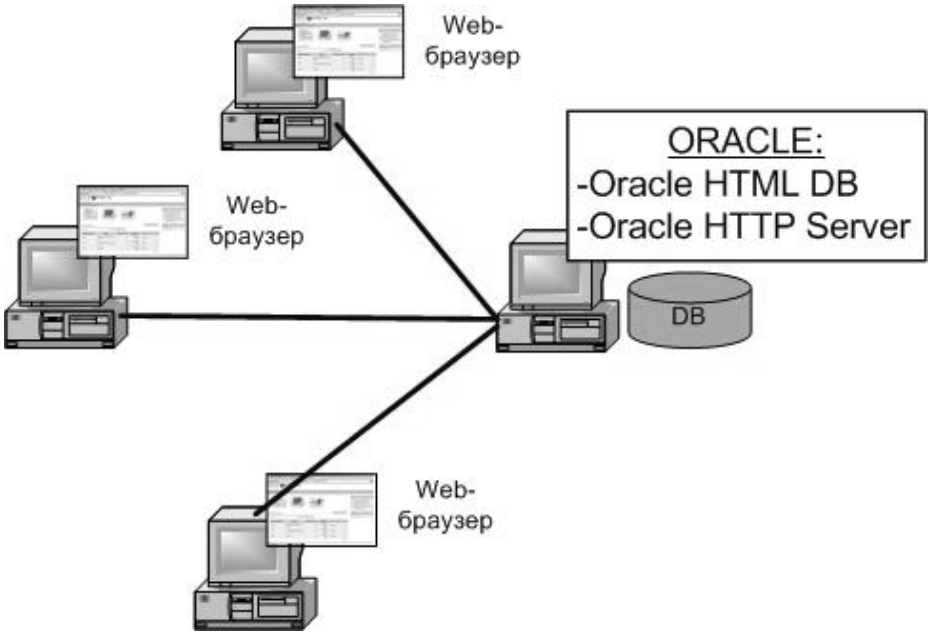


Рис. 1. Схема развертывания

Таким образом, Oracle HTML DB имеет целый ряд преимуществ перед существующими средствами разработки Web-приложений для баз данных, предоставляя разработчику приложений одновременно и простоту использования, и надежность, присущую базам данных Oracle.

Oracle HTML DB является наилучшим средством с точки зрения:

- минимизации стоимости программного обеспечения;
- минимизации затрат на администрирование;
- максимизации возможностей построения несложных приложений.

Oracle HTML DB как среда разработки Web-приложений

Средство разработки Oracle HTML DB состоит из следующих компонентов (см. рис. 2):

- конструктор приложений (Application Builder);
- SQL-мастерская (SQL Workshop);
- мастерская данных (Data Workshop).

Конструктор приложений – центральная часть HTML DB, предназначенная для создания страниц Web-приложений к базам данных.

В более узком смысле конструктор приложений используется для наложения HTML-интерфейса (приложения) на объекты базы данных, такие как таблицы, представления, процедуры и функции.

Приложение, созданное в Oracle HTML DB, представляет собой совокупность Web-страниц, основанных на информации, содержащейся в базе данных, и связанных между собой кнопками и гиперссылками.

Страница – это базовый блок (элемент) приложения. Каждая страница приложения может содержать кнопки, поля, а также элементы логики приложения. Разработчик может организовать условные переходы от одной страницы приложения к другой, производить вычисления, проверять правильность вводимых пользователем значений, а также отображать отчеты, различные диаграммы и формы. Страницы приложения соединяются между собой с помощью условных переходов.

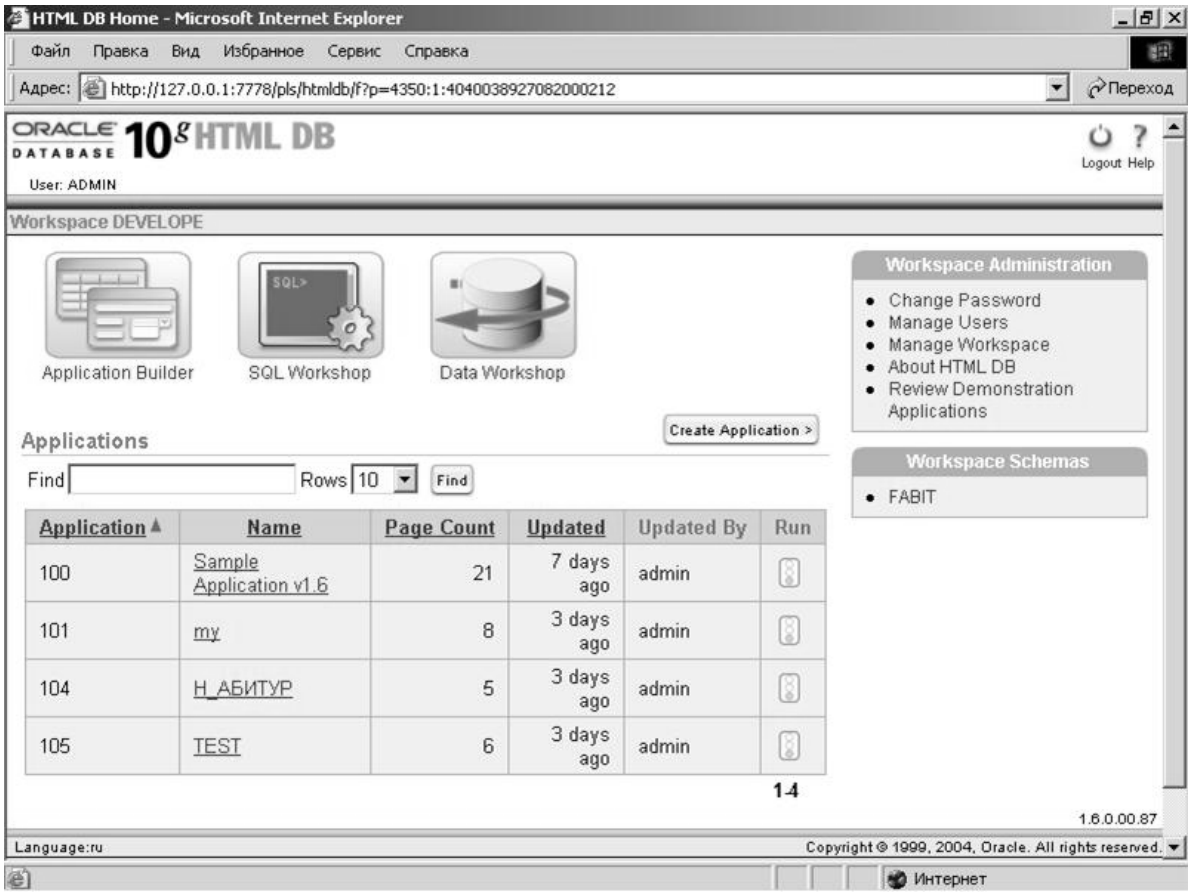


Рис. 2. Компоненты Oracle HTML DB

Oracle HTML DB предоставляет возможность создания различных отчетов, графиков и диаграмм. Можно формировать различные элементы интерфейса, такие перелистывание страниц (возможность листать, нумеровать страницы результирующего набора); сортировка по нескольким столбцам, управляемая пользователем; вставка гиперссылок на другие отчеты или формы.

Пользовательский интерфейс (оформление страниц, отчетов, кнопок, вкладок, меню, меток полей и т.д.) определяется шаблонами и элементами, которые выбираются разработчиком при создании приложения.

Таким образом, благодаря встроенным функциям, таким как шаблоны оформления (дизайна), элементы навигации, мастера создания форм, гибкая система создания отчетов, Oracle HTML DB позволяет просто создавать приложения.

Важным свойством среды разработки Oracle HTML DB является то, что ядро и хранилище данных HTML DB размещены в базе данных; таким образом, разработчик приложений имеет полный доступ к SQL и PL/SQL.

Работа с данными БД осуществляется в Oracle HTML DB выполнением операций манипулирования данными (DML) с помощью переменных связывания.

Рассмотрим алгоритм вставки пользовательских данных из формы приложения в таблицу БД на примере.

Необходимо извлечь из приложения имена и фамилии людей, а также номера их телефонов и сохранить эту информации в базе данных. Предполагая, что в БД уже имеется таблица для хранения этой информации, нужно начать с добавления к странице полей, называемых элементами (items). Элементы имеют уникальные имена, поэтому их значения можно позднее извлечь из данных сеанса (сессии). После добавления к странице кнопки «Submit» для сохранения информации в БД нужно произвести процесс вставки данных в целевую таблицу. Предполагая, что элементы названы P1_FIRST_NAME, P1_LAST_NAME и P1_PHONE, процесс вставки может быть реализован в виде следующего оператора вставки:

```
insert into T (first_name, last_name, phone_number)
Values (:P1_FIRST_NAME, :P1_LAST_NAME, :P1_PHONE);
```

Компонент **SQL-мастерская** предназначен для просмотра и управления объектами базы данных с помощью Web-браузера.

SQL мастерская позволяет извлекать и сохранять информацию в базе данных, а также производить следующие действия:

- запускать SQL-инструкции;
- загружать и исполнять SQL-скрипты;
- сохранять историю исполненных SQL-инструкций;
- создавать и изменять объекты базы данных;
- запрашивать данные по образцу;
- просматривать словарь данных.

Таким образом, SQL-мастерская позволяет пользователю взаимодействовать с базой данных таким же образом, как и с помощью инструмента SQL*Plus, но с возможностью использования всех преимуществ интуитивно-понятного графического пользовательского интерфейса и без необходимости написания «вручную» SQL-инструкций.

Мастерская данных используется для импорта данных в и экспорта данных из базы данных.

Мастерская данных Oracle HTML DB поддерживает следующие форматы данных для импорта:

- текст (с разделителями: запятая, знак табуляции);
- XML-документ;
- электронная таблица (spreadsheet).

Мастерская данных Oracle HTML DB поддерживает следующие форматы данных для экспорта:

- текст (с разделителями: запятая, знак табуляции);
- XML-документ.

Например, с помощью мастерской данных можно быстро распределить данные между различными пользователями, преобразовав электронную таблицу в таблицу базы данных при помощи мастера табличных данных (Spreadsheet Data Wizard). Данный мастер создает новую таблицу и загружает в нее данные. Данная операция не требует от разработчика создания какого-либо программного SQL-кода.

После того, как данные загружены в таблицу базу данных, можно создать приложение на основе данных этой таблицы (например, отчет), так же как и на основе любой другой таблицы базы данных.

Приложение «Деканат» информационной системы университета

В данной части описан пример использования Oracle HTML DB для создания части приложения, обеспечивающего работу деканатов университета для управления учебным процессом факультетов.

Рассмотрим использующиеся в настоящий момент деканатами университета формы «Печать экзаменационных листов» и «Ввод экзаменационного листа», предназначенные для регистрации и выдачи направлений для сдачи экзаменов и зачетов, а также для регистрации результатов данных испытаний (рис. 3, 4):

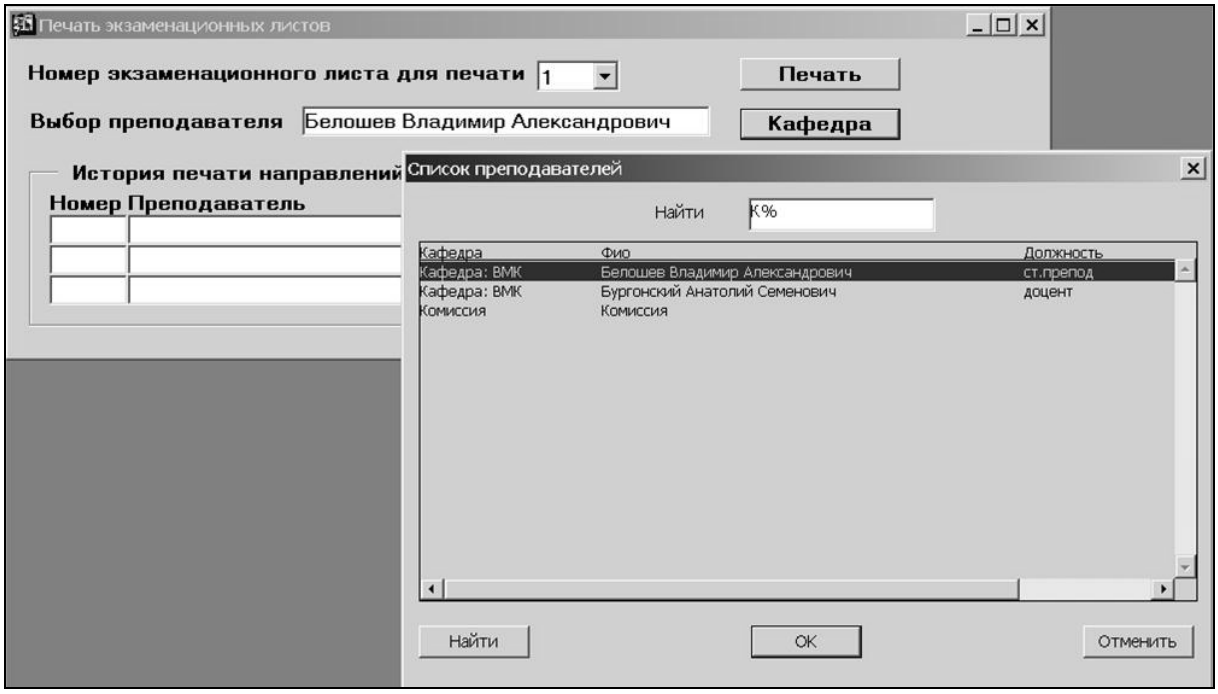


Рис. 3. Форма «Печать экзаменационных листов» (выполненная в Oracle Developer)

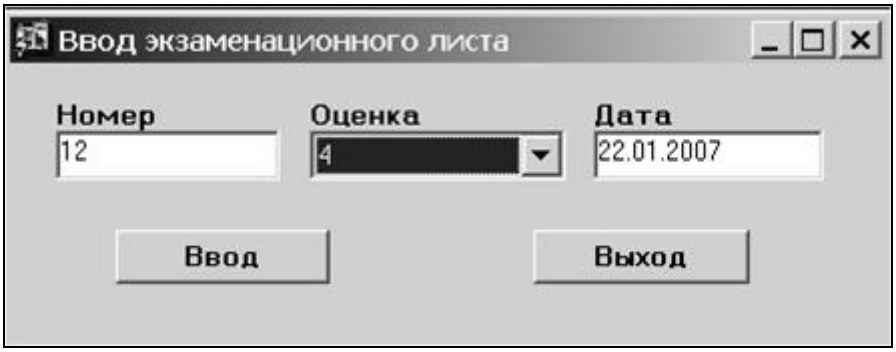


Рис. 4. Форма «Ввод экзаменационного листа» (выполненная в Oracle Developer)

Для создания приложения, позволяющего заменить данные формы, необходимо создать приложение, позволяющее:

- регистрировать выдачу экзаменационных листов (для этого необходима форма, на которой генерируется номер экзаменационного листа, а также производится выбор преподавателя из списка);
- вносить данные о полученных студентами оценках (для этого необходима форма, позволяющая вносить номер экзаменационного листа, а также полученную оценку и дату).

Для удобства заполнения и восприятия предлагается разделить данное приложение на две страницы («Печать экзаменационных листов» и «Ввод экзаменационного листа»), переход между которыми осуществляется с помощью закладок (tabs).

При успешной авторизации производится переход на страницу печати экзаменационного листа (рис. 5).

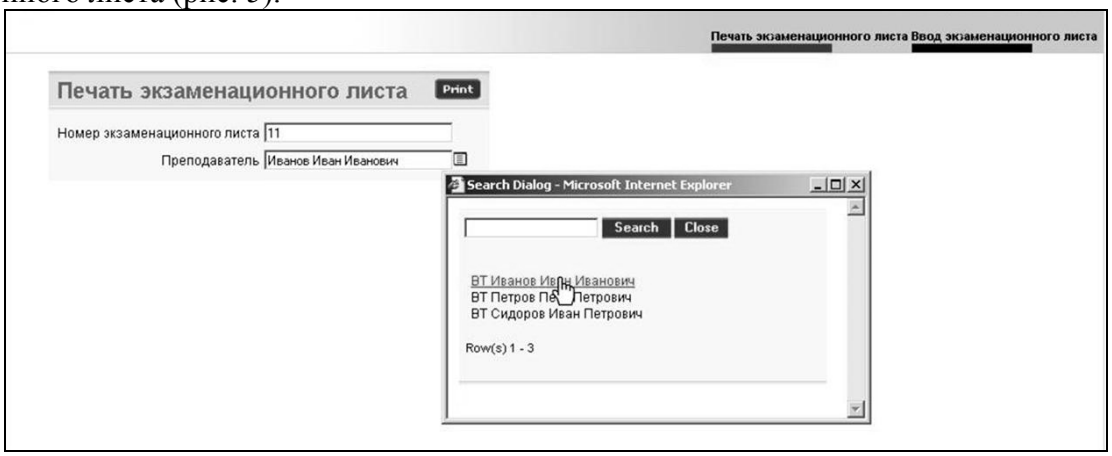


Рис. 5. Форма «Печать экзаменационных листов» (выполненная в Oracle HTML DB)

Для печати экзаменационного листа необходимо нажать кнопку «Print».

Для ввода экзаменационного листа необходимо перейти на вкладку «Ввод экзаменационного листа» (рис. 6).

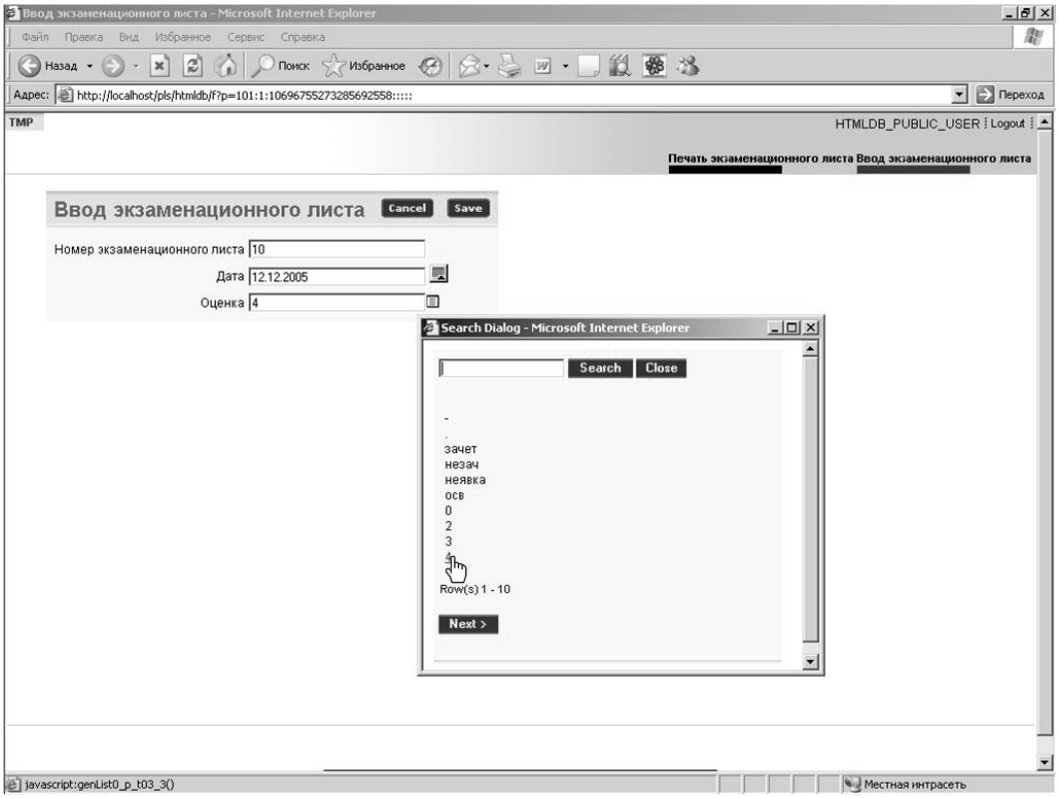


Рис. 6. Форма «Ввод экзаменационного листа» (выполненная в Oracle HTML DB)

При успешном добавлении данных (после нажатия кнопки «Save») производится переход на страницу, содержащую сообщение об успешном добавлении данных.

Интерфейс среды разработки приложений Oracle HTML DB

В данном подразделе рассматривается интерфейс среды разработки приложений Oracle HTML DB. На рис. 7 представлен интерфейс для создания и редактирования основных элементов приложения среды Oracle HTML DB, страниц приложения. На данной странице представлен список страниц, содержащихся в приложении, а также эле-

менты управления для добавления страниц приложения и для редактирования атрибутов приложения. В процессе редактирования атрибутов приложения можно изменить схему авторизации для приложения, шаблоны, с помощью которых отображается приложение и т.д.

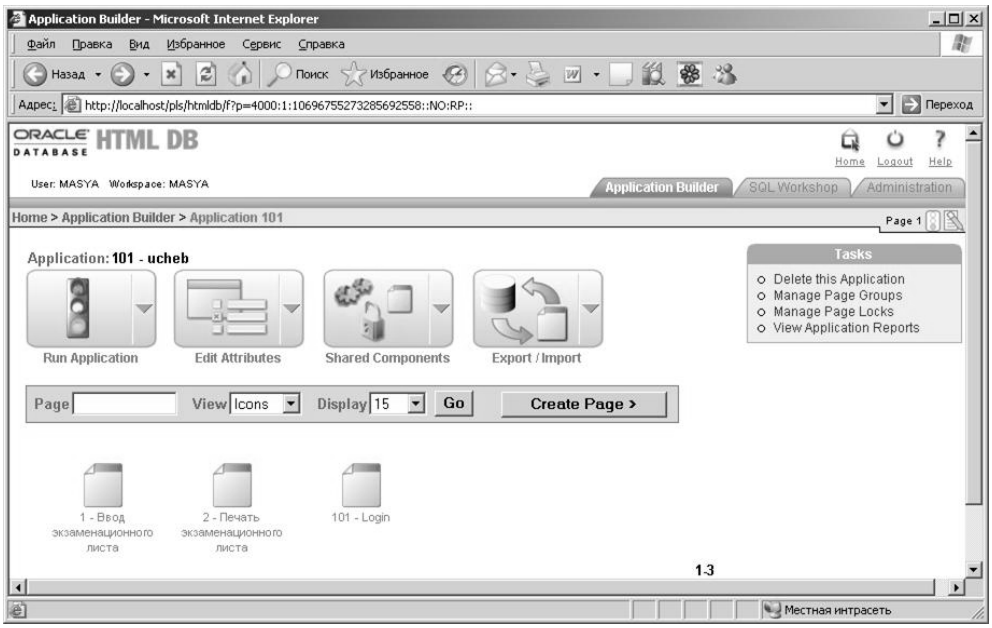


Рис. 7. Интерфейс для создания и редактирования основных элементов приложения среды Oracle HTML DB

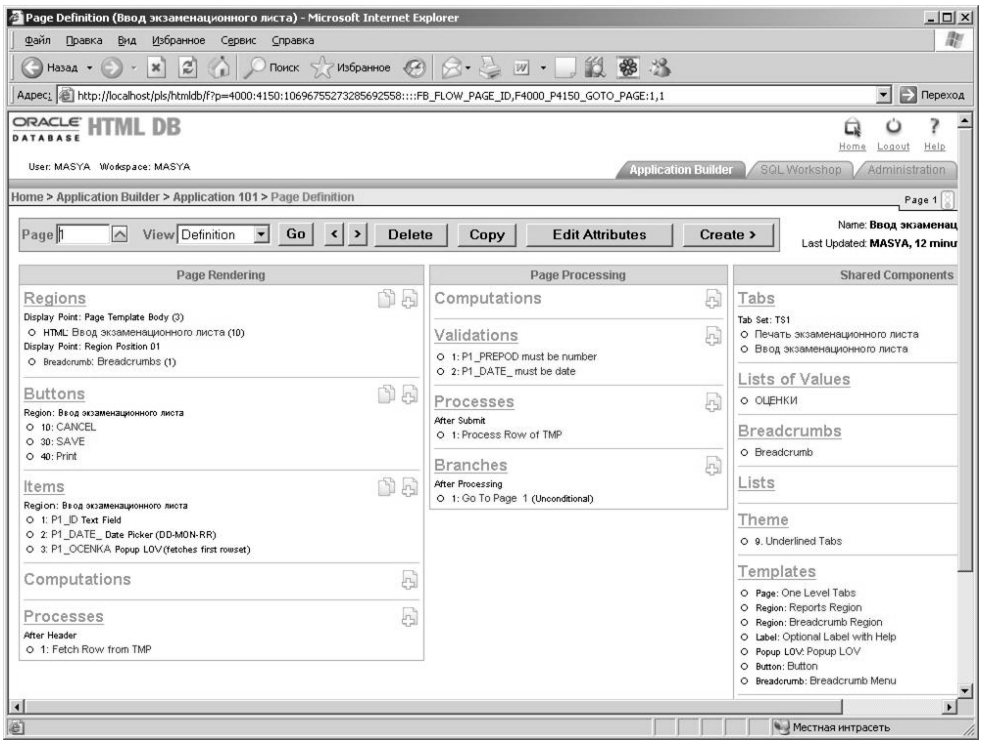


Рис. 8. Интерфейс редактирования атрибутов страниц

На рис. 8 представлен интерфейс редактирования атрибутов страниц: областей (regions), кнопок (buttons), элементов (items), процессов (processes), закладок (tabs), списков значений (lists of values) и т.д. Для создания нового элемента (например, списка значений) необходимо на интерфейсе редактирования атрибутов страницы выбрать действие создания необходимого элемента, а затем на появившейся странице задания атрибутов данного элемента задать необходимые параметры и значения (рис. 9).

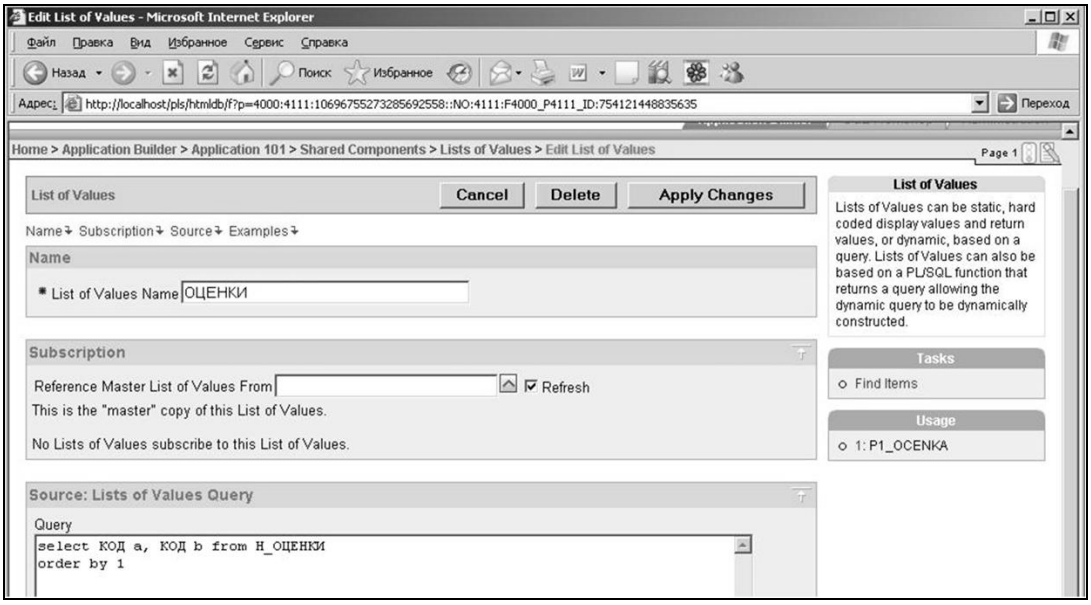


Рис. 9. Создание списка значений «Оценки» для страницы «Ввод экзаменационного листа»

Для создания списка значений задается его уникальное имя, а также запрос, на основе которого формируется список значений (если список значений динамический) или элементы списка значений (если список значений статический).

Заключение

В результате проведенных исследований среды разработки Oracle HTML DB, а также построения части приложения для информационной системы университета можно сделать вывод о том, что система Oracle HTML DB может использоваться для построения приложений в рамках данной информационной системы. Таким образом, проблема больших трудозатрат на администрирование информационной системы университета может быть решена переводом приложений на Oracle HTML DB.

Литература

1. Кириллов В.В. Основы проектирования реляционных баз данных. Учебное пособие. СПб: ИТМО, 1994. 88 с. http://www.cs.ifmo.ru, http://www.citforum.ru
2. Кузнецов С.Д. Проектирование и разработка корпоративных информационных систем. Центр Информационных Технологий, 1998. 182 с.
3. Oracle9i: Разработка программных единиц PL/SQL. Руководство слушателя. Том 1. Oracle Press, 2002.
4. Oracle® HTML DB User's Guide. Oracle Documentation Library.
5. Oracle® HTML DB Installation Guide. Oracle Documentation Library.

МНОГОСЛОЙНОЕ РЕЛЬЕФНОЕ ТЕКСТУРИРОВАНИЕ: РЕШЕНИЕ ВОПРОСОВ ФИЛЬТРАЦИИ И ПОСТРОЕНИЯ МЯГКИХ ТЕНЕЙ

А.А. Безгодов
Научный руководитель – к.т.н., доцент Э.В. Стародубцев

В работе рассмотрен вопрос отображения в реальном масштабе времени деталей поверхности, не представимых картой высот. Решаются проблемы билинейной фильтрации текстур и построения мягких теней на рельефной поверхности.

Введение

Одной из проблем современной компьютерной графики является представление высокодетализированных поверхностей. Примерами таких поверхностей могут служить поверхности скал, земли, деревьев, технических сооружений и т.д. Явное геометрическое представление таких поверхностей непрактично по причине больших затрат памяти и сложной геометрической обработки.

Существует большое количество методов представления деталей поверхностей, базирующихся на изображении (image-based). Как правило, это различные способы текстурирования. Для представления рельефных деталей, таких как заклепки, трещины, выбоины, решетки и т.д., используется ряд методов рельефного текстурирования (bumpmapping, BM):

1. emboss BM;
2. Dot3 BM;
3. environment BM;
4. parallax BM;
5. parallax Occlusion Mapping (POM).

Информация о высоте и/или нормали в каждой точке поверхности выбирается из двумерных текстур. В данной работе рассматривается вариант POM для деталей, которые не могут быть представлены картой высот (non-height-field details). На рис. 1 представлены примеры деталей, представимых картой высот (слева), и не представимых картой высот (справа).

Рис. 1. Детали, представимые и не представимые картой высот

Недостатком оригинального алгоритма, предложенного в [1], является невозможность использования билинейной фильтрации (linear filtering) текстур. Как следствие, при близком рассмотрении объекты не кажутся гладкими, визуально поверхность состоит из большого числа маленьких кубиков.

В данной работе решается проблема билинейной фильтрации текстур. Предложенный алгоритм основан на алгоритме, представленном в [1]. Также предлагается вариант построения мягких теней на поверхности от детальных элементов этой поверхности.

Проблема билинейной фильтрации в оригинальном алгоритме

Необходимым условием для работы алгоритма, предложенного в [1], является расположение слоев в текстуре строго по высоте: в красном канале должен находиться самый верхний слой, в зеленом – следующий по высоте слой и т.д. (см. рис. 2).

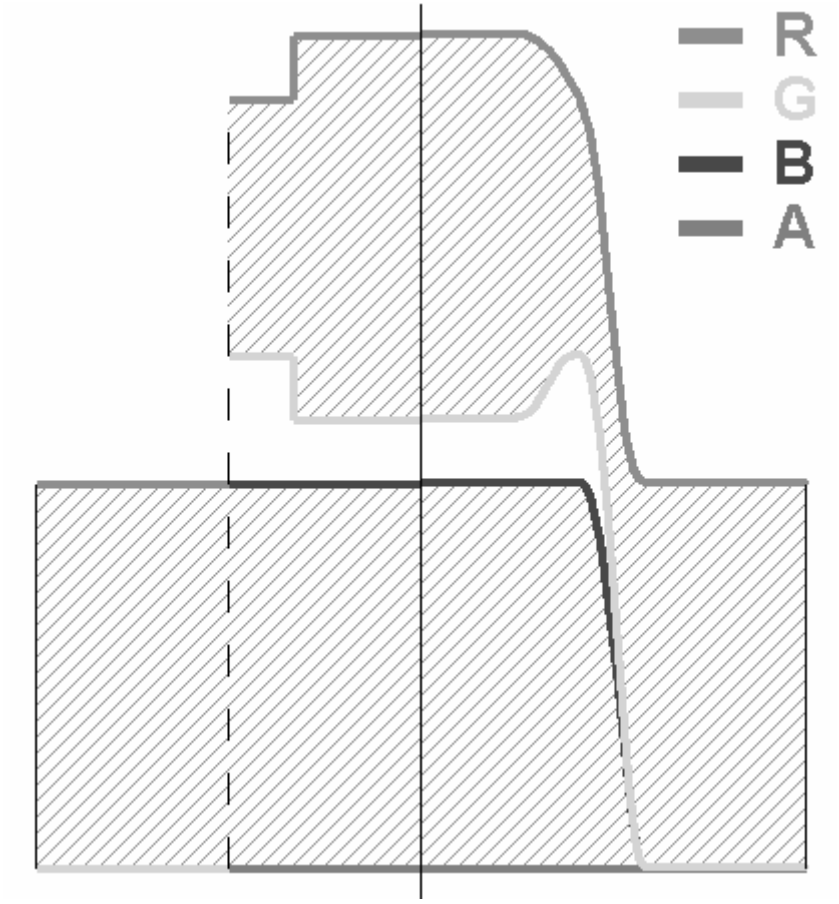


Рис. 2. Представление слоев в оригинальном алгоритме

Если используется приближенная фильтрация (nearest filtering), то карта высот имеет вид рис. 2, слева. В случае перекрытия объектов по высоте имеет место разрыв, и элементы четко разделяются. В случае билинейной фильтрации (рис. 2, справа), карта высот имеет непрерывный вид, а элементы не могут быть четко разделены, что приводит к явным дефектам изображения.

В ходе исследования проблемы рассматривались два способа борьбы с этим недостатком алгоритма.

1. Выборочная фильтрация областей текстуры, где нет разрывов. Такой подход дал неудовлетворительный результат как по качеству, так и по производительности.
2. Альтернативные способы задания слоев в текстуре. Этот подход рассматривается в данной работе и заключается в отказе от необходимости располагать слои в текстуре строго по высоте.

Описание предлагаемого алгоритма

Принцип алгоритма

Объект, представляющий элемент детализации, определяется как совокупность из двух сложных геометрических тел T_1 и T_2 (см. рис. 3). Каждое геометрическое тело ограничивается двумя поверхностями (слоями) и размерами текстуры (на рис. 2 плоскость текстуры находится в плоскости *uv*), в которой хранится информация об элементе

детализации. Одно геометрическое тело ограничено поверхностями *x* и *y*. Второе геометрическое тело ограничено поверхностями *z* и *w*.

Предлагаемый алгоритм многослойного текстурирования состоит из следующих этапов.

1. Нахождение точки пересечения луча из точки наблюдения с одним из тел. Например, луч *EA* пересекается с телом T_1, а лучи *EB* и *EC* – с телом T_2.
2. Определение текстурных координат соответствующей точки для выборки значений из диффузной текстуры и карт нормалей.
3. Определения нормали в точке пересечения.
4. Вырезка пикселей, если луч зрения не пресекает ни одно из геометрических тел.
5. Определения затенения точки. В случае, представленном на рис. 3, затененной точкой является точка *B*, а *A* и *C* – освещены.
6. Расчет освещения с учетом затенения по одной из известных моделей освещения, например по Блинну.

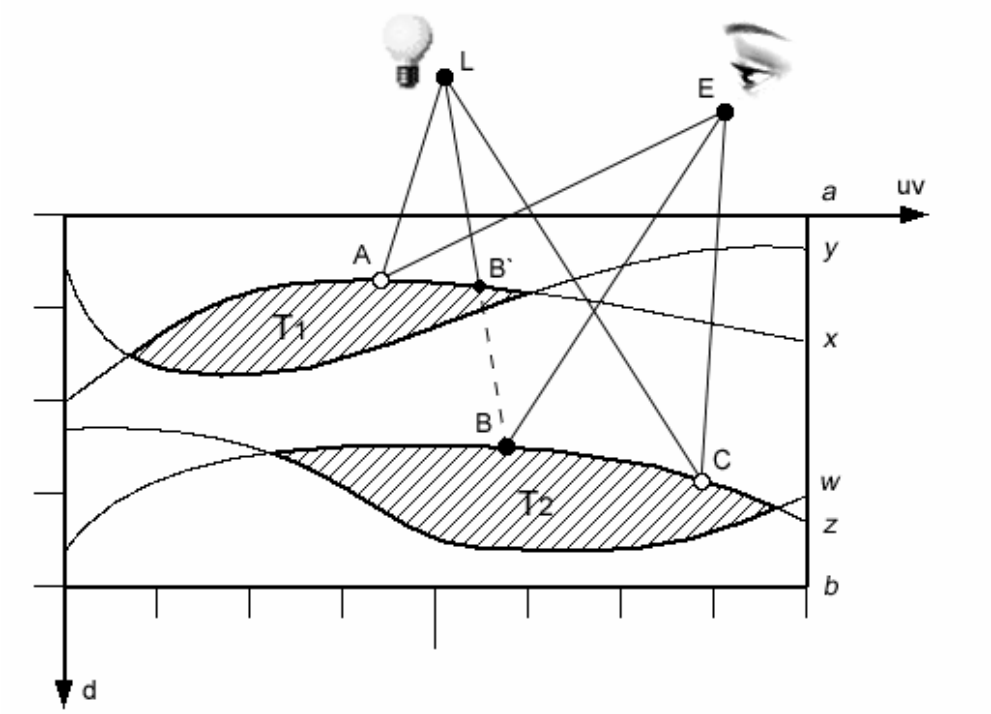


Рис. 3. Пересечения луча с поверхностью, не представимой картой высот

Представление информации о деталях поверхности

Информация о деталях поверхности выбирается из четырех текстур:

1. текстура высот. В каждом канале (R,G,B,A) хранится высота соответствующего слоя;
2. текстура *x*-компонент нормалей для каждого соответствующего слоя;
3. текстура *y*-компонент нормалей для каждого соответствующего слоя;
4. цветовая текстура. Одна для всех слоев.

На рис. 4 представлена трехмерная модель, используемая в данной работе, как пример элемента детализации поверхности.

На рис. 5 слева направо, сверху вниз представлены: диффузная текстура, карта высот, карта *x*-компонент нормалей и карта *y*-компонент нормалей для элемента детализации рис. 4. Карты высот и карты нормалей представлены поканально. Так как в текстуре могут храниться значения от 0 до 1 с восьмибитной точностью, нормали хранятся в запакованном виде – каждая компонента увеличена на единицу и разделена на два. Данные текстуры можно получать на основе трехмерных моделей.

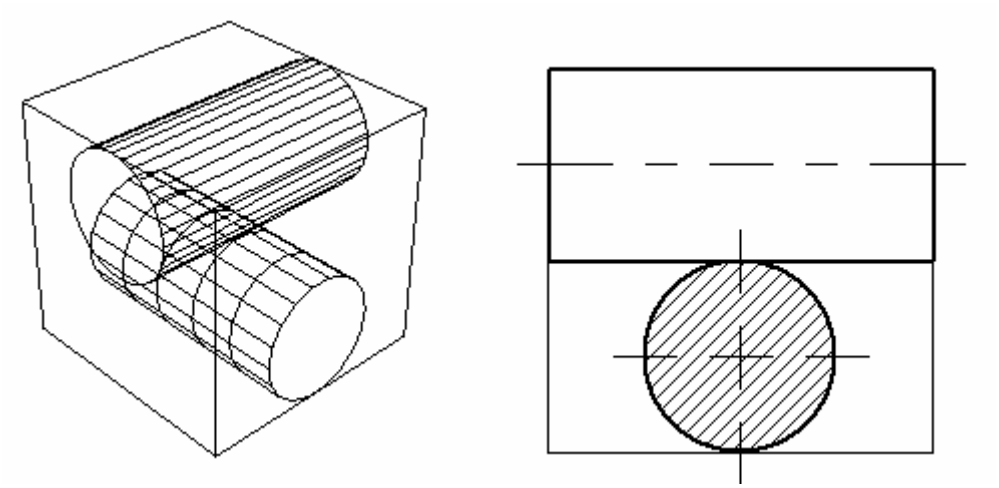


Рис. 4. Пример элемента детализации поверхности

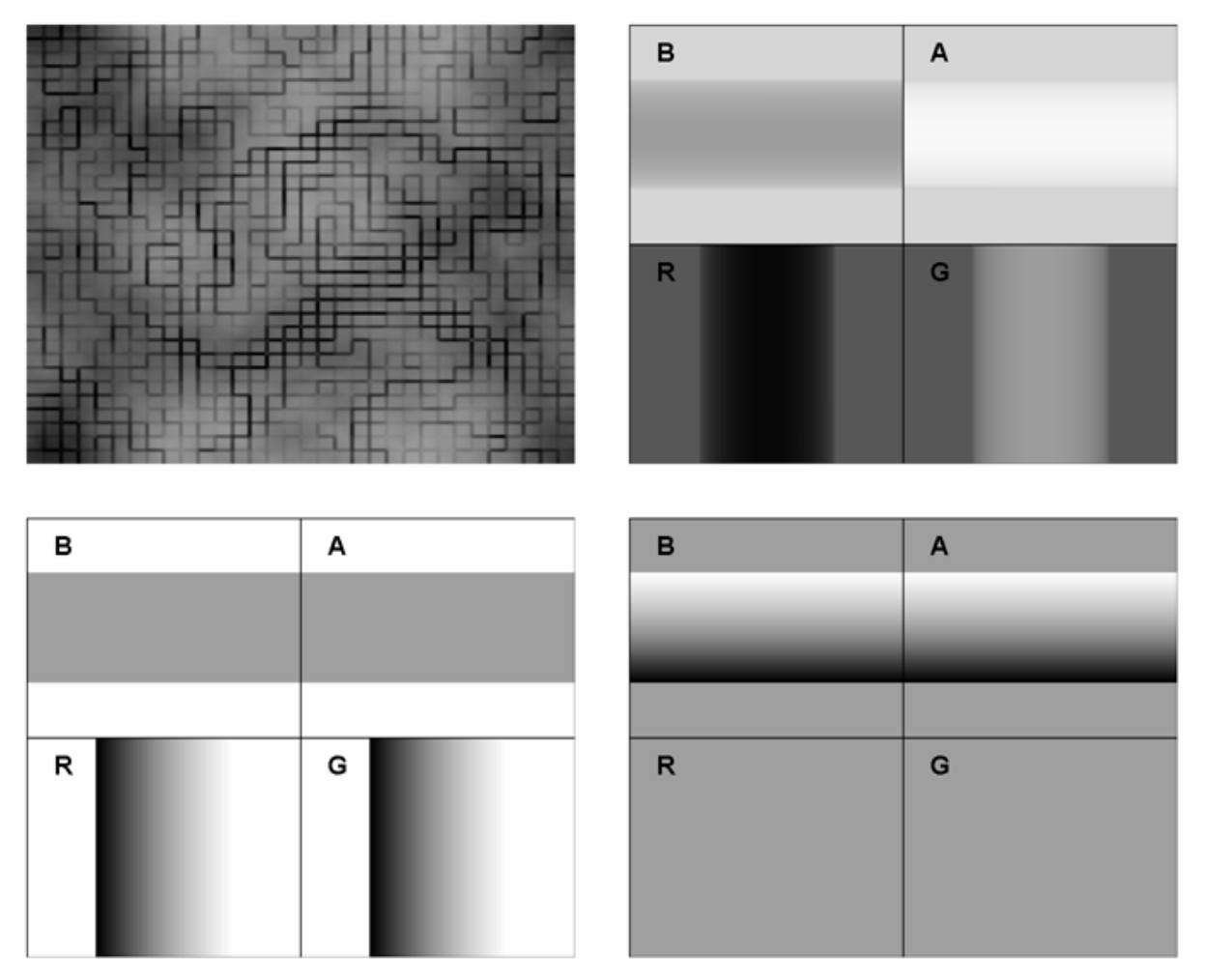


Рис. 5. Пример текстур

Определение точки пересечения

Точка пересечения луча, пущенного из глаза наблюдателя, с элементом поверхности может быть найдена путем трассировки луча в пространстве текстуры (см. рис. 6). Отрезок *SS`* делится на N равных частей. Последовательно перебираются все точки до тех пор, пока не будет найдена точка внутри одного из геометрических тел. Для каждой из точек на луче определяется позиция относительно карт высот. Условием попадания внутрь тела является

$$\left[\begin{array}{l}\begin{cases} d_x < d_y \\ d_x < d_f < d_y \end{cases} \\ \begin{cases} d_z < d_w \\ d_z < d_f < d_w \end{cases}\end{array}\right. .$$

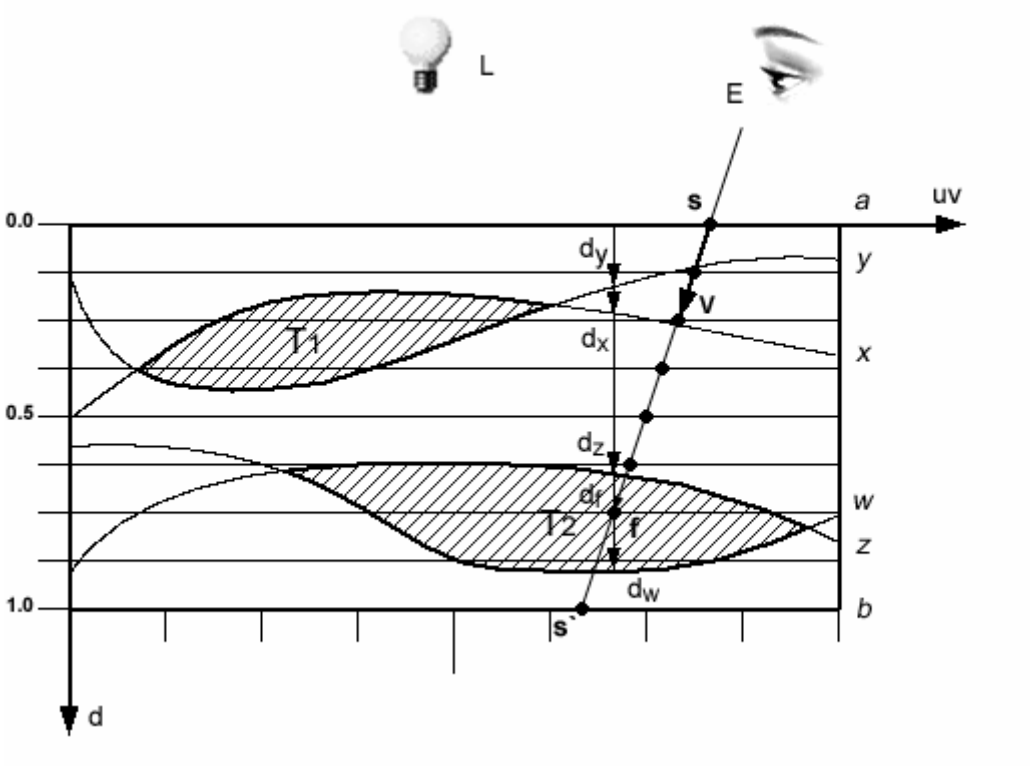


Рис. 6. Линейный поиск

Ниже приводится код функции линейного поиска на языке HLSL.

```
void QDRM_LinearSearch(
        sampler2D qdmap,
        inout   float3  s,
        in      float3  V,
        out     float   t)
{
        t       =       0;
        float dt =      1.0f / LINEAR_ITERS;
        V.z     *=      -1;
        V.xyz   /=      abs(V.z);
        V.xy    *=      SCALE;
        float dir =     sign(V.z);
        if (dir < 0) s.z = 1.0f;
                else s.z = 0.0f;

        for (int i=0; i<LINEAR_ITERS; i++)
        {
                float3 uvz              =       s + V*t;
                float4  qdh     =       tex2D(qdmap, uvz.xy);
                float   depth   =       uvz.z;
                float   factor = 1.0f;

                if ((qdh.r < qdh.g) && (qdh.r < depth) && (depth < qdh.g)) factor = 0;
                if ((qdh.b < qdh.a) && (qdh.b < depth) && (depth < qdh.a)) factor = 0;
                t += dt * factor;
        }
        s = s + V*t;
}
```

qdmap – текстурный 2D-сэмплер;

V – входная переменная, вектор в пространстве текстуры из глаз наблюдателя;

s – входная и выходная переменная. Начальная и конечная точка трассировки луча, соответственно;

t – выходная переменная, параметр уравнения прямой SS`, используется для отбрасывания визуализируемых пикселей;

LINEAR_ITERS – количество итераций в линейном поиске, порядка 10–20;

SCALE – коэффициент, определяющий рельефность объекта, в диапазоне [0...1].

Бинарный поиск

Линейный поиск дает лишь грубое приближение к поверхности, для более точного отыскания точки пересечения после линейного поиска используется бинарный поиск, см. рис. 7. Отрезок B1B2 делится точкой B3 на две равные части. Если точка удовлетворяет условию (1), то для дальнейшего деления используется B3B1, в противном случае – B2B3. Операция повторяется для полученного отрезка. B2 – точка, в которой останавливается линейный поиск, B1 – предшествующая точка.

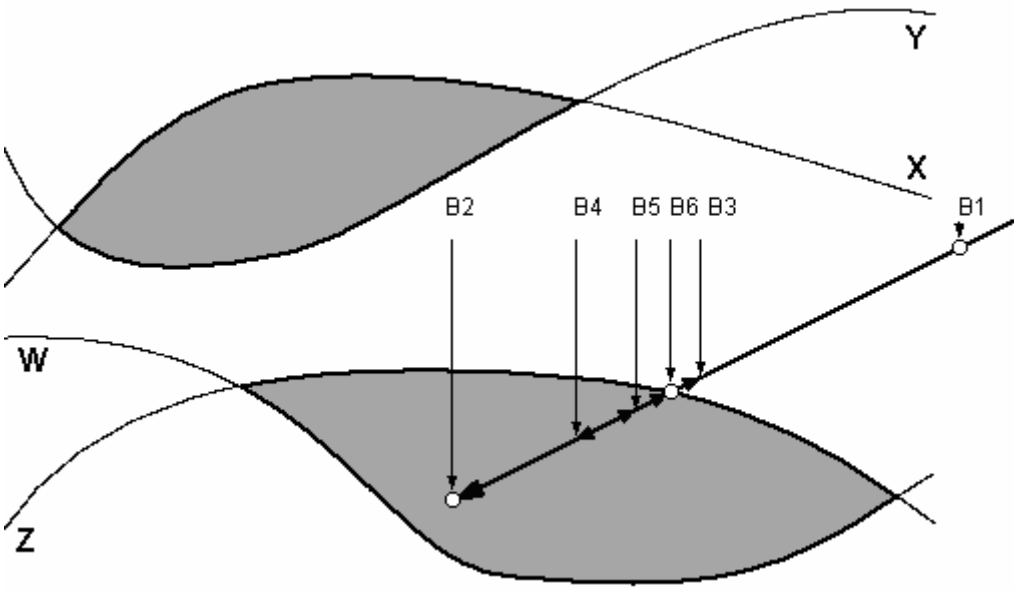


Рис. 7. Бинарный поиск

Ниже приводится код на HLSL функции бинарного поиска.

```
void QDRM_BinarySearch(
        sampler2D qdmap,
        inout   float3  s,
        in      float3  V,
        inout   float   t)
{
        float dt    =       1.0h / LINEAR_ITERS;

        V.z     *=      -1;
        V.xyz   /=      abs(V.z);
        V.xy    *=      SCALE;

        float   old_t   = t - dt;
        float3  uvz_old = s + V * ( - dt );
        float3  uvz_new = s;

        for (int i=0; i<BINARY_ITERS; i++)
        {
                float3 uvz_mid      =       0.5h * (uvz_old + uvz_new);
                float  mid_t        =       0.5h * (t + old_t);

                float4  qdh     =       tex2D(qdmap, uvz_mid.xy);
                float   depth   =       uvz_mid.z;
```

```
                float       factor = 1.0f;

                if ((qdh.r < qdh.g) && (qdh.r < depth) && (depth < qdh.g)) factor = 0;
                if ((qdh.b < qdh.a) && (qdh.b < depth) && (depth < qdh.a)) factor = 0;

                uvz_old   =         lerp(uvz_old, uvz_mid, factor);
                uvz_new   =         lerp(uvz_mid, uvz_new, factor);

                old_t     =         lerp(old_t, mid_t, factor);
                t         =         lerp(mid_t, t,          factor);
        }

        s = uvz_new;
}
```

qdmap – текстурный 2D-сэмплер;

V – входная переменная, вектор в пространстве текстуры из глаз наблюдателя;

s – входная и выходная переменная. Начальная и конечная точка трассировки луча, соответственно;

t – выходная переменная, параметр уравнения прямой SS`, используется для отбрасывания визуализируемых пикселей;

BINARY_ITERS – количество итераций в бинарном поиске, порядка 5–10;

SCALE – коэффициент определяющий рельефность объекта, в диапазоне [0...1].

После выполнения линейного и бинарного поиска в переменной s хранится значение текстурных координат и глубина проникновения луча под поверхность объекта.

Построение мягких теней

Эффективный приближенный метод построения мягких теней для POM был предложен в [2]. Основная идея заключается в нахождении суммы расстояний между лучом и картой высот при трассировке луча от точки пересечения с картой высот к источнику света. В данной работе метод был модифицирован и применен к случаю многослойного рельефного текстурирования.

Луч трассируется с заданным шагом от точки, найденной в ходе линейного и бинарного поиска, к источнику света. Для каждой точки определяется факт попадания в одно из геометрических тел. И, если таковое происходит, минимальное расстояние от точки до одной из ограничивающих поверхностей добавляется в общую сумму. На рис. 8 это точка A, и расстояние берется от точки A до поверхности *x*. Общая сумма домножается на весовой коэффициент и ограничивается диапазоном [0…1]. Полученное число рассматривается как коэффициент затенения.

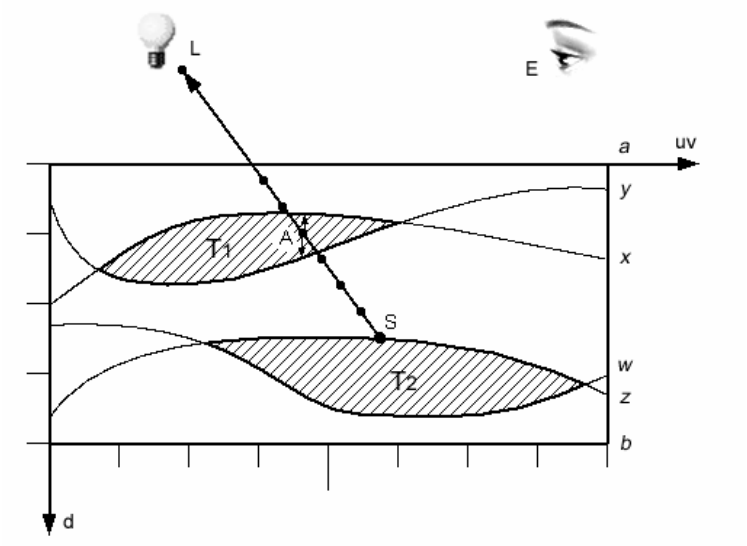


Рис. 8. Построение мягких теней

Во избежание эффекта «расслоения» тени степень затененности делается зависимой от угла между поверхностью объекта и направлением на источник света.

Ниже приводится код на HLSL функции бинарного поиска.

```
void QDRM_Shadow(
        sampler2D       qdmap,
        in              float3  s,
        in              float3  L,
        out             float   shadow)
{
        shadow =0;
        float t          =      0;
        float dt    =   1.0f / LINEAR_ITERS_SHADOW;

        float depth_bias;
        depth_bias      =       abs(L.z);
        L.z             *=      -1;
        L.xyz           /=      abs(L.z);
        L.xy            *=      SCALE;
        L.xy            *=      depth_bias;

        for (int i=0; i<LINEAR_ITERS_SHADOW; i++)
        {
                float3 uvz          =       s - L*t;

                float4  qdh             =       tex2D(qdmap, uvz.xy);
                float   depth   =       uvz.z;

                if ((qdh.r < qdh.g) && (qdh.r < depth) && (depth < qdh.g)) shadow += min(abs(depth - qdh.r),
                                                                                        abs(depth -
qdh.g));
                if ((qdh.b < qdh.a) && (qdh.b < depth) && (depth < qdh.a)) shadow += min(abs(depth - qdh.b),
                                                                                        abs(depth -
qdh.a));

                t += dt;
        }
        shadow = saturate(shadow * SHADOW_WEIGHT * depth_bias);
}
```

qdmap – текстурный 2D-сэмплер;

L – входная переменная, вектор направления источника света в пространстве текстуры;

s – входная переменная. Начальная точка трассировки луча;

LINEAR_ITERS_SHADOW – количество итераций в бинарном поиске, порядка 5–10;

SCALE – коэффициент, определяющий рельефность объекта, в диапазоне [0…1];

SHADOW_WEIGHT – резкость тени, значение – порядка 5–10;

shadow – коэффициент затенения.

Определение нормали в точке пересечения

x- и *y*- компоненты нормали хранятся, соответственно, в двух текстурах. Каждый канал текстуры соответствует одному из четырех слоев. Возникает задача определения нормали в точке пересечения, и как следствие – слоя, с которым пересекся луч зрения.

z-компонента определяется через *x*- и *y*- компоненты. Если пересечение произошло со слоями Y или W, то z-компонента домножается на –1.

Ниже приводится код HLSL, выполняющий расчет нормали:

```
float4  qd      =       tex2D(qdmap, s.xy);
float4  selector =       float4(0,0,0,0);

float4  dists   =       abs(s.z - qd);
float   sgn     =       1;

if (dists.x <= min(dists.y, min(dists.z, dists.w)) )  { selector = float4(1,0,0,0);   sgn = +1; }
if (dists.y <= min(dists.x, min(dists.z, dists.w)) )  { selector = float4(0,1,0,0);   sgn = -1; }
```

```
if (dists.z <= min(dists.x, min(dists.y, dists.w)) )  { selector = float4(0,0,1,0);   sgn = +1; }
if (dists.w <= min(dists.x, min(dists.y, dists.z)) )  { selector = float4(0,0,0,1);   sgn = -1; }

float4  nx =        tex2D(sm[2], s.xy);
float4     ny       =       tex2D(sm[3], s.xy);
float4     c        =       tex2D(sm[0], s.xy);

float3 n;
n.x        =        dot(nx, selector)*2-1;
n.y        =        dot(ny, selector)*2-1;
n.z        =        sqrt(max(0, 1.0-dot(n.xy,n.xy))); // max() используется во избежание извлечения корня
                                                      из отрицательного числа
n.z        *=       sgn;
n          =        normalize(n);
```

Отбрасывание визуализируемых пикселей

Визуализация пикселей отменяется в случае, если луч зрения не пересек ни одно из тел. Отследить такой случай можно с использованием переменной t. Если ее значение близко к 0 или 1, то пиксель отбрасывается функцией HLSL clip().

Ниже приведен код HLSL, осуществляющий отбрасывание пикселей:

```
if (t > 0.999)        clip(-1);
if (t < 0.001)        clip(-1);
```

Расчет освещения

Освещенность рассчитывается любым известным методом (в данной работе используется модель освещения Блинна). Результат домножается на коэффициент затенения.

Заключение

В ходе работы были изучен и реализован ряд известных методов рельефного текстурирования. Была решена проблема билинейной фильтрации текстур, которая была свойственна оригинальному методу. Кроме того, был применен метод приближенного построения мягких теней для случая многослойного рельефного текстурирования.

Недостатком предложенного алгоритма является его низкая производительность как следствие большого количества циклов, операций сравнения и зависимых и независимых выборок из текстуры, а также большое количество кэш-промахов как следствие нелинейной выборки значений из текстуры.

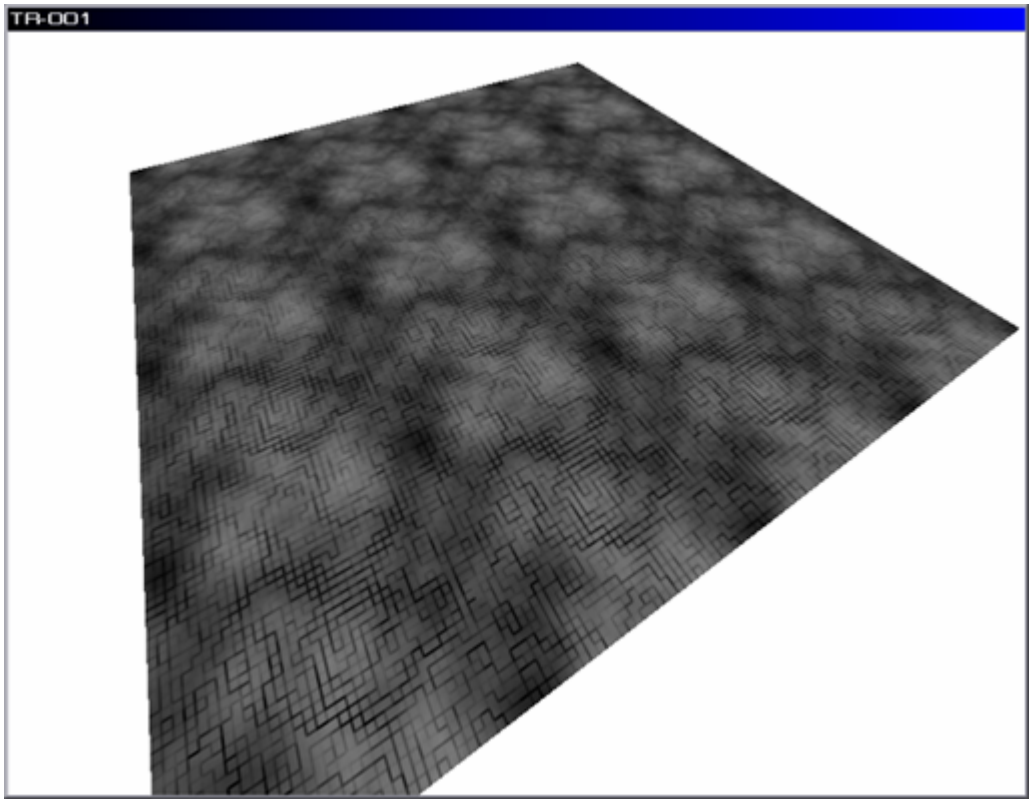
Разработанное в рамках работы приложение тестировалось на видеокартах nVidia GeForce 7800 GT и GeForce 6600 GT. Получена производительность 65–70 и 10–15 кадров в секунду соответственно. Результаты работы программы представлены в Приложении.

Данный метод может использоваться для отображения сложных поверхностей, таких как решетки, барельефы, элементы механизмов и т.д., в случае, когда геометрическое представление таких объектов нецелесообразно. Важно отметить, что производительность данного метода зависит только от размеров текстур, а не от сложности объектов, представляемых в данных текстурах.

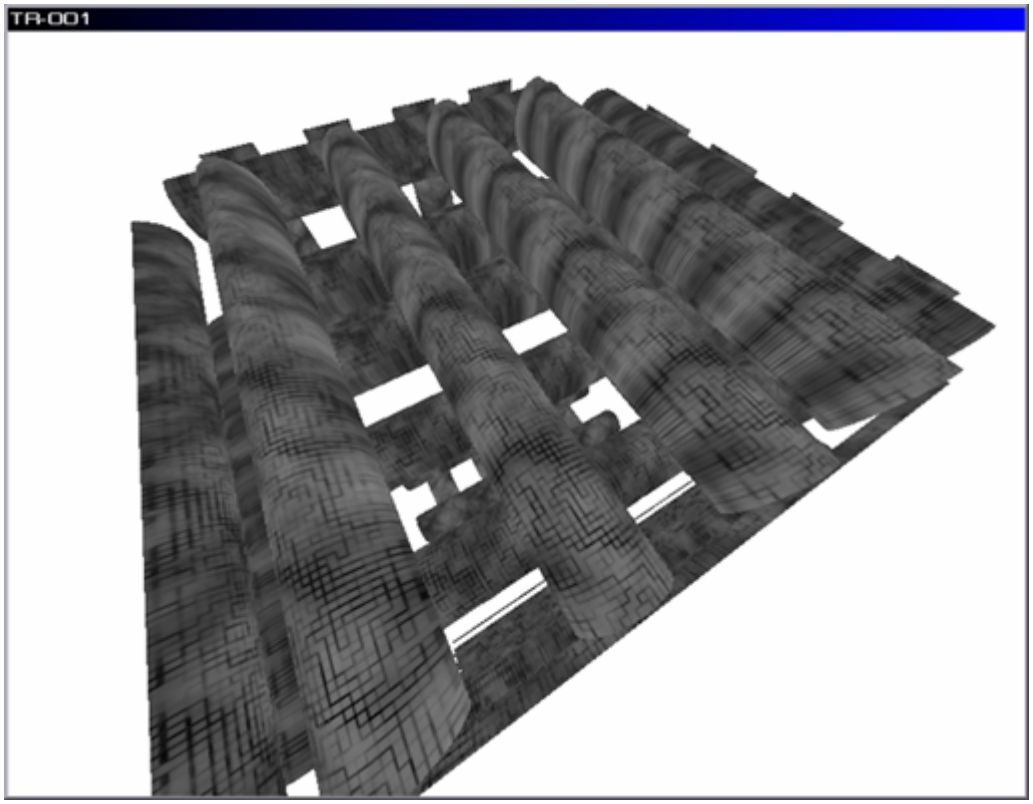
Литература

1. Oleveira M.M., Policarpo F. 2005. Relief Mapping For Non-Height-Field Surface Details.
2. N. Tatarchuk. Dynamic Parallax Occlusion Mapping with Approximate Soft Shadows. // ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games. 2006.

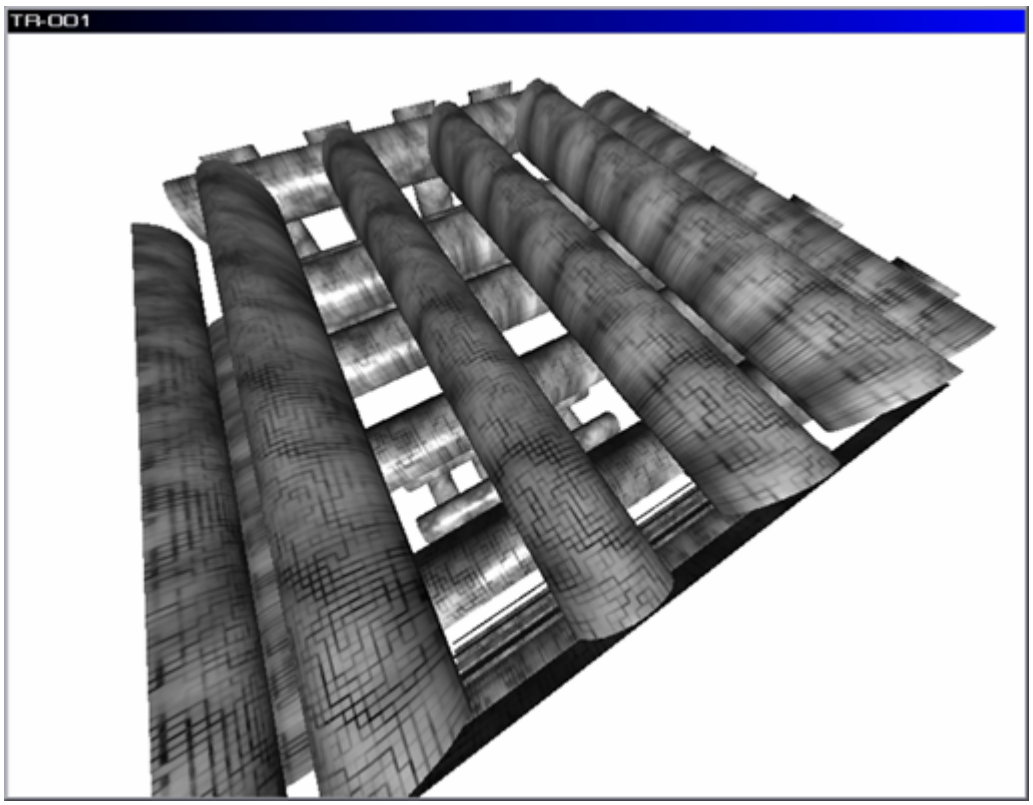
Приложение



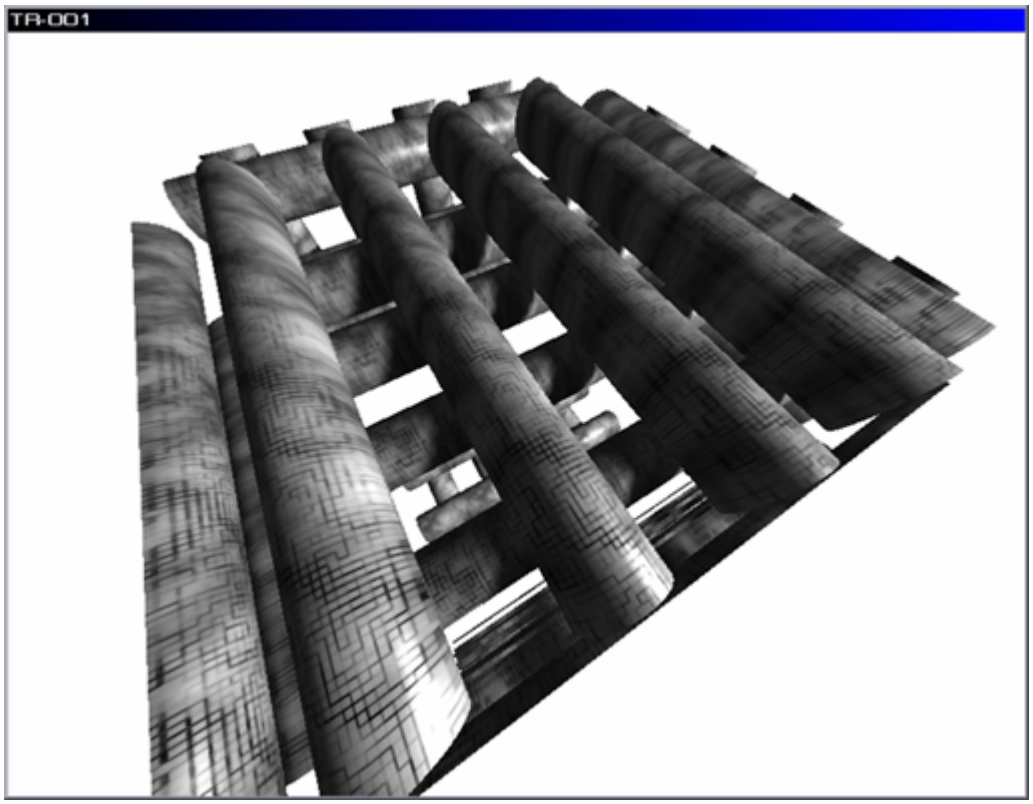
Обычное текстурирование



Многослойное рельефное текстурирование

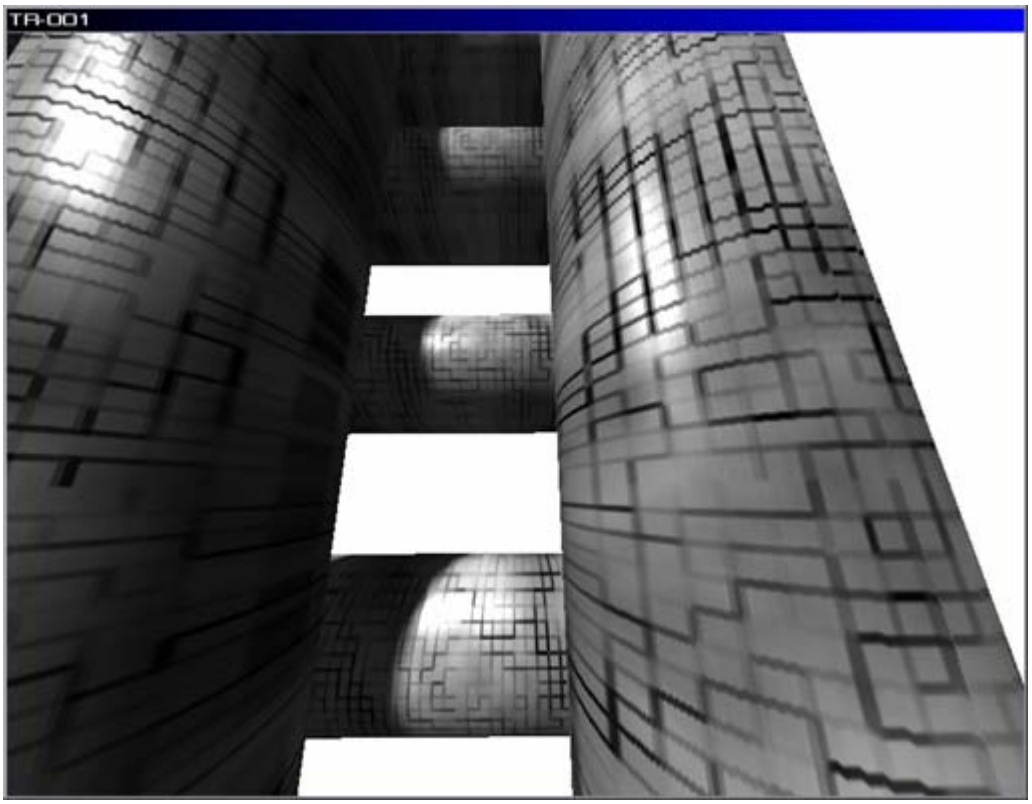


Многослойное рельефное текстурирование с освещением



Многослойное рельефное текстурирование с освещением и затенением

Вид изнутри куба



Вид вблизи от поверхности куба

ИСПОЛЬЗОВАНИЕ МОДЕЛЕЙ ДВИЖЕНИЯ В РЕШЕНИИ НАВИГАЦИОННОЙ ЗАДАЧИ ПО СИГНАЛАМ СПУТНИКОВЫХ РАДИОНАВИГАЦИОННЫХ СИСТЕМ

И.В. Бедняков
Научный руководитель – к.т.н., доцент А.Е. Платунов

В работе исследуется возможность повышения точности решения навигационной задачи по сигналам спутниковых радионавигационных систем (СРНС) за счет использования моделей движения потребителя. Под моделью движения понимается характер изменения параметров движения потребителя в определенной системе координат. Исследуется характер изменения псевдодальности до НКА системы ГЛОНАСС до объекта с примитивной моделью движения. Цель исследования – синтез алгоритма решения навигационной задачи по сигналам СРНС с учетом модели движения потребителя.

Введение

Решение навигационной задачи по сигналам СРНС основано на обработке измеренных радионавигационных параметров: задержки распространения сигнала от НКА до приемника и доплеровского смещения частоты, обусловленного высокой скоростью НКА относительно приемника сигнала. Существует несколько методов решения навигационной задачи по сигналам СРНС: дальномерный метод, псевдодальномерный метод, разностно-дальномерный метод, радиально-скоростной (доплеровский) метод, псевдорадиально-скоростной метод, разностно-радиально-скоростной метод, а также комбинированные методы [2]. Суть всех вышеперечисленных методов решения навигационной задачи сводится к одному: измеряются временные задержки распространения сигнала, а также доплеровское смещение частоты сигнала минимум от трех НКА до потребителя. Зная скорость распространения сигнала, а также реальную частоту, с которой излучает НКА, можно рассчитать относительную дальность и скорость потребителя относительно НКА. А по известным координатам и скоростям трех (четырех) НКА, а также по измеренной дальности и относительной скорости до каждого из них можно вычислить искомый вектор состояния потребителя: $\{t, x, y, z, Vx, Vy, Vz\}$. Различие методов заключается в количестве информации, известной априори, и во временных характеристиках получения результата (некоторые из методов не могут функционировать в режиме реального времени).

Точность решения навигационной задачи по сигналам СРНС обусловлена такими характеристиками, как: геометрическим фактором рабочего созвездия, т.е. степенью «разнесенности» НКА, по которым ведутся измерения, в пространстве; количеством НКА рабочего созвездия; точностью опорного генератора приемника потребителя; реализацией алгоритма решения навигационной задачи (учитывающая/не учитывающая ионосферные поправки в распространении излучаемого сигнала, явление многолучевости). Дополнительным и не менее важным источником погрешности в решении навигационной задачи является погрешность эфемеридной информации, передаваемой с НКА, т.е. координаты НКА, относительно которых рассчитываются координаты потребителя, содержат в себе ошибку. Кроме того, главным фактором, определяющим точность, является частота символов дальномерного кода, излучаемого НКА – ведь из оценки именно смещения принимаемого кода относительно эталонного рассчитывается задержка распространения навигационного сигнала. Как в системе ГЛОНАСС, так и в системе GPS реализованы два канала дальномерного кода: стандартной точности (СТ-код) и высокой точности (ВТ-код).

Проблема заключается в том, что сигнал, поступающий от передатчиков космических аппаратов спутниковых радионавигационных систем, является шумоподобным и с энергетической точки зрения очень слабым. Следящая система, выполняющая поиск и выделение полезного сигнала, для правильной его идентификации вынуждена выпол-

нять накопление входного сигнала длительностью не менее 10 мс, а в случае поиска в условиях большого количества помех – и того дольше. Т.е. радионавигационные параметры определяются с дискретностью, имеющей предел частоты. В случае отсутствия этого предела точность решения навигационной задачи была бы прямо пропорционально скорости работы вычислителя, на котором эта задача решается. Это достигается накоплением и фильтрацией входного сигнала. Однако, как было сказано ранее, в спутниковой навигации имеется ограничение, не зависящее от характеристик вычислителя, реализующего решение навигационной задачи. Это ограничение не позволяет выполнить решение навигационной задачи в режиме реального времени с точностью, превосходящей порядка 10 м по каждой из координат при работе по СТ-коду. В данном случае термин «реальное время» имеет тот смысл, что изменение параметров движения потребителя в каждый момент времени является безусловным и непредсказуемым. С этой позиции даже для физически неподвижного потребителя вычислитель, выполняющий решение навигационной задачи, выдает в каждый момент времени новое решение, результат которого отличается от всех предшествующих. Суть проведенного исследования заключается в оценке возможности накопления решений навигационной задачи и их фильтрации с целью повышения точности местоопределения для потребителей с различными моделями движения.

Модели движения

Динамика объекта характеризуется такими параметрами, как радиус-вектор $\vec{R} = \{r_x, r_y, r_z\}$ в конкретный момент времени *t*, вектор мгновенной скорости объекта $\vec{V} = \{v_x, v_y, v_z\}$ в момент *t* и вектор мгновенного ускорения $\vec{A} = \{a_x, a_y, a_z\}$. Все эти векторы разлагаются по компонентам в выбранной системе координат (СК). При работе с СРНС принято использовать Гринвичскую геоцентрическую систему координат (ГСК) и инерциальную систему координат (ИСК). В Гринвичской СК, неподвижной относительно Земли, находится потребитель; в инерциальной СК, условно неподвижной относительно Солнца, зафиксированы орбитальные плоскости спутниковой навигационной системы. Рассматривая различные модели движения потребителя, мы будем использовать ГСК. Динамика потребителя в различные моменты времени может быть различной, но существует ограниченное число возможных моделей движения:

- потребитель неподвижен ($\vec{R} = const; \vec{V} = 0; \vec{A} = 0;$);
- потребитель подвижен ($\vec{R} \neq const; \vec{V} = const; \vec{A} = 0;$);
- потребитель равноускорен ($\vec{R} \neq const; \vec{V} \neq const; \vec{A} = const;$);
- потребитель имеет неравномерное ускорение ($\vec{R} \neq const; \vec{V} \neq const; \vec{A} \neq const;$).

Однако возникает вопрос, как можно отнести динамику конкретного потребителя к одной из представленных моделей? Очевидно, измерение любого из параметров движения является относительным: для радиус-вектора измеряется смещение объекта относительно точки-эталона с известными координатами, для вектора скорости измеряется скорость изменения координат объекта относительно известного эталона. Точность подобных измерений зависит от точности известных параметров движения точки-эталона и от точности используемых измерительных приборов. Специфика СРНС – в том, что все расчеты и измерения, необходимые для определения своих параметров движения, потребитель обязан выполнить самостоятельно. Космические аппараты глобальной навигационной системы выступают в роли эталонов с известными в любой момент времени параметрами движения. Потребитель измеряет свое смещение относительно этих эталонов и рассчитывает свои параметры движения. Измерения, выпол-

няемые потребителем, как говорилось ранее, носят дискретный характер, и частота их ограничена.

Рассмотрим простейшую модель движения, в которой потребитель неподвижен (в выбранной СК), т.е. $\vec{R} = const; \vec{V} = 0; \vec{A} = 0$. Проведем следующий эксперимент: вначале определим с высокой точностью компоненты радиус-вектора потребителя в ГСК, а затем произведем оценку измеряемой дальности от потребителя до выбранных НКА системы ГЛОНАСС. Оценка будет производиться путем сравнения расчетной дальности с измеренной. Формула для расчетной дальности между потребителем и $\underline{i}$-м НКА имеет следующий вид:

$$D_i = \sqrt{(x_{НКА(i)} - x_П)^2 + (y_{НКА(i)} - y_П)^2 + (z_{НКА(i)} - z_П)^2},$$

где D_i – дальность между потребителем и i-м НКА в момент времени t, вектор $\{x_{НКА(i)}, y_{НКА(i)}, z_{НКА(i)}\}$ – вектор координат i-ого НКА в момент времени t в ГСК, вектор $\{x_П, y_П, z_П\}$ – вектор координат потребителя в момент t. Для расчетной дальности берутся координаты точки-эталона и известные координаты НКА СРНС ГЛОНАСС. Измеренная дальность получается от приемоизмерительного модуля с дискретностью 2 с.

Координаты точки-эталона рассчитываются в эксперименте путем накопления и усреднения координат, полученных путем решения навигационной задачи 16-ти канальным двухчастотным приемоизмерительным модулем по сигналам СРНС ГЛОНАСС/GPS на протяжении 30-ти минут. Т.е. фактически усреднение проводится по 900 отсчетам.

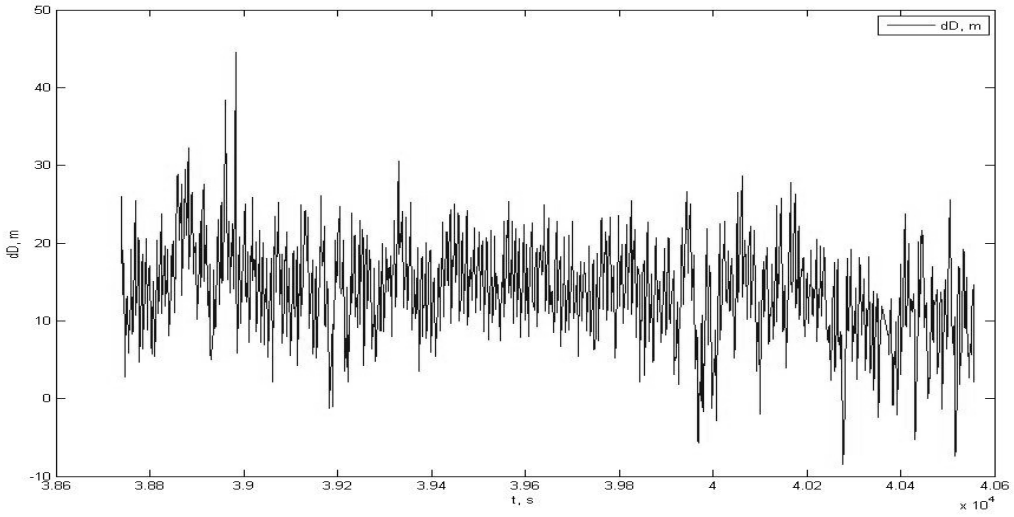


Рис. 1. Оценка измеренной псевдодальности относительно расчетной для выбранной модели движения: R=const, V=0; A=0

Здесь по оси ординат отложена разность расчетной и измеренной дальности в метрах, по оси абсцисс – момент времени измерения относительно начала текущих суток в секундах. Из рис. 1 видно, что одномоментные измерения дальности от НКА до потребителя содержат большую погрешность (порядка нескольких метров). Отклонение математического ожидания относительно нуля вызвано некоторой остаточной неточностью в определении точки-эталона потребителя. Т.е. для неподвижного потребителя можно существенно повысить точность измерения дальности путем статистической обработки некоторой выборки измерений дальности.

В проведенном эксперименте измерения производились с частотой 2 Гц. Теперь проведем оценку получаемых решений навигационной задачи (от приемоизмерительного модуля), соответствующих оцененным дальностям до НКА. Оценка производится относительно координат точки-эталона, рассчитанных путем усреднения значений 900 отсчетов.

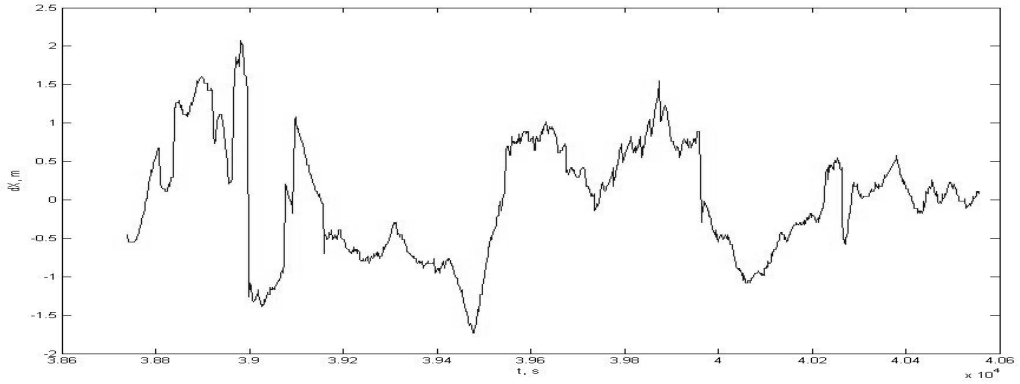


Рис. 2. График изменения координаты X при Vx=0

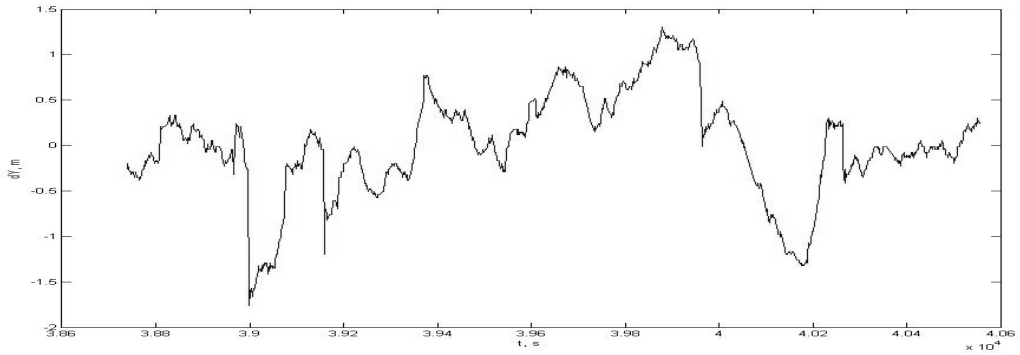


Рис. 3. График изменения координаты Y при Vy=0

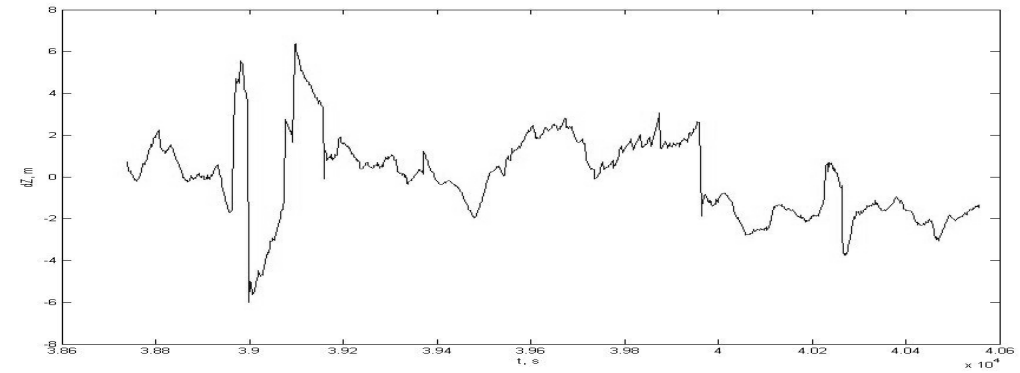


Рис. 4. График изменения координаты Z при Vz=0

Т.е., как видно из графиков, одномоментные навигационные решения несут в себе ошибку, обусловленную, очевидно, погрешностью измерения дальностей до НКА. Эту погрешность можно существенно снизить для модели неподвижного потребителя, статистически обработав выборку решений навигационной задачи за длительный период, т.е. выполнив накопление и фильтрацию измеренной дальности. Однако такая статистическая фильтрация невозможна применительно к моделям движения потребителя, в которых происходит изменение параметров движения во времени, поскольку с каждым новым измерением дальности и относительной скорости координаты и скорости потребителя изменяются.

Тем не менее, есть несколько решений этой проблемы:

- если скорость изменения параметров движения потребителя невысока, его на некотором временном интервале можно считать неподвижным и, таким образом, выполнять статистическую фильтрацию для неподвижного потребителя;

- компенсация изменения параметров движения потребителя, учитывая информацию о его модели движения.

Рассмотрим другой пример, а именно: сравним изменение расчетной и измеренной дальности между НКА и потребителем во времени:

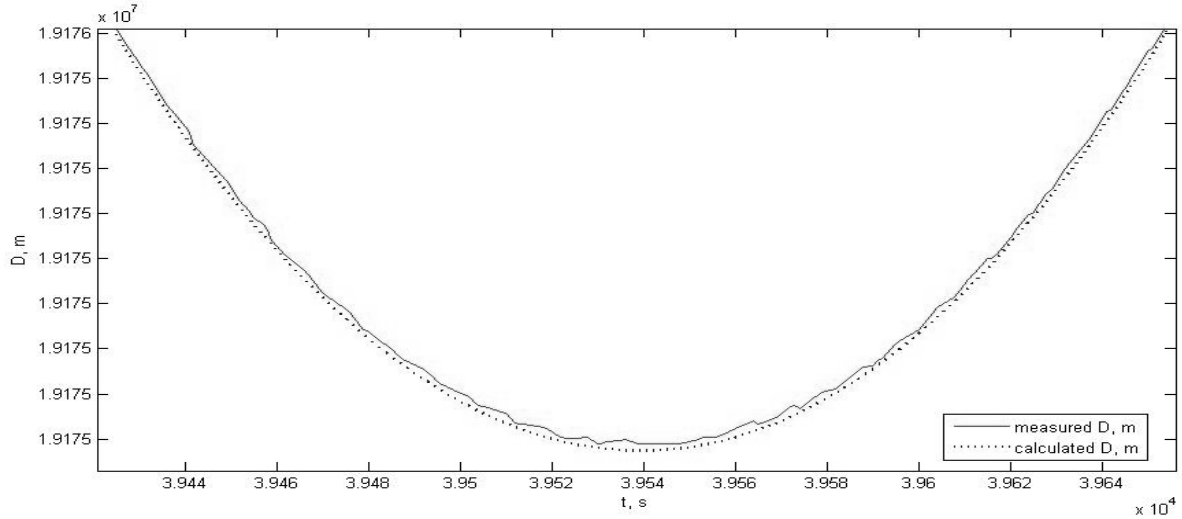


Рис. 5. Огибающая расчетной дальности, огибающая измеренной дальности между НКА и потребителем

Анализируя представленные графики для расчетной и измеренной дальности, можно сделать несколько выводов и предложений. Во-первых, огибающая расчетной дальности, как того и следовало ожидать, представляет собой более правдоподобную функцию, отражающую изменение координат среднеорбитального космического аппарата (какими являются аппараты космической группировки глобальной навигационной системы). Во-вторых, изменение дальности между НКА и потребителем во времени носит периодический характер. Два периодических процесса – движение НКА по орбите в ИСК и движение потребителя в ИСК (неподвижного в ГСК), вызванное вращением Земли вокруг своей оси – формируют периодический процесс изменения дальности.

Очевидно, что повышение точности определения дальности между НКА и потребителем связано с устранением шумовой составляющей и сглаживанием огибающей изменения дальности. Оптимальный вариант – использовать более точную расчетную дальность вместо измеренной. Однако не следует забывать, что в исследовании использована расчетная дальность, полученная на основе априори известных параметров движения потребителя. В реальной же задаче параметры движения потребителя являются искомой величиной.

Так как изменение дальности во времени для неподвижного в ГСК потребителя – некоторая периодическая функция (разложимая в ряд Фурье), можно найти коэффициенты этой функции (так как известны параметры периодических процессов, лежащих в ее основе) и на основе некоторых данных, известных априори, построить огибающую расчетной дальности между выбранным НКА и потребителем. В ситуации, когда параметры движения потребителя изменяются во времени, предлагается на следующем этапе (после построения огибающей расчетной дальности) выполнять компенсацию огибающей на основе известной информации о модели движения потребителя.

Модель движения потребителя может быть задана различными способами: это и уравнения движения потребителя (например, для низкоорбитальных космических аппаратов), и навигационные системы иного принципа действия (как частный, но широко распространенный вариант – инерциальные навигационные системы), и известным характером изменения параметров движения потребителя. Однако все это – частные случаи решения одной и той же задачи, т.е. поиск средств повышения точности решения

навигационной задачи. Важно также заметить, что даже в развивающейся в текущее время инерциально-спутниковой навигации (ИСНС) нет такого понятия, как характер движения потребителя. А ведь это очень важный момент: если навигационную задачу решать, ориентируясь на специфику модели движения потребителя, можно достичь более качественных результатов, чем при универсальном подходе. Более качественный результат достигается путем фильтрации решений и устранения шумов навигационного приемника.

Повышение точности решения навигационной задачи при использовании данных модели движения потребителя происходит нелинейно. Это связано, прежде всего, с алгоритмом решения навигационной задачи, в котором используется метод наименьших квадратов. Т.е. в итерационном процессе ищутся такие поправки к начальным оценкам искомых параметров, чтобы суммы квадратов невязок измерений (в данном случае – псеводальностей) были минимальны. В обработке большого количества измерений для удобства принято использовать матричное описание систем решаемых уравнений. Для приведения обрабатываемых матриц к безразмерному виду используют матрицу весовых коэффициентов, элементы которой рассчитываются через дисперсии погрешностей измерений [4]. Так как известная модель движения потребителя учитывается в решении как дополнительный измеренный параметр, размерность матрицы весовых коэффициентов увеличивается – добавляются элементы, отражающие точностные характеристики средств представления используемой модели движения. Кроме того, так как модель движения потребителя характеризует лишь изменение параметров движения объекта по сравнению с предыдущим решением, в элементы матрицы весовых коэффициентов по модели движения потребителя добавляются точностные характеристики предыдущего решения.

В итоге возникает задача синтеза алгоритмов комплексирования СРНС со средствами представления модели движения потребителя, как следствие – задача разработки архитектуры вычислителя, реализующего комплексный алгоритм решения навигационной задачи и создание универсального интерфейса параметров движения потребителя, который позволил бы оптимальным образом выполнить комплексирование. Здесь под универсальным интерфейсом параметров движения потребителя следует понимать совокупность величин (сигналов), характеризующих потребителя как некий динамический объект в гринвичской системе координат.

Комплексирование СРНС с инерциальными навигационными системами

Как было сказано выше, сейчас широко развивается направление комплексирования спутниковых радионавигационных систем (СРНС) и инерциальных навигационных систем (ИНС). Построение интегрированных инерциально-спутниковых навигационных систем (ИСНС) позволяет улучшить потребительские характеристики по сравнению с автономными измерителями. Используемые в ИНС датчики угловой скорости (гироскопы) и датчики ускорения (акселерометры) позволяют учитывать в расчете вектора состояния потребителя данные о его скорости и ускорении на различных этапах обработки. В статье [3] приводится пример синтеза комбинированного алгоритма комплексирования на разных уровнях в ИСНС.

Однако следует отметить, что ИНС – лишь частный случай использования известных параметров движения потребителя в решении навигационной задачи. Возможны ситуации, в которых использование комплексирования СРНС с ИНС окажется неэффективным (влечет значительные дополнительные затраты и не предоставляет существенного выигрыша в решении навигационной задачи). Ввиду этого видится целесообразным разработка универсального интерфейса между СРНС и средством представления модели движения потребителя.

Заключение

Были проведены исследования характера изменения псевдодальности до выбранных навигационных космических аппаратов глобальной спутниковой навигационной системы ГЛОНАСС для потребителя с выбранной (фиксированной в ГСК) моделью движения. Исследования производились на следующей аппаратуре: модуль приемоизмерительный, двухчастотный 16-ти канальный (СРНС GPS/ГЛОНАСС, разработка ОАО «Российский институт радионавигации и времени»), позволяющий работать с «сырыми» дальностями. Использованная модель движения позволяет продолжить исследование и для других моделей путем программной имитации параметров движения потребителя на основе данных об уже исследованной простейшей модели движения. В качестве дальнейшей цели исследования видится синтез алгоритмов комплексирования СРНС со средствами представления модели движения потребителя, как следствие – разработка архитектуры вычислителя, реализующего комплексный алгоритм решения навигационной задачи и создание универсального интерфейса параметров движения потребителя, который позволил бы оптимальным образом выполнить комплексирование.

Литература

1. Интерфейсный контрольный документ ГЛОНАСС, М., Координационный научно-информационный центр, 2002. 57 с.
2. Бакитько Р.В., Булавский Н.Т., Горев А.П. и др. ГЛОНАСС. Принципы построения и функционирования. / Под редакцией А.И. Перова, В.Н. Харисова. М.: Радиотехника, 2005. 690 с.
3. Перов А.И., Шатилов А.Ю. Методы и алгоритмы оптимальной обработки сигналов в спутниковой навигации. // Радиотехника. 2005. №7. С. 4–7.
4. Шебшаевич В.С., Дмитриев П.П., Иванцевич Н.В. и др. Сетевые спутниковые радионавигационные системы. / Под редакцией В.С. Шебшаевича. М.: Радио и связь, 1993. 414 с.

JPEG-ПОДОБНЫЙ АЛГОРИТМ СЖАТИЯ ИЗОБРАЖЕНИЙ С АДАПТИВНЫМ ВЫБОРОМ ЛОКАЛЬНЫХ ОБЛАСТЕЙ

Ю.В. Лужков

Научный руководитель – д.т.н., профессор А.Ю. Тропченко

Предложена схема сжатия изображений, использующая алгоритм адаптивной сегментации. Рассмотрены особенности ее реализации, показана эффективность применения для сжатия статических изображений. По сравнению со стандартной схемой JPEG эффективность сжатия данных повышается на 6 %.

Введение

Несмотря на наблюдающуюся в последние годы ярко выраженную моду на использование вейвлет-преобразований для сжатия изображений, ДКП по-прежнему остается актуальным инструментом, используемым в схемах сжатия, основанных на дискретных ортогональных преобразованиях. Термин «JPEG-подобная» в применении к схеме компрессии будем понимать в смысле классического варианта схемы, когда при кодировании изображение разбивается на фрагменты, которые обрабатываются по отдельности друг от друга по схеме, обязательными элементами которой являются: ортогональное преобразование, квантование преобразованных данных и их последующее статистическое кодирование.

За последние годы было предложено множество модификаций стандартной схемы JPEG. Так, в работе [1] рассматривается вопрос использования контекстного арифметического кодирования, в исследовании [2] предлагается подход, учитывающий особенности пространственно-коррелированных элементов изображения.

Суть предлагаемой нами модификации заключается в том, что изображение разбивается на фрагменты разного размера, причем разбиение является адаптивным по отношению к данному изображению. Предлагаемое решение позволяет использовать схемы статического кодирования, описанные в стандарте JPEG, без их существенной переработки.

Адаптивная сегментация изображения

Идея выбора размера блока для проведения ДКП является одной из ключевых при создании схемы эффективного JPEG-подобного сжатия. Ввиду особенностей ДКП, выбор слишком большой окрестности с последующим ее квантованием ведет к тому, что восстановленный сигнал будет сильно искажен, четкие контуры сольются с фоном, т.е. произойдет снижение качества сжатия. Напротив, слишком маленькая окрестность не позволит извлечь всей выгоды из проведения дискретного преобразования. При этом выбор фиксированного значения окна ведет к тому, что некоторые целостные объекты изображения разбиваются на части, в результате чего при восстановлении проявляются блочные артефакты.

Однако недостаток «большого окна» является несущественным, если конкретная область не содержит резких перепадов амплитуд сигнала и выраженных деталей, т.е. является однотонной.

В данной работе под однотонной областью будем понимать такую область изображения, для которой значение специального критерия однородности, вычисленное как функция от величин яркостей точек этой области, не превышает значения заданной величины – порога расщепления. В данной работе предложено несколько критериев однородности.

Однотонный участок может быть условно отнесен к фону, и дискретное преобразование, примененное к нему, будет вполне эффективным. Соответственно, детализи-

рованные места изображения, т.е. участки с резкими перепадами амплитуд сигнала, должны обрабатываться малым окном. Вышесказанное позволяет выдвинуть идею разбиения исходного рисунка на области разной размерности.

Сформулируем требования к сегментации исходного изображения.

1. Сегментация должна быть адаптивной. Это означает, что необходимо учитывать особенности локальных областей обрабатываемого изображения.
2. В процессе сегментации должны быть выделены прямоугольные блоки, которые полностью покрывают поверхность данного изображения. При этом желательно избежать ситуации, при которой некоторые блоки пересекаются.
3. Информация о разбиении блоков должна быть представлена в компактном виде.

Рассмотрим основные алгоритмы сегментации изображений, которые могут быть использованы для решения поставленной задачи.

Центроидное связывание

Данный подход описан в [3, С. 69] и представляет собой наращивание областей от выбранных заранее стартовых точек. Он требует некоторого априорного знания о расположении объектов в пределах данного изображения для задания стартовых позиций наращивания. Однако в качестве таких позиций допустимо взять несколько точек с наибольшей для данного изображения яркостью.

Не будем подробно останавливаться на описании этого алгоритма, но отметим некоторые его особенности.

Во-первых, фрагменты, полученные в результате разбиения, хотя и не пересекаются, но не являются прямоугольными областями.

Очевидно, что самое простое решение этой проблемы – вписывать каждый фрагмент в прямоугольник наименьшего размера, после чего применять к ним ДКП либо другое преобразование. При этом могут возникнуть следующие ситуации.

1. Возможно пересечение областей, вследствие чего при проведении ДКП одна и та же информация будет обрабатываться многократно. Это приведет к увеличению объема сжатого изображения.
2. Как частный случай предыдущей проблемы возможна ситуация, когда отношение площади размеченной области к области прямоугольника, в который она вписана, близко к нулю, см. рис. 1. Это также значительно снизит эффективность сжатия.

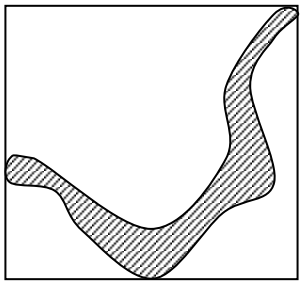


Рис. 1. Пример «невыгодной» однородной области с точки зрения ее последующей обработки

Указанные особенности алгоритма позволяют сделать вывод о его неэффективности для решения поставленной задачи.

Отметим также, что практическую реализацию алгоритма усложняют следующие его особенности:

1. необходимость определения порога слияния;
2. необходимость повторного сканирования изображения при слиянии областей.

Последовательное слияние четырех областей

Данный алгоритм относится к группе алгоритмов слияния-расщепления и предусматривает разбиение исходного изображения на непересекающиеся прямоугольные области [3, С. 72]. Приведем его краткое описание и предложим модификации данного алгоритма.

Пусть дана область D размерности $m \times m$, причем $m = 2^k$. Пусть каждая точка на начальном этапе (уровне) разбиения представляет собой самостоятельную область – $S_{ij}^0,\ i, j = \overline{1,2^k}$. Далее рассматриваются четверки смежных областей, таких, что координаты правой нижней области кратны 2^p, где p – уровень разбиения. Если модуль разницы яркостей для каждой пары из четырех областей не превосходит значения заданного порога, области объединяются в одну. Проход по изображению осуществляется до тех пор, пока не будут объединены все удовлетворяющие условию области. Крайний случай – когда исходное изображение будет представлять собой одну область. Пример разбиения приведен на рис. 2.

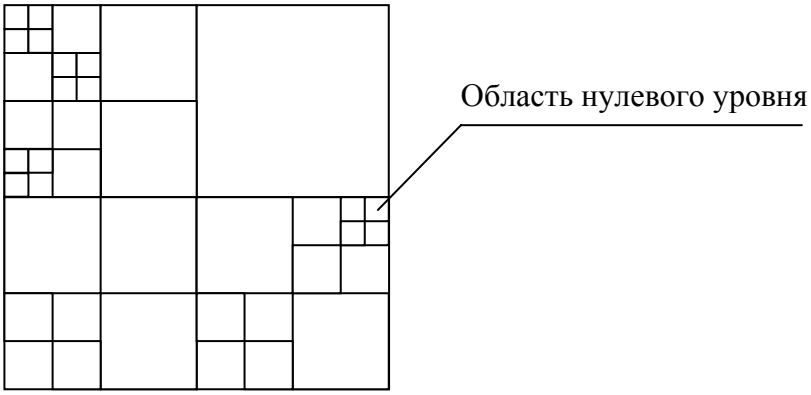


Рис. 2. Пример сегментации исходного изображения по алгоритму слияния четырех областей

К достоинствам этого алгоритма можно отнести отсутствие пересекающихся прямоугольных областей, что позволяет заострить на нем внимание.

Однако для нашей задачи критичными недостатками этого алгоритма являются:

1. ограничение на размерность исходного изображения;
2. недостаточная гибкость данного алгоритма к разбиению на области. Так, если однородный объект будет находиться в центре изображения, он может и не быть локализован одной областью.

С целью частичного устранения указанных недостатков произведем последовательную модификацию данного алгоритма.

Алгоритм половинного расщепления области

Основное отличие данного алгоритма от предыдущего заключается в следующем.

1. Исходный рисунок предварительно не делится на мелкие фрагменты, а рассматривается как целостная область, которая последовательно разбивается на более мелкие участки.
2. Разбиение осуществляется на 2 части, а не на 4. При этом оно может проводиться как по горизонтали, так и по вертикали.

Итак, изначально картинка рассматривается как единое целое, как одна большая область. Необходимо определить, является ли данная область *H* однотонной. Для этого

вычисляется значение критерия однотонности $\widehat{P}_h$ для этой области. Если оно превышает заданный порог расщепления $\widehat{P}_{\max}$, то принимается решение – разбить область по вертикали или по горизонтали. Для этого для каждой из предполагаемых половинок вертикальной и горизонтальной пары областей вычисляется значение критерия однородности $\widehat{P}_{ijh}$. Одна из двух пар с минимальной общей суммой критериев однородности выбирается как результат разбиения области.

Данная последовательность операций проводится для каждой области из числа полученных в результате разбиений (на предыдущих шагах), т.е. алгоритм является рекурсивным. Условием завершения разбиения, помимо условия удовлетворения критерию однородности, служит также и условие достижения минимального размера области: если какая-то из областей, которая может получиться после очередного разбиения, будет по своим размерам меньше минимально допустимой, то возможность разбиения в этом направлении не рассматривается.

Пример сегментации изображения методом половинного расщепления приведен на рис. 4.

В нашем случае минимальные линейные размеры области можно задать равными 8. Кроме того, допустимо также ввести ограничение и на максимальный размер области ввиду сложности проведения ДКП для матриц большой размерности.

Несмотря на явный прогресс по сравнению с предыдущим способом сегментации, можно выделить ряд недостатков алгоритма.

1. Алгоритм, в первую очередь, предназначен для локализации однотонных областей, нет гарантии, что будут локализованы отдельные объекты на изображении.
2. Если однотонный объект находится посередине рассматриваемой в данной момент области, то он может и не быть локализован.
3. Несмотря на возможность задания минимальных размерностей окрестности $x_{\min}$ и $y_{\min}$, де-факто они устанавливаются из диапазона $x_{\min} \le \hat{x}_{\min} < 2x_{\min}$ и $y_{\min} \le \hat{y}_{\min} < 2y_{\min}$, причем этой вариации нет только для изображений, исходные размерности которых удовлетворяют условиям $L_x = x_{\min} 2^k$ и $L_y = y_{\min} 2^k$, где k – целое положительное число.

Алгоритм N-кратного расщепления области

Модифицируем предыдущий алгоритм с целью повышения эффективности сегментации однородных областей. Ключевым отличием данного алгоритма от предыдущего является использование большего числа вариантов разбиения области.

При рассмотрении очередной области принимается решение, на какую из $2R$ пар разбить данную область в случае ее неоднородности, где R – число возможных линий деления по горизонтальному или вертикальному направлению. При этом предполагается, что шаг разбиения должен быть равномерным, см. рис. 3.

В качестве пары областей опять предлагается выбирать ту, чей суммарный критерий однородности минимален.

На рис. 4 представлено изображение, сегментированное по алгоритму N-кратного расщепления для $R = 7$. Как видно, при выделении границ локальные особенности изображения учитываются в большей степени, нежели при использовании алгоритма половинного расщепления, рельефные области и области с резкими границами локализуются в малых прямоугольных сегментах. В рамках проводимого исследования данный алгоритм был реализован в среде программирования Visual C++ 6.

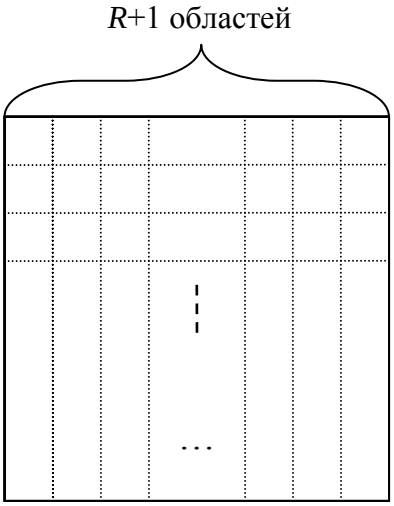


Рис. 3. Деление областей в алгоритме N-кратного расщепления

Отметим, что алгоритм *N*-кратного расщепления требует выполнения большего числа операций, чем алгоритм половинного расщепления. Это обусловлено, прежде всего, необходимостью вычислять значения критерия однородности для большего числа пар сегментов.

Рис. 4. Примеры сегментации изображений. Изображение слева сегментировано методом половинного расщепления, справа – N-кратного расщепления

К достоинствам данного подхода можно отнести гибкость локализации однородных областей за счет большего числа вариантов их деления по сравнению с ранее рассмотренными алгоритмами.

Наряду с этим достоинством можно выделить и следующие недостатки.

1. Чем больше R, тем больше значений критериев однородности необходимо вычислить.
2. Чем больше R, тем больше байт данных потребуется для сохранения результата разбиения.

Указанные недостатки позволяют сделать вывод о необходимости ограничивать значение параметра R.

Кодирование информации о сегментированной области

Простой способ сохранения координат левого верхнего и правого нижнего углов является крайне неэкономным и может свести на нет выигрыш от разбиения изображения на области. Однако можно пойти другим путем – сохранять последовательность шагов разбиения в закодированном виде. В случае алгоритма половинного расщепления для каждого шага предлагается следующий код:

1 – разбиение по вертикали;

2 – разбиение по горизонтали;

3 – завершение разбиения при условии однородности данной области.

Очевидно, что в этом случае для кодирования одного шага разбиения будет вполне достаточно двух бит информации.

Отметим также, что завершение разбиения при условии достижения минимальных границ области можно вообще не фиксировать, так как данный факт может быть предсказан на основании априорных знаний размеров исходного изображения L_x и L_y и минимальных размеров локальной окрестности $x_{\min}$ и $y_{\min}$. В этом случае даже при самом неблагоприятном исходе разбиения, когда вся исходная картинка делится на максимальное число областей, количество информации, необходимое для сохранения шагов разбиения, будет равно в битах

$N_{\text{областей}} \times 2$,

где $N_{\text{областей}}$ – максимально возможное число областей для данной картинки. Это значение, как показывает эксперимент, составляет порядка 1–3 % от общего объема файла рисунка, сжатого по стандартному алгоритму JPEG.

Конечно, при более благоприятном исходе, когда при разбиении выявляется много однотонных областей, необходимых для сохранения данных будет существенно меньше.

Для алгоритма *N*-кратного расщепления число бит, необходимое для представления одного шага разбиения, определяется из выражения

$$]\log_2 2R+1[,$$

где фигурные скобки означают округление до ближайшего большего целого числа. Единица прибавляется из-за необходимости кодировать завершение разбиения в заданном направлении, если для данной области выполняется условие однородности.

Критерии однородности локальной окрестности

Очевидно, что требуемый критерий должен адекватно учитывать как факт биения отдельных пикселей, так и факт плавного изменения амплитуд по данной области. При этом для окрестности с точками одинаковых амплитуд он должен принимать максимальное значение (или минимальное, если он инверсный).

Очевидным решением является использование модификаций способов оценки качества восстановленного изображения, которые рассмотрены, например, в [4]. Модификация заключается в том, что в качестве эталонного значения берется средняя амплитуда пикселей $\overline{A}$ данной области (размерностью $M \times N$).

Тогда можно выделить следующие способы вычисления критерия $\widehat{P}$.

1. Квадратическое отклонение: $\widehat{P} = \sqrt{\sum_{x,y}^{M,N}\left(A_{x,y} - \overline{A}\right)^2}$. Деление на число пикселей не производится (в отличие от среднеквадратического отклонения), так как иначе не будет адекватно учитываться площадь рассматриваемой области.
2. Отношение, аналогичное PSNR: $\widehat{P} = 10\lg\dfrac{255^2}{\sum_{x,y}^{M,N}\left(A_{x,y} - \overline{A}\right)^2}$.
3. Максимальное отклонение от среднего: $\widehat{P} = \max_{x,y}\left|A_{x,y} - \overline{A}\right|$.

4. Среднее отклонение от среднего значения: $\widehat{P} = \frac{1}{MN} \sum_{x,y}^{M,N} \left| A_{x,y} - \overline{A} \right|$.

Однако, помимо критериев оценки однородности, образованных от критериев оценки качества восстановленного изображения, можно использовать и другие.

5. Градиентный критерий: $\widehat{P} = \frac{1}{MN} \sum_{x,y}^{M-1,N-1} \left| A_{x+1,y+1} - A_{x,y} \right|$.

Также могут быть использованы и другие алгоритмы вычисления критерия однородности области. В рамках данной работы ставились эксперименты с использованием квадратического критерия и критерия максимального отклонения от среднего.

Очевидной проблемой является необходимость задания порога $\widehat{P}_{\max}$ – максимального значения критерия $\widehat{P}$, при котором область считается однородной. Он может быть получен в результате статистического анализа большого числа изображений, а также вычисляться на основе априорных данных о данном конкретном изображении.

Если необходимо учитывать значения амплитуд всех цветовых плоскостей изображения, то практическое вычисление, например, квадратического отклонения для цветовой модели RGB будет иметь вид:

$$\widehat{P} = \sqrt{\sum_{x,y}^{M,N} (R_{x,y} - \overline{R})^2} + \sqrt{\sum_{x,y}^{M,N} (G_{x,y} - \overline{G})^2} + \sqrt{\sum_{x,y}^{M,N} (B_{x,y} - \overline{B})^2}, \qquad (1)$$

где $R_{x,y}$, $G_{x,y}$ и $B_{x,y}$ – значения амплитуд цветовых компонент, а $\overline{R}$, $\overline{G}$ и $\overline{B}$ – их средние значения для оцениваемой окрестности. Для критерия максимального отклонения от среднего

$$\widehat{P} = \max_{x,y} \left| R_{x,y} - \overline{R} \right| + \max_{x,y} \left| G_{x,y} - \overline{G} \right| + \max_{x,y} \left| B_{x,y} - \overline{B} \right|. \qquad (2)$$

Практическое использование алгоритмов сегментации области

Стандартная схема JPEG предусматривает перевод изображения из цветового пространства RGB в YUV. При этом информация о цвете и светимости распределяется между новыми плоскостями неравномерно. Это позволяет сделать предположение о том, что каждую плоскость необходимо сегментировать независимо от других. Эксперименты показали, что при одинаковом пороге $\widehat{P}_{\max}$ число областей, получаемых в результате разбиения плоскостей цветности U или V, как правило, в несколько раз меньше, чем при сегментации плоскости светимости Y. Это говорит о том, что компоненты цветности в среднем являются более однотонными. Соответственно, при их сегментации полученные фрагменты будут в среднем большего размера, чем при сегментации по формулам (1) и (2), что в итоге даст выигрыш при сжатии.

Таким образом, мы получаем выигрыш даже в том случае, если не выполняем операции прореживания (укрупнения пикселей) хроматических компонент.

В предлагаемой реализации алгоритма сжатия коэффициенты результата дискретного преобразования масштабируются с тем расчетом, чтобы они по модулю не превышали 2^{10}. Это позволяет использовать набор предлагаемых стандартом JPEG таблиц Хаффмана при проведении этапа вторичного сжатия.

Кодировать разницу DC-коэффициентов не имеет смысла (ввиду порядка следования сегментированных участков на изображении), поэтому они обрабатываются вместе с AC-коэффициентами. В остальных аспектах этап вторичного сжатия аналогичен стандартной схеме JPEG.

Схема реализации конвейера разработанного алгоритма представлена на рис. 5.

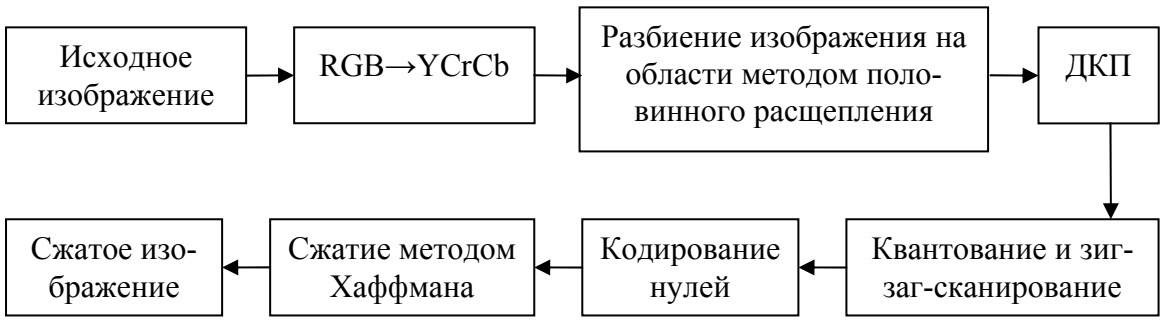


Рис. 5. Схема реализации JPEG-подобного алгоритма сжатия изображений с адаптивным выбором локальных областей

Отметим, что эффективность предлагаемой схемы выше стандартной в среднем на 5–7 %, т.е. показатель качества (PSNR) для восстановленного изображения доминирует при том же значении коэффициента сжатия K на данную величину процента. Однако такой результат не является предельным, и в дальнейшем возможно создание более эффективной схемы сжатия на основе адаптивного разбиения исходного изображения.

Заключение

Нами был рассмотрен JPEG-подобный алгоритм сжатия изображений на основе адаптивного выделения локальных областей. Ключевой особенностью предложенной в работе схемы сжатия является использование алгоритма N-кратного расщепления для сегментации рисунка. Было показано, что его использование эффективно для сжатия неподвижных изображений. Отметим, что эффективность разработанной схемы выше стандартной в среднем на 5–7 %.

К числу недостатков адаптивной схемы сжатия можно отнести ее большую вычислительную сложность по сравнению со стандартной схемой и необходимость задания некоторых параметров.

Среди возможных путей дельнейшего повышения степени сжатия можно выделить исследование эффективности применения других дискретных преобразований, а также поиск путей модификации статистического кодирования.

Литература

1. Умняшкин С.В. Использование контекстного арифметического кодирования для повышения сжатия данных по схеме JPEG. // Известия вузов. Электроника. 2001. № 3. С. 96–98.
2. Andrew B. Watson. A standard model for foveal detection of spatial contrast. // Journal of Vision. 2005. №5. P. 717–740. http://journalofvision.org/5/9/6/
3. Тропченко А.Ю., Тропченко А.А. Методы сжатия и вторичной обработки изображений. Распознавание объектов на изображении.
4. Ватолин Д., Ратушняк А., Смирнов М., Юкин В. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео. М.: ДИАЛОГ-МИФИ, 2002. 384 с.

ПРОБЛЕМЫ И ЗАДАЧИ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ РАСПРЕДЕЛЕННЫХ БАЗ ДАННЫХ

В.Б. Новосельский
Научный руководитель – к.т.н., профессор Т.А. Павловская

Базы данных (БД) занимают центральное место в автоматизированных информационно-управляющих системах. В связи с возрастающими требованиями к оперативности и достоверности информации создание распределенных информационных систем является весьма актуальной задачей. В статье описаны основные этапы проектирования распределенных БД, проведен анализ существующих систем автоматизированного проектирования (САПР) в области проектирования БД, рассмотрены перспективные направления проектирования БД. Сформулированы принципы построения эффективных САПР БД.

Введение

Базы данных занимают центральное место в автоматизированных информационно-управляющих системах (АИУС). От эффективности и качества их построения во многом зависит эффективность разрабатываемых информационных систем. Поэтому разработка систем автоматизированного проектирования БД (САПР БД) является важной и актуальной задачей.

Процесс проектирования распределенных баз данных (РБД) включает анализ предметных областей пользователей, синтез оптимальных логических структур РБД, логических и физических структур локальных баз данных (ЛБД) и сетевых каталогов, выполняемых на этапах технико-экономического обоснования проекта, разработки технического задания и технорабочего проектирования РБД [1, 2, 7]. Ключевой проблемой проектирования РБД является разработка компонентов логического уровня, включающего логические структуры РБД, локальных БД и базы метаданных (БмД) репозитория [8].

В статье описываются основные этапы проектирования РБД, рассматриваются существующие системы автоматизации проектирования БД и перспективные направления проектирования БД и формулируются принципы построения эффективных САПР БД.

Анализ предметных областей пользователей

Одним из основных этапов проектирования РБД является анализ предметных областей пользователей. Основной целью данного этапа является изучение и анализ предметных областей пользователей, определение целесообразности проектирования РБД, обеспечение формализованного описания и структуризации предметных областей пользователей. Процесс анализа включает четыре взаимосвязанных этапа: предпроектный анализ информационных множеств пользователей; анализ предметных областей и построение внешних моделей данных пользователей; построение обобщенной внешней (глобальной концептуальной) модели РБД; нормализация обобщенной внешней модели (ОВМ) и построение канонической структуры РБД [2, 7, 9]. При этом под внешней моделью пользователя понимается формализованное описание множества устойчивых характеристик и свойств данных, процедур их обработки и отношений между ними, которые в совокупности составляют предметную область. Основным требованием, предъявляемым к способам описания и структуризации ОВМ, определяющим возможность использования результатов анализа для автоматизации проектирования РБД, является уровень формализации представления предметных областей пользователей.

Актуальным направлением развития моделей данных для описания предметных областей пользователей является расширение их возможностей для отражения семантики данных. Учет семантических свойств элементов и их взаимосвязей обеспечивает

достаточно высокий уровень независимости данных, позволяет организовать взаимодействие широкого круга пользователей, а также создает предпосылки для алгоритмического описания процедур проектирования на высокоуровневых языках и автоматизации процессов согласования и объединения информационных требований пользователей, проверки полноты и совместимости описания предметных областей.

Синтез оптимальной логической структуры

Синтез оптимальной логической структуры РБД является одной из центральных проблем разработки РБД. Решаемые на этом этапе задачи отличаются большой размерностью, сложностью и трудоемкостью, что определяется распределенностью мест хранения и обработки данных; необходимостью учета большого количества характеристик используемой информации и процедур ее обработки; многообразием критериев эффективности; ограничениями СУРБД, СУБД, операционных систем и технических средств; недостаточной разработкой используемых формальных методов проектирования [2]. Большинство разработанных в настоящее время моделей синтеза логических структур РБД основывается на исследовании задач размещения баз данных или информационных массивов и программ по узлам ВС заданной конфигурации, которые решаются с использованием методов целочисленного линейного, нелинейного, квадратичного и динамического программирования. Основным задачами при этом являются определение рационального уровня избыточности информации (рациональное количество копий) и выбор эффективной пропускной способности каналов связи маршрутов передачи информации в ВС.

Основной целью формализованных моделей и методов синтеза логической структуры РБД является отображение канонической структуры РБД в логическую, которая обеспечивает экстремальное значение заданного критерия эффективности функционирования АИУС и удовлетворяет требованиям и ограничениям, отраженным в техническом задании [2, 10]. Результатом логического синтеза являются оптимальные по заданному критерию эффективности состав и структура типов логических записей; размещение их и локальных БмД репозитория по узлам ВС с оптимальной степенью дублирования, определяемой характеристиками информации и запросов пользователей, топологией, характеристиками и ограничениями вычислительной сети. Основными критериями оптимизации являются [16]:

- уменьшение объема памяти, необходимой для хранения всей информации;
- уменьшение времени поиска требуемой информации;
- уменьшение времени первоначальной загрузки информации;
- уменьшение вероятности перестраивания набора отношений при введении новых типов данных;
- освобождение набора отношений от аномалий добавления, изменения и удаления.

Физическое проектирование БД

Основной целью физического проектирования БД является выбор таких методов физической структуризации данных, при которых обеспечивается экстремум заданного критерия эффективности функционирования БД. К основным факторам, влияющим на качество физического проекта БД, относятся: обеспечение минимальной избыточности представления структур данных; эффективное использование внешней памяти для хранения больших объемов данных; обеспечение минимального трафика для РБД в архитектуре «клиент-сервер»; обеспечение минимального среднего времени доступа к данным и невысокой средней стоимости хранения информации. Результаты физического проектирования БД существенно зависят от следующих факторов: параметров логиче-

ской структуры, методов и средств хранения, доступа и обработки данных, предоставляемых используемой СУБД; ограничения на доступные объемы внешней памяти, ограничений выбранной политики безопасности и сохранности информационных, программных и технических ресурсов системы; показателей динамики использования запросами и транзакциями информационных ресурсов, а также режимов и характера обработки данных.

Процесс проектирования физических структур РБД включает в себя выбор рациональной стратегии размещения информационных массивов по узлам ВС. Такие задачи решаются с учетом затрат на хранение данных, задержек в очередях и стоимости передачи данных при ограничениях на ожидаемое время доступа к каждой БД, объем доступной внешней памяти и др. Задачи решаются с использованием эвристических алгоритмов. Исследование особенностей размещения информации в ВС с радиальной, распределенной и иерархической структурами позволило разработать модели и стратегии, обеспечивающие размещение информации в ВС с учетом особенностей топологии ВС [8].

Несмотря на многообразие моделей и методов проектирования РБД, в настоящее время отсутствует единый комплекс взаимосвязанных формализованных моделей и методов анализа и синтеза оптимальных по заданным критериям эффективности логических структур РБД, локальных БД и БмД репозитория, позволяющих автоматизировать процесс проектирования РБД на всех основных этапах их разработки. Реализация формализованных моделей и методов, пакетов прикладных программ и систем автоматизированного проектирования представлена в настоящее время ограниченным числом разработок отдельных процедур проектирования РБД [11].

Обзор существующих САПР БД

На рынке систем автоматизированного проектирования структур БД можно выделить такие средства разработки, как S-Designer от Sybase, ERWin, Oracle Designer/2000. Менее распространены отечественные CASE-средства, среди них можно выделить: ИРИС (ВНИИСИ РАН), АПРОБАЗ (Автоматизированное ПРОектирование БАЗ данных), «Дизайнер баз данных» (Университет, г. Санкт-Петербург), СИНТЕЗ+ (академия им. Жуковского), КОМОД (Институт кибернетики, г. Киев), CASE.Аналитик (предприятие «Эйтекс», г. Москва) [2, 12–14]. Системы отличаются развитыми информационными, сервисными и графическими возможностями. Однако анализ возможностей и характеристик данных систем показывает, что им присущи следующие существенные недостатки.

Во-первых, проектирование структур БД на начальных этапах (концептуальном и логическом) проводится в основном вручную: исходные данные для проектирования в виде списков сущностей предметных областей, атрибутов сущностей и связей между сущностями задаются самим проектировщиком, исходя из анализа постановки задач, его опыта и интуиции. Этот недостаток связан с ограниченностью модели «сущность–связь», используемой в данных системах, которая не позволяет отразить все сложности и особенности деятельности моделируемой предметной области.

Во-вторых, отсутствуют средства оптимизации логических и физических структур БД по различным эксплутационным критериям эффективности функционирования информационных систем. Сами по себе процедуры нормализации обеспечивают эффективность структур БД только с точки зрения их администрирования (поддержания целостности и непротиворечивости БД), но не эксплуатации БД пользователями системы. Отсутствуют также средства оптимизации размещения БД на устройствах внешней памяти, а также размещения РБД по узлам вычислительной сети.

В-третьих, данные системы ориентированы на проектирование только реляционных структур локальных БД и не содержат методов и средств проектирования СБД в архитектуре «клиент-сервер», РБД и ООБД. Этот фактор ограничивает область их применения в современных условиях. Кроме того, даже при проектировании реляционных БД используется достаточно устаревшая методология Дж. Мартина, базирующаяся на двухэтапном подходе. Современная методология использует трехэтапный подход с выделением в качестве отдельного и важного этапа логического проектирования.

В-четвертых, данные системы не содержат методов оценки качества выработанных проектных решений. За качество сформированной структуры РБД отвечает сам проектировщик. Отсутствуют средства оценки генерируемых структур БД при выполнении информационно-поисковых процессов – обслуживания запросов пользователей и транзакций, т.е. формируемые структуры БД не отражают динамических характеристик предметных областей. Данные системы совершенно не гарантируют приемлемых эксплутационных характеристик БД, так как отсутствуют средства расчета и сравнения временных, стоимостных и объемных характеристик для разных вариантов структур БД.

В-пятых, данные системы не содержат методов и средств проектирования структур БД по критериям достоверности и защиты структур данных от несанкционированного доступа. Самое большое, что они предоставляют – это средства контроля целостности данных.

Проведенный анализ CASE-средств в области автоматизации проектирования БД показывает, что они обеспечивают в основном информационную поддержку процесса проектирования, облегчают рутинную работу проектировщиков по формированию структурных и графических диаграмм и документированию описаний структур БД, автоматизируют описания структур БД на языке описания данных выбранной СУБД. Наиболее продвинутые из них имеют удобный визуальный интерфейс разработки, развитые сервисные средства, а также средства генерации программного кода приложений. Однако функциональные возможности данных систем зачастую ограничены и не охватывают всех важных задач проектирования, создания и эксплуатации БД. При этом практически отсутствуют средства автоматизации разработки РБД и ООБД; не обеспечивается комплексное решение задач проектирования БД и получение оптимальных проектных решений. Процесс проектирования практически проводится разработчиком вручную с использованием его знаний и практического опыта.

Перспективные направления проектирования БД

Рассмотрим подробнее объектно-ориентированную модель данных (ООМД) и в целом объектно-ориентированный подход к построению БД, являющиеся в настоящее время наиболее перспективными направлениями проектирования БД. Возникновение направления ООБД определилось потребностями практики – необходимостью сложных типов данных: текста, графики, данных, аудио и видеоинформации. ООБД тесно связаны с развивающимися объектно-ориентированными языками и системами программирования.

В ООБД информация хранится в форме объектов. Полностью объектно-ориентированная СУБД должна обеспечивать также объектно-ориентированный интерфейс взаимодействия с пользователем. В основе ООБД лежит понятие объекта. Объектно-ориентированные БД характеризуются свойствами инкапсуляции, наследования и полиморфизма [6, 17].

Объектно-ориентированная технология призвана устранить ограничения, присущие реляционной технологии проектирования БД, и предоставить разработчикам более

естественные и совершенные средства моделирования предметной области. Перечислим основные принципы ООМД.

1. Классификация – объекты, обладающие одинаковыми свойствами и поведением, могут рассматриваться как члены одного класса. Таким образом, индивидуальный объект может рассматриваться как частный случай общего понятия.
2. Подклассы и суперклассы – экземпляры некоторого класса могут образовывать подмножество другого класса.
3. Подклассы наследуют атрибуты и поведение своих суперклассов.
4. Наследование атрибутов и поведения позволяет построить иерархию классов: суперкласс, обладающий общими для ряда классов атрибутами, порождает ряд подклассов, которые, наследуя атрибуты своего класса-родителя, добавляют к ним ряд атрибутов, определяющих их собственные свойства. Этот механизм практически реализует концепции обобщения и конкретизации.
5. Агрегирование позволяет создать сложные объекты из объектов-компонентов, определять отношение типа «часть-целое».

В объектно-ориентированных СУБД технология объектов охватывает и концептуальную, и логическую стадии проектирования. При этом механизмы моделирования данных находятся в распоряжении разработчиков вплоть до этапа конкретной реализации модели данных во внешней памяти.

Следует отметить, что, несмотря на достаточно большое количество публикаций в области ООБД и наличие нескольких коммерческих систем БД и ООСУБД, в настоящее время отсутствуют единые взгляды и стандарты на ООМД. В общем виде ООМД должна сочетать в себе лучшие черты ER-модели и реляционной модели.

ООБД по сравнению с реляционными БД обеспечивают следующие преимущества:

- в таких БД хранятся не только данные, но и методы их обработки, инкапсулированные в одном объекте;
- ООБД позволяют обрабатывать мультимедийные данные;
- ООБД допускают работу на высоком уровне абстракции;
- ООБД позволяют пользователям создавать структуры данных любой сложности.

Объектно-ориентированный подход к проектированию БД и в целом АИУС является альтернативным широко используемым структурным методам.

Принципы построения эффективных САПР БД

В современных условиях глобализации бизнеса, требующего резкого снижения затрат на выполнение производственных функций, мобильности персонала, возможности работы в любой точке мира, к методам и CASE-технологиям проектирования и эксплуатации БД должны предъявляться следующие важные требования.

1. Использование компонентной (модульной) технологии проектирования БД, которая обеспечивает параллельное спиральное проектирование структурных компонентов БД с их дальнейшим комплексированием по мере необходимости. Данная технология, в отличие от существующей в настоящее время каскадной структурной схемы проектирования, обеспечивает большую гибкость и открытость системы БД (переносимость, интероперабельность, масштабируемость).
2. Использование технологии корпоративных хранилищ данных (Data Warehouse) в качестве составной части интегрированных БД для хранения исторической информации организаций. Требуется разработка методов и средств анализа и синтеза оптимальных структур хранилищ данных, процедур администрирования такого рода БД (извлечение данных и операционных БД, «очистки» и размещения данных в

хранилище, поиска данных и др. операций). БД хранилищ должны строиться на объектно-ориентированных моделях данных.

3. Применение нескольких моделей данных для описания предметных областей пользователей. Как показала практика, использование только какой-либо одной модели (например, популярной ER-модели) накладывает жесткие ограничения на реальную структуру предметной области, что не позволяет учесть динамические характеристики и специфические особенности объектов и процессов предметной области.
4. Применение методов оптимизации проектирования структур БД по эксплутационным критериям эффективности, в том числе для ЛБД и РБД, методов оценки качества проектных решений. Развитие моделей управления запросами и транзакциями для повышения эффективности функционирования систем. В связи с ростом требований к надежности и безопасности информации, особенно для БД класса VLDB, необходимо применение моделей и методов анализа и синтеза структур БД по критериям правильности содержимого (достоверности) и защиты данных.
5. Развитие методологии объектно-ориентированного анализа предметных областей пользователей, объектно-ориентированного проектирования и программирования БД.
6. Развитие средств репозитория CASE: расширение функциональных возможностей для управления метаданными распределенных БД, решение задач оптимального размещения БмД репозитория по узлам ВС корпоративной информационной системы; поддержание эволюции структуры данных и метаданных; создание систем управления локальными и распределенными репозиториями, подобных СУБД и СУРБД; поддержка параллельного проектирования компонентов независимыми разработчиками.

Таким образом, в основе современной САПР БД должна лежать методология оптимального анализа и синтеза локальных и распределенных БД и модульного прикладного программного обеспечения приложений, включающая:

- формализованные модели и методы спецификации и анализа информационных требований пользователей, кластерного анализа информации и структуризации предметных областей пользователей локальных и распределенных БД, построения оптимальных канонических (концептуальных) структур локальных и распределенных БД;
- объектно-ориентированные методы анализа и структуризации предметных областей, формирования концептуальных структур базы метаданных репозитория системы;
- модели и методы синтеза оптимальных по различным критериям эффективности логических структур локальных и распределенных БД, структур БмД репозитория и ее размещения по узлам вычислительной сети, а также формирования оптимальных спецификаций и путей доступов запросов и транзакций;
- модели и методы синтеза оптимальных физических структур локальных и распределенных БД, спецификаций модулей прикладного программного обеспечения приложений, транзакций и SQL-запросов;
- модели и методы обеспечения целостности, достоверности и информационной безопасности БД;
- модели и методы локального и распределенного управления сопровождением и развитием БД.

Заключение

В статье описаны основные этапы проектирования локальных и распределенных БД, проведен анализ зарубежных и отечественных CASE-средств в области автомати-

зации проектирования БД, рассмотрены перспективные направления проектирования БД. Сформулированы принципы построения эффективных САПР баз данных.

Литература

1. Кириллов В.В. Основы проектирования реляционных баз данных, СПб: ИТМО, 1994. 90 с.
2. Кульба В.В., Ковалевский С.С., Косяченко С.А., Сиротюк В.О. Теоретические основы проектирования оптимальных структур распределенных баз данных. Серия: Информатизация России на пороге XXI века. М.: Синтег, 1999. 660 с.
3. Хаббард Дж. Автоматизированное проектирование баз данных. М.: Мир, 1984. 296 с.
4. Швецов В.И., Визгунов А.Н., Мееров И.Б. Базы данных. Нижний Новгород. Издательство Нижегородского госуниверситета, 2004. 271 с.
5. Дейт К.Дж. Введение в системы баз данных: Пер. с англ. 6-е изд. К.: Диалектика, 1998. 784 с.
6. Харрингтон Дж. Проектирование объектно-ориентированных баз данных. М.: ДМК Пресс, 2001. 269 с.
7. Тамер Оззу М., Валдуриз П. Распределенные и параллельные базы данных. // СУБД, 1996. №4. С. 4–26.
8. Мамиконов А.Г., Кульба В.В., Косяченко С.А., Ужастов И.А. Оптимизация структур распределенных баз данных в АСУ. М: Наука, 1990.
9. Мартин Дж. Вычислительные сети и распределенная обработка данных / пер. с англ. Вып. 1, 2. М: Финансы и статистика, 1985, 1986.
10. Сиротюк В.О. Модели и методы синтеза оптимальных логических структур и базы метаданных репозитария распределенных баз данных в АСУ. // Автоматика и телемеханика. 1999. №1. С. 166–179.
11. Шаймарданов Р.Б. Моделирование и автоматизация проектирования структур баз данных. М.: Радио и связь, 1984.
12. Кабаков Ю.Б., Медведева А.И., Фурман Г.И. КОМОД-91 – система поддержки концептуальных схем и гипертекстов. // УсиМ. 1991. №7.
13. Алтухова А.Н., Чумаков Ю.С. Словарь проекта – CASE – средство для аналитиков и проектировщиков баз данных. / Семинар «CASE-технология». Сб. материалов. М.: 1992.
14. CASE. Аналитик для IBM PC. Руководство аналитика. М.: Эйтекс, 1993.
15. Кульба В.В., Косяченко С.А., Ужастов И.А. Задачи проектирования распределенных баз данных. М.:НИИТЭХИМ, 1985. 190 с.
16. Бабанова Н.И. Разработка и оптимизация моделей и алгоритмов автоматизированного проектирования локальных и распределенных баз данных: Автореф. дис., Владикавказ, 2000. С. 8–15.
17. Багуи С. Объектно-ориентированные базы данных: достижения и проблемы. // Открытые системы. 2004. №3.

ПРОБЛЕМЫ ИНТЕГРАЦИИ PDM-И CAD-ИСТЕМ. УНИФИЦИРОВАННЫЙ ПОДХОД

Т.Д. Голицына
Научный руководитель – к.т.н., профессор Т.А. Павловская

Статья посвящена вопросам интеграции систем управления данными об изделии (PDM) и систем автоматизированного проектирования (CAD) в машиностроении. Рассматривается существующий подход к решению этой задачи. Предлагается архитектура программного комплекса, приводятся результаты реализации его прототипа.

Введение

По данным экспертов [1], четверть совокупного бюджета сферы информационных технологий современной России приходится на долю машиностроения, при этом качество автоматизации данной отрасли остается одним из самых низких. К настоящему времени разработано достаточно большое число систем, позволяющих автоматизировать работу предприятия на различных участках производства: системы автоматизации проектирования, системы хранения и версионного контроля данных, автоматизированной технологической подготовки производства, системы управления производством и др. При этом основной проблемой является так называемая «лоскутная автоматизация» предприятий, т.е. разрозненность систем автоматизации и их несовместимость.

Связывание систем автоматизации в единое целое, создание единого информационного пространства предприятия или группы предприятий представляется наиболее актуальной задачей, поскольку это позволит избежать избыточности данных, хранимых в каждой из систем, уменьшить временной цикл их передачи между подразделениями, а также обеспечить непротиворечивость и своевременное обновление данных.

Для машиностроения интеграция PDM-систем и CAD-систем представляет наибольший интерес, поскольку обеспечивает автоматизацию конструкторской деятельности, хранения и управления информацией об изделии и связанных с ним процессами, т.е. основных направлений деятельности при подготовке производства. К этому «ядру» впоследствии возможно подключение других информационных систем предприятия.

PDM-система [2] (англ. Product Data Management – система управления данными об изделии) – организационно-техническая система, обеспечивающая управление всей информацией об изделии и связанных с ним процессах на протяжении всего его жизненного цикла, начиная с проектирования и производства до снятия с эксплуатации. PDM-системы (TeamCenter, ЛОЦМАН PDM, Lotsia PLM, PDM Step Suite и др.) включают в себя управление хранением данных (в том числе и графическими изображениями изделия) и документами, управление конфигурацией изделия, автоматизацией создания выборок и отчетов, а также механизмы авторизации. Основным принципом хранения данных в PDM-системе является то, что любые данные хранятся только один раз (без избыточности) в защищенной системе, называемой хранилищем данных.

CAD-система [3] (англ. Computer-Aided Design – система автоматизированного проектирования) – организационно-техническая система, предназначенная для выполнения проектной деятельности и позволяющая создавать конструкторскую и технологическую документацию. CAD-системы (CATIA, SolidWorks, Pro/ENGINEER, PCAD, КОМПАС 3D и др.) охватывают создание геометрических моделей изделия (трехмерных, составных), а также генерацию чертежей изделия и их сопровождение.

Традиционный подход

До настоящего времени интегрирование PDM- и CAD-систем осуществлялось для каждого конкретного случая по принципу «один на один»: конкретная PDM-система и

конкретная CAD-система взаимодействовали в рамках решения конкретной задачи через API (англ. Application Programming Interface – прикладные программные интерфейсы) обеих систем. Принципиальная схема такого взаимодействия представлена на рис. 1.

Рис. 1. Схема взаимодействия PDM и CAD систем через API

Данный способ взаимодействия обладает следующими недостатками.

(1) Ограничено применение разработанного программного обеспечения для интеграции CAD-системы и PDM-системы, поскольку это решение, как правило, специфично для решаемой задачи.
(2) Для того чтобы заменить по каким-либо причинам одну из систем, потребуется полная переработка программного обеспечения.
(3) В случае необходимости использования нескольких систем одновременно необходима существенная модификация программного обеспечения.
(4) Изменение API какой-либо из систем влечет за собой необходимость внесения изменений в прикладное программное обеспечение.

Основные преимущества этой схемы:

(5) относительная простота реализации;
(6) относительно низкая стоимость реализации.

Таким образом, вариант интегрирования PDM-системы и CAD-системы через API напрямую, хотя и позволяет создать ядро единого информационного пространства предприятия, но делает его легко уязвимым для любых изменений: от изменений в функциях API до замены систем.

Унифицированный подход

Альтернативным для подхода к интеграции «один на один» является унифицированный подход, позволяющий сделать взаимодействие PDM-систем и CAD-систем единообразным и легко расширяемым. Общая схема предлагаемой архитектуры программного комплекса представлена на рис. 2.

Приведенная архитектура изначально нацелена на расширяемость программного комплекса. Это предоставляет сразу несколько возможностей:

- использование нескольких PDM-систем и/или CAD-систем одновременно;
- замена одной системы на другую;
- применение одного программного комплекса для нескольких предприятий.

Структура программного комплекса. Как видно из рисунка, программное обеспечение, обеспечивающее интеграцию, распадается на три части:

(1) интерфейсный компонент для подключения PDM-систем;
(2) интерфейсный компонент для подключения CAD-систем;
(3) центральный модуль, отвечающий за передачу данных между системами.

Такое разделение позволяет отделять особенности реализации конкретных систем от алгоритмов интеграции этих систем.

Интерфейсные компоненты должны реализовывать следующие функции:

(1) организация взаимодействия с широким набором PDM-систем/CAD-систем;
(2) локализация информации об особенностях конкретной системы;
(3) поддержка единого набора функций API для доступа к PDM-системам и CAD-системам.

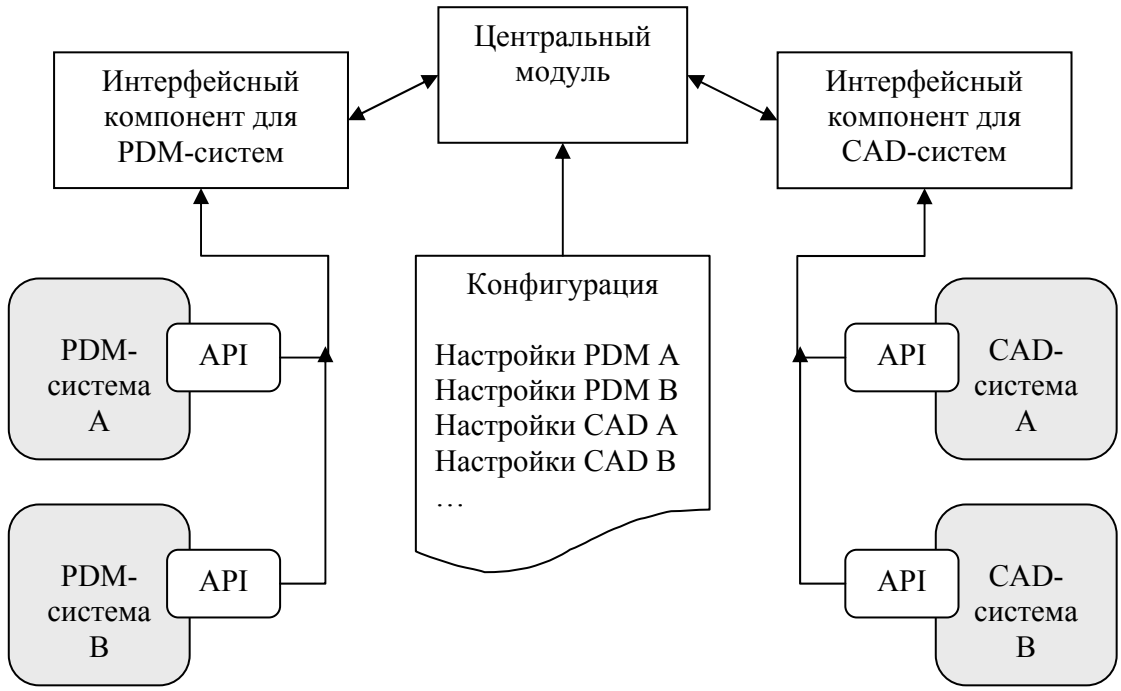


Рис. 2. Архитектура программного комплекса

Центральный модуль решает следующие задачи.

(1) Организация сеанса взаимодействия между конкретными PDM-системой и CAD-системой. Это подразумевает авторизацию и подключение к обеим системам с соответствующими правами.
(2) Обмен данными между PDM-системой и CAD-системой. Так, модель изделия из CAD-системы должна преобразоваться в иерархическую структуру изделия, его составляющих и их параметров в PDM-системе. И, наоборот, структура изделия в PDM-системе, имеющая описание геометрии каждой компоненты изделия и их взаимного расположения, должна передаваться в CAD-систему.
(3) Синхронизация данных между моделью изделия в CAD-системе и его параметрами в PDM-системе. При синхронизации необходимо указать направление синхронизации, т.е. систему, которая содержит наиболее актуальную информацию об изделии. Результаты сравнения данных, находящихся в каждой из систем, должны визуализироваться, чтобы пользователь мог понять, что изменилось в параметрах изделия в результате обновления.
(4) Обеспечение обработки запросов на модификацию с резервацией (check-in/check-out) для поддержки одновременной работы нескольких пользователей. Так, параметры изделия в PDM-системе не должны быть доступны для изменения, если модель этого изделия в этот момент модифицируется в CAD-системе. И наоборот, модификация модели должна быть запрещена на то время, пока идет модификация соответствующего изделия в PDM-системе.

Преимущества подхода. Предлагаемая архитектура программного комплекса, отвечающего за интеграцию PDM-систем и CAD-систем, лишена недостатков подхода «один на один». В качестве преимуществ предлагаемой архитектуры можно отметить следующие:

(1) высокая степень универсальности предлагаемого решения, его применимость для многих задач;
(2) простота расширения программного комплекса другими PDM-системами и/или CAD-системами. При этом изменения, вносимые в центральный модуль для поддержки новой функциональности, минимальны;
(3) унификация взаимодействия с PDM-системами и CAD-системами;
(4) относительная независимость центрального модуля от изменений, вносимых в API PDM-систем или CAD-систем;
(5) как результат, представленная архитектура является экономически выгодной: добавление новых систем производится по уже имеющемуся «образцу», а внедрение и поддержка программного комплекса – по аналогичным сценариям.

Результаты прототипирования. Для апробации предложенной архитектуры по заказу Tree Tronix Technology [4] был разработан прототип программного комплекса. Следуя предложенной архитектуре (рис. 2), он реализует взаимодействие PDM Step Suite [2] со следующими CAD-системами: PRO/ENGINEER, CATIA, Inventor, Solid Works, Solid Edge, Компас, PCad.

В качестве интерфейсного компонента для CAD-систем использовалась библиотека GSCADLink [5] компании «Глосис-Сервис», которая обеспечивает единый интерфейс доступа ко всем перечисленным CAD-системам. Интерфейсный компонент для PDM-систем был разработан с использованием API PDM Step Suite. Разработка интерфейсного компонента для PDM-систем и центрального модуля велась на Visual C++ 6.0.

Реализация прототипа программного комплекса выявила следующие проблемы.

(1) Сложность сбора требований. Поскольку внедрение программного комплекса предполагается на нескольких предприятиях, то сбор и обработка требований представляет собой достаточно сложную задачу по сведению требований нескольких заказчиков в единую систему.
(2) Сложность разработки методологии при создании модели изделия в CAD-системе, которая решала бы две задачи: учет уже существующих в PDM-системе данных о разрабатываемом изделии и обеспечение корректной передачи данных о созданном/измененном изделии в PDM-систему.
(3) Разработка такой внутренней структуры данных, которая могла бы адекватно представлять информацию о продукте и при этом позволять импортировать в неё данные из различных PDM-систем и CAD-систем.
(4) Интерпретация данных, полученных в результате выгрузки или синхронизации информации: поскольку предлагаемая схема интеграции предполагает использование различных PDM-систем и различных CAD-систем, то необходимо обеспечить корректную передачу данных из одной системы в другую.

Последние две проблемы проиллюстрированы на рис. 3.

Вследствие принципиального различия между PDM-системой и CAD-системой, а также между двумя PDM-системами или двумя CAD-системами, наименование одного и того же по смыслу параметра в различных системах может отличаться. Более того, PDM-системы и CAD-системы отличаются различной структурой хранения информации об изделии и его составе.

В качестве основного стандарта для реализации внутренних структур данных предполагается использовать ISO 10303 (ГОСТ Р ИСО 10303 [6]), поскольку он был разработан специально для интеграции систем автоматизированного проектирования, и к настоящему времени большинство PDM-систем его поддерживают.

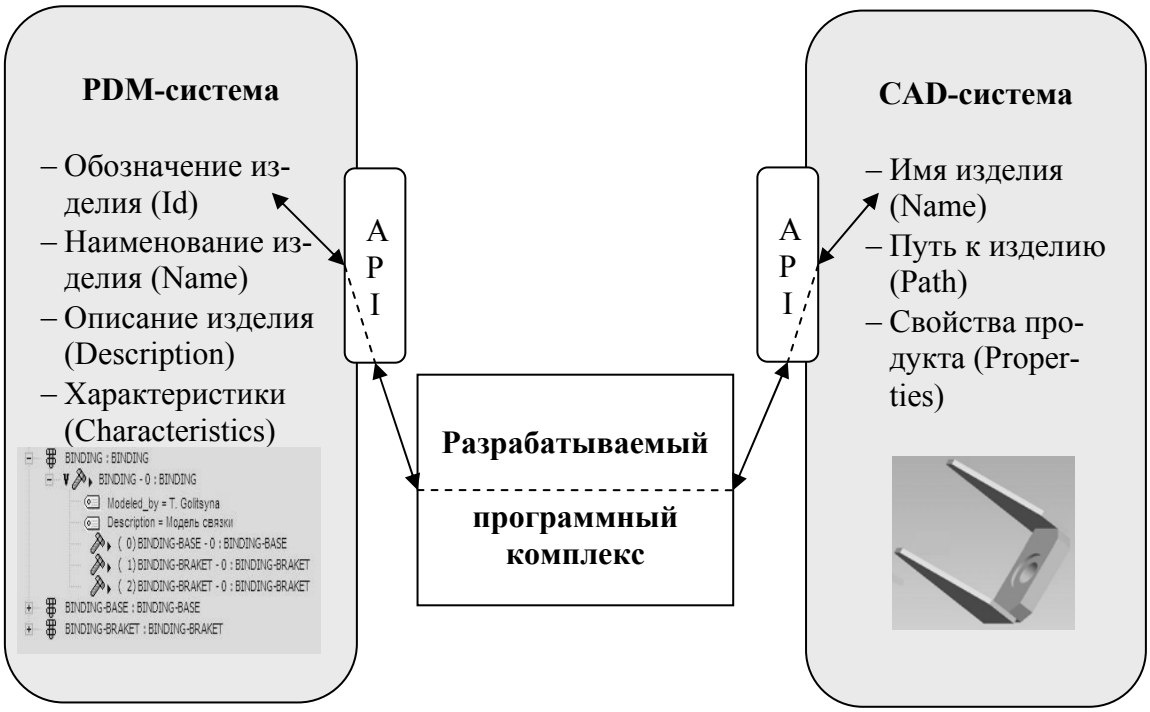


Рис. 3. Схема преобразования информации об изделии при синхронизации PDM-системы и CAD-системы

Минимально необходимо корректно интерпретировать обозначение и авторство изделия (для предотвращения ошибок идентификации), его размеры и массу (для предотвращения ошибок в расчетах) и иерархическую структуру изделия, причем для каждого составляющего изделия интерпретация его параметров также должна быть корректной.

Для решения этой задачи предполагается использовать настраиваемый конфигурационный файл, позволяющий определить, как интерпретировать тот или иной параметр в каждой из систем.

Заключение

В настоящей работе были рассмотрены два подхода к интеграции PDM-систем и CAD-систем в машиностроении, оценены преимущества и недостатки каждого из подходов. Разработана архитектура программного комплекса, реализующего унифицированный подход, приводятся результаты реализации его прототипа.

Литература

1. Чурсина М. Лоскутное одеяло машиностроителей не согреет. // Уральский рынок металлов. 2004. №9. С. 24.
2. PDM Step Suite. Техническое описание // http://pss.cals.ru/DOC/PSS_TD_2_1.pdf
3. Ли К. Основы САПР (CAD/CAM/CAE) – СПб.: Питер, 2004. 560 с.
4. http://www.tronix.ru/
5. http://www.glosys.ru/products/cad/GSCADLink.htm
6. ГОСТ Р ИСО 10303 «Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными». М.: Госстандарт России, 2000.

АЛГОРИТМЫ УКЛАДКИ ДИАГРАММ СОСТОЯНИЙ

М.А. Коротков
Научный руководитель – к.ф.-м.н., ст.н.с. Ф.А. Новиков
(Санкт-Петербургский государственный политехнический университет)

В статье приведен обзор методов плоской укладки графов, поставлена задача плоской укладки диаграмм состояний *UML*. Описан аналитический алгоритм QMATH-STATECHART.

Введение

Программный пакет с открытым исходным кодом *UniMod* [1] обеспечивает разработку и выполнение автоматно-ориентированных программ. Он позволяет создавать и редактировать диаграммы классов и состояний *UML* [2], которые соответствуют графу переходов и схеме связей конечного автомата [3]. После создания диаграмм существует возможность выполнить их. При этом содержимое диаграмм преобразуется в *XML*-описание, которое передается интерпретатору, также входящему в пакет *UniMod*. Разработанный пакет базируется на парадигме автоматного программирования [4].

Во многих случаях возникает потребность автоматической укладки диаграмм. В частности, в первых версиях *UniMod* отсутствовал графический редактор, поэтому при загрузке сохраненных файлов предыдущих версий необходимо обеспечить автоматическую укладку. В рамках проекта *UniMod* основной интерес представляет укладка диаграмм состояний языка *UML*.

При укладке диаграммы необходимо учитывать ряд критериев, которые могут противоречить друг другу, например, минимизация площади, занимаемой диаграммой, и наличие достаточного количества свободного места («воздуха»). Каждый критерий оценивает «качество» диаграммы для того или иного применения (отображения на мониторе, печати и т.д.). Такие критерии называются *эстетиками*. Задавшись некоторым набором эстетик, можно построить штрафную функцию (которая тем больше, чем больше расхождения между изображением диаграммы и критериями), и пытаться минимизировать ее, перемещая элементы, или разработать алгоритм, последовательно модифицирующий исходную диаграмму так, чтобы результат соответствовал выбранным критериям.

Постановка задачи

Рассмотрим диаграмму состояний конечного автомата. Она включает в себя состояния и переходы (и у тех, и у других есть дополнительные сущности, связанные с ними, – метки, но мы сейчас не будем останавливаться на этом вопросе). Диаграмме состояний (без вложенных состояний) можно сопоставить граф [5] (при укладке диаграммы состояний нам не важна ориентация дуг). Каждому состоянию соответствует вершина, а переходу – ребро. Нам необходимо несколько сузить поле деятельности, зафиксировав представление элементов и тип носителя, на котором мы планируем изображать диаграмму.

Диаграмма может быть изображена на плоскости, или, например, может быть построена объемная модель [6]. Для представления на мониторе современного персонального компьютера наиболее естественным представляется выбор плоского носителя.

Также нам необходимо выбрать вид элементов графа. Будем изображать вершину как прямоугольник, а ребро – как ломаную линию с конечным числом изломов. Выбор именно такого вида элементов объясняется субъективными эстетическими соображениями и сложившимися в области построения диаграмм традициями.

Для изображения вершины нам достаточно знать координаты левого верхнего угла, ширину и высоту, а для изображения ребра – координаты его начала, конца и всех точек излома.

Укладкой графа $G = (V; E)$ в декартовой системе координат $(X; Y)$ назовем множество $L = (G; F_V; F_E)$, где F_V – функция из множества вершин в множество параметров, необходимых для представления вершины в выбранной системе координат (в нашем случае $F_V : V \to X \times Y \times R \times R$, где последняя пара параметров – ширина и высота прямоугольника). F_E – функция из множества ребер в множество параметров, необходимых для представления ребра в выбранной системе координат (в нашем случае $F_E : V \to (X \times Y)^n, n \in N$, где параметр n – количество изломов, вообще говоря, меняется от ребра к ребру). Для простоты будем говорить, что в уложенном графе заданы геометрические параметры каждого элемента.

Укладку на плоскости мы будем называть *плоской***.** Данный термин не является общепринятым, в литературе (например, [7]) чаще говорят о плоском представлении графа. Укладку, в которой ребра представляют собой ломаные, состоящие только из горизонтальных и вертикальных отрезков, будем называть плоской ортогональной (или просто *ортогональной*).

Теперь необходимо оценить качество укладки (точнее, качество изображения, полученного по некоторой укладке). Выделим набор эстетик, по которым может оцениваться качество укладки. Основная задача такого выделения – обеспечить «читаемость» графа (однозначность представления информации). В работе [8] мною были выделены следующие эстетики:

- минимизация числа пересечений;
- минимизация площади;
- ограничение «свободного места»;
- минимизация изломов;
- минимизация общей длины ребер;
- минимизация коэффициента сторон;
- унификация длин ребер.

В результате анализа результатов работы алгоритмов, созданных в рамках работы [8], а также новых алгоритмов, следующие эстетики были отброшены как менее значимые:

- минимизация коэффициента сторон;
- унификация длин ребер.

Кроме того, следующие критерии в большой степени учитываются в других критериях и нарушаются только в очень специфических алгоритмах [9]:

- минимизация изломов;
- минимизация общей длины ребер.

Граф, соответствующий диаграмме состояний, отличается от обычного графа возможностью вложения состояний друг в друга, а также выделенных начальных и конечных состояний. Для оценки качества его укладки необходимо учесть дополнительные критерии:

- унификация размеров вершин, соответствующих простым состояниям – размер вершин для всех простых состояний (состояний, не имеющих вложенных состояний) необходимо максимально приблизить к заданному оптимальному размеру;
- разнесение начального и конечного состояний – расстояние между начальным и конечным состоянием должно быть достаточно велико.

Приведенные выше критерии практически покрывают набор критериев из работы [10] (те из них, которые применимы к ортогональной укладке). Неохваченными остались критерии построения по возможности симметричного изображения и унификации количества изломов на ребре. Первый критерий сложен для оценивания и учета, учет второго не приводит к значительному улучшению читаемости.

Обратим внимание на то, что принятые в диаграмме состояний *UML* начальное и конечное состояние не имеют прямоугольной формы. Для простоты заменим их объемлющими прямоугольниками.

Задача, которая ставится в настоящей работе, состоит в построении качественной ортогональной укладки диаграммы состояний. Наличие алгоритма укладки принципиально важно не только для редактора диаграмм, но и для организации импорта файлов, созданных в сторонних редакторах, так как для *UML* редакторов не существует унифицированного механизма обмена данными, сохраняющего информацию о местоположении элементов.

Окончательный список критериев, наиболее значимых при рассмотрении алгоритмов данной работы:

- минимизация числа пересечений;
- минимизация площади;
- ограничение «свободного места»;
- унификация размеров вершин, соответствующих простым состояниям.

Типы алгоритмов укладки графов

Для задачи укладки диаграммы не существует универсального решения, в связи с тем, что набор критериев, применяемых для оценки качества укладки, зависит от типа диаграммы. В [9] приведена классификация алгоритмов, применяемых для решения данной задачи. Рассмотрим две большие группы, принципиально отличающиеся подходом к решению:

- алгоритмы с физическим аналогом [11];
- аналитические алгоритмы [7].

Алгоритмы с физическим аналогом ставят в соответствие графу некоторую физическую модель, например, систему пружин, которые стремятся сжаться до некоторой заданной длины (такие алгоритмы называют «пружинным методом») или систему стержней и шарниров, с вершинами – одноименными электрическими зарядами. Для описания физической модели вводится понятие штрафной функции, задающей потенциальную энергию системы (такие алгоритмы еще называют алгоритмами минимизации потенциала). При этом задача укладки преобразуется в задачу нахождения минимума этой функции, которая решается с помощью сдвига на некоторый вектор каждой вершины графа и проверки изменения значения штрафной функции. Сдвиг вершин выполняется в цикле, условием выхода из которого является либо достижение локального минимума, либо достижение максимума числа допустимых итераций.

Наиболее популярная подгруппа группы алгоритмов с физическим аналогом – методы отжига. Они выделяются тем, что «колебания» системы затухают с каждой итерацией.

В рамках бакалаврской работы ([8]) был разработан и реализован алгоритм укладки диаграмм состояний методом отжига с применением ортогонализации. Проходя по состояниям, этот алгоритм пытается сдвинуть каждое из них. Если производится сдвиг простого состояния, вложенного в составное состояние, то запрещается выход за пределы объемлющего составного состояния. После сдвига каждого состояния вычисляется новое значение штрафной функции. Если штрафная функция уменьшилась, то новое расположение вершин фиксируется, и производится сдвиг следующего состояния, если

увеличилась – перемещение вершины отклоняется. Колебания системы затухают – расстояния, на которые сдвигаются состояния, уменьшаются на каждом шаге. Отдельным шагом выполняется ортогонализация укладки. Подробное описание штрафной функции, особенностей алгоритма и метода ортогонализации можно найти в [8].

Аналитические алгоритмы, в отличие от алгоритмов с физическим аналогом, представляют собой последовательность различных преобразований графа, приводящую к построению укладки. Это позволяет получать с их помощью гарантированный результат, удовлетворяющий выбранным критериям, так как критерии используются не для построения штрафной функции, а для построения самого алгоритма укладки.

К минусам аналитических алгоритмов можно отнести сложность их построения. Кроме того, они плохо поддаются модификации, и для постановки экспериментов приходится вносить серьезные изменения в сам алгоритм, а не в набор параметров, как это можно делать в случае использования, например, алгоритмов, базирующихся на штрафных функциях.

На практике (рассмотрено три реализации [12], [13], [14]). Аналитические алгоритмы позволяют относительно быстро получать наиболее качественные и красивые графы, в связи с этим для укладки диаграммы состояний в проекте *UniMod* было принято решение использовать эту группу алгоритмов в качестве основной.

Ниже будет описан алгоритм QMATH-STATECHART, базирующийся на аналитическом алгоритме QMATH,

Алгоритм QMATH-STATECHART

Данный алгоритм использует в качестве основы алгоритм QMATH, но адаптирован для укладки диаграмм состояний. Алгоритм состоит из следующих шагов:

1. разделение графа на слои;
2. укладка подграфов в каждом слое;
3. объединение графов.

Основная проблема, с которой мы встречаемся при укладке графа – это проблема укладки составных состояний. В рамках рассматриваемого алгоритма на первом шаге будем рассматривать множества состояний, находящихся непосредственно в одном составном состоянии, отдельно. Таким образом, будет получен набор графов, которые будут подвергнуты укладке независимо друг от друга. Для ребер, соединяющих состояния из различных подграфов, будут зарезервированы порты (это необходимо при определении размеров состояний). При этом необходимо учитывать st-нумерацию на каждом слое (чтобы определить, будут ли ребра, которые необходимо добавить позже, входящими или исходящими) На рис. 1 приведен пример укладки графа (выполненной алгоритмом QMATH-STATECHART), в вершинах v2, v5 и v6 которого есть вложенные слои.

Каждый подграф, полученный на предыдущем шаге, укладывается с помощью алгоритма QMATH. Алгоритм QMATH работает только с st-ориентированными двусвязными графами (вопросы преобразования графа к диграфу, st-ориентации, обратного преобразования, а также основных идей, на которых базируется алгоритм QMATH, не рассматриваются в связи с тем, что объем работы ограничен). Все ребра направлены от вершин с меньшим номером к вершинам с большим номером. Размер прямоугольника, соответствующего вершине, определяется в момент добавления вершины в укладку. Все исходящие ребра выходят из портов на верхней грани вершины, все входящие ребра входят в порты на боковых сторонах. При этом мы не используем точечные вершины (таким образом, минимальный размер вершины – 1).

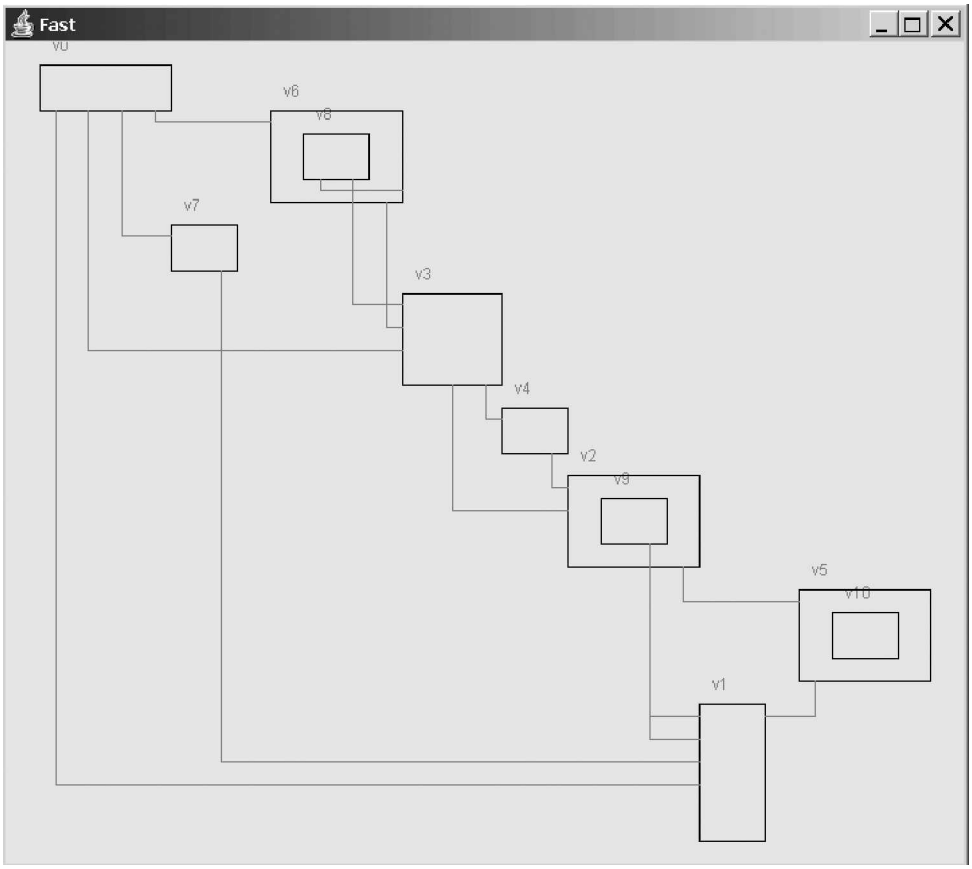


Рис. 1. Укладка графа со слоями

Рассматривая очередную вершину, распределим входящие ребра между правой и левой гранями прямоугольника. Пусть в вершину входит $in(v)$ ребер, тогда $\left\lfloor \frac{in(v)}{2} \right\rfloor$ ребер входят в правую сторону, а оставшиеся $\left\lceil \frac{in(v)}{2} \right\rceil$ – в левую. Таким образом, при добавлении вершины нам потребуется добавить не более $\left\lceil \frac{in(v)}{2} \right\rceil$ рядов и $out(v)$ колонок (исключением являются вершины с одним входящим или исходящим ребром, так как для них добавляется две колонки или ряда, соответственно).

Рассмотрим вершину v и все вершины u такие, что существует ребро $e = (u, v)$. Пусть число $in(v)$ – четное. Тогда найдутся колонки c_1 и c_2, соответствующие ребрам e_1 и e_2, такие, что:

- c_1 находится левее c_2;
- есть ровно $\frac{in(v)}{2} - 1$ колонок, содержащих ребра, входящие в v, и расположенных левее c_1;
- есть ровно $\frac{in(v)}{2} - 1$ колонок, содержащих ребра, входящие в v, и расположенных правее c_2.

Ребра e_1 и e_2 назовем входящими медианами вершины v. e_1 – левая медиана, e_2 – правая медиана. Если же число $in(v)$ – нечетное, то есть ровно одно ребро-медиана e, справа и слева от колонки соответствующей которому есть $\left\lfloor \frac{in(v)}{2} \right\rfloor$ колонок с ребрами, входящими в v.

При добавлении вершины в укладку добавляется необходимое количество колонок и рядов. Причем колонки добавляются между входящими медианами (если их две) или справа от медианы (если она одна).

Для уменьшения занимаемой площади к алгоритму может быть применена техника использования рядов и колонок, аналогичная таковой в алгоритме QMATH-4 (подробно эта адаптация описана в [16]). На рис. 2 приведена укладка графа алгоритмом QMATH.

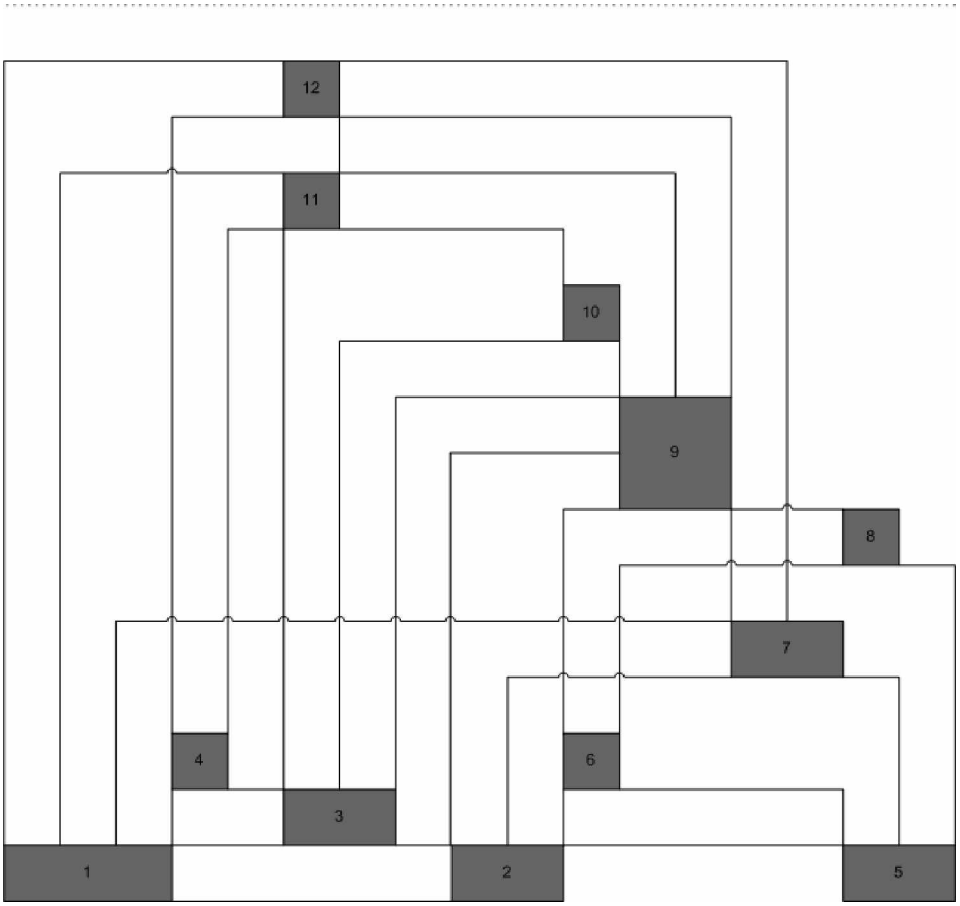


Рис. 2. Укладка графа алгоритмом QMATH

При определении размеров состояний учитывается информация о входящих и исходящих ребрах, которые соединяют вершины различных подграфов.

На этапе объединения графов добавляются ребра между слоями, а также рассчитываются окончательные размеры состояний, в которых есть вложенные.

Алгоритм QMAT-STATECHART без повторного использования рядов и колонок был реализован в рамках проекта UniMod и входит в UniMod версии 1.3. На рис. 3 приведен пример конечного автомата, уложенного с помощью этого алгоритма.

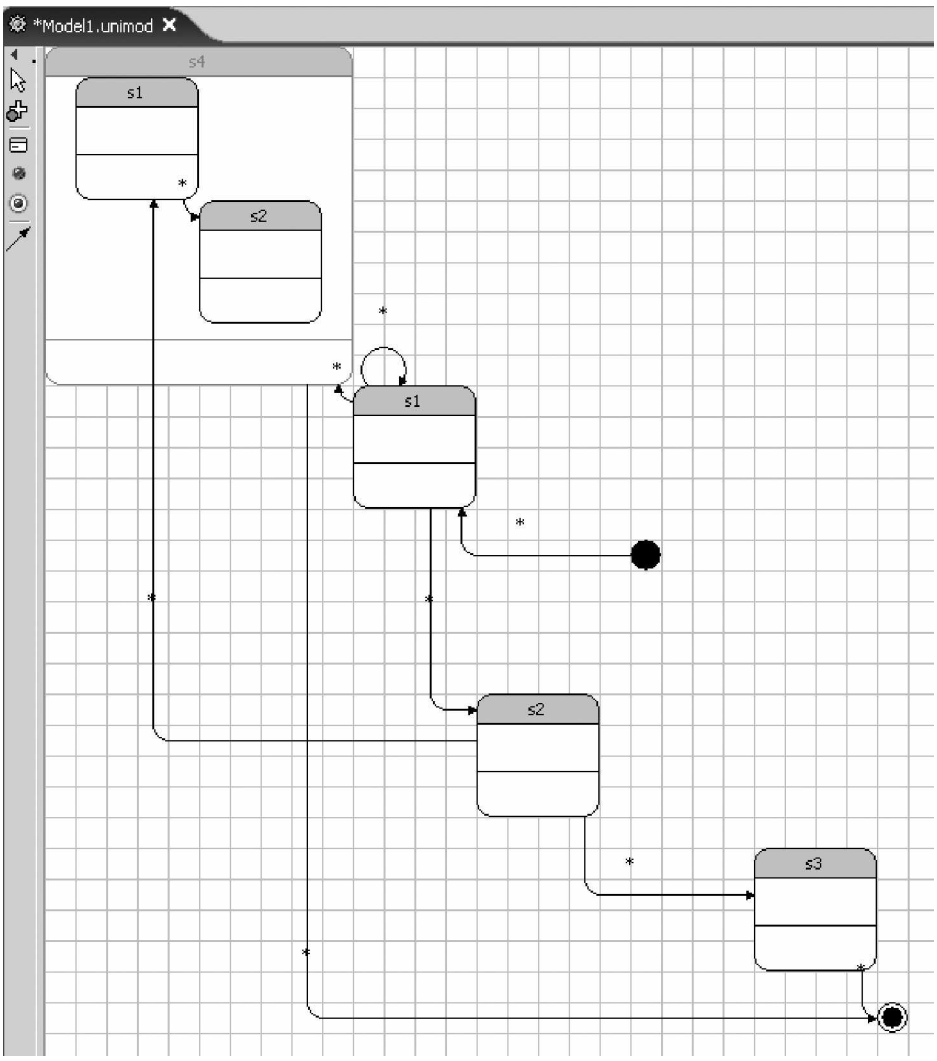


Рис. 3. Результат укладки с помощью алгоритма QMATH-STATECHART

Заключение

В работе исследована проблема плоской укладки диаграмм состояний *UML*. Поставлена задача укладки, дан обзор наиболее популярных методов. На базе существующих аналитических алгоритмов плоской укладки графа разработан алгоритм QMATH-STATECHART, позволяющий уложить произвольную диаграмму состояний языка *UML*. Ценность алгоритма заключается в том, что он разработан специально для укладки *UML*-диаграммы состояний.

Литература

1. Гуров В.С., Мазин М.А. Веб-сайт проекта UniMod. // http://unimod.sourceforge.net/
2. Rumbaugh J., Jacobson I., Booch G. The Unified Modelling Language Reference Manual, Second Edition. MA //Addison-Wesley, 2004.
3. Шалыто А.А. Switch-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука. 1998.
4. Шалыто А.А., Туккель Н.И. Танки и автоматы. //BYTE/Россия. 2003. № 2. С. 69–73.
5. Новиков Ф.А. Дискретная математика для программистов. СПб: Питер. 2001.
6. Tamassia R. Advances in the Theory and Practice of Graph Drawing. http://www.cs.brown.edu/people/rt/papers/ordal96/ordal96.html

7. Battista G., Eades P., Tamassia R., Tollis I. Graph Drawing. Algorithms for the Visualization of Graphs. New Jersey: Prentice Hall. 1999.
8. Коротков М. Разработка и реализация алгоритма укладки диаграмм состояний. СПбГУ ИТМО. Дипломная работа. 2005.
9. Sugiyama K. Graph Drawing and Applications for Software and Knowledge Engineers. Singapore: Mainland Press. 2002.
10. Касьянов В.Н., Евстигнеев В.А. Графы в программировании: обработка, визуализация, применение. СПб: БХВ-Петербург, 2003.
11. Frutcherman T., Reingold E. Graph Drawing by Force-directed Placement. // Software - Practice and Experience. 1991. № 21. P. 1129–1164.
12. Веб-сайт проекта AGD (Algorithms for Graph Drawing). // http://www.ads.tuwien.ac.at/AGD/
13. Веб-сайт проекта GDT (Graph Drawing Toolkit). //http://www.dia.uniroma3.it/~gdt
14. Веб-сайт проекта Graph Layout Toolkit компании Tom Sawyer Software // http://www.tomsawyer.com/tsl/tsl.java.php
15. Battista G., Garg A., Liotta G., Tamassia R., Tassinari E., Vargiu F. An Experimental Comparison of Four Graph Drawing Algorithms. // Computational Geometry. 1997. V. 7. № 5–6. P. 303–325.
16. Papakostas A., Tollis I.G. Algorithms for area-efficient orthogonal drawings. // Computational Geometry. 1998. V. 9. № 1–2. P. 83–110.

ПОДХОД К ТЕСТИРОВАНИЮ ПРОГРАММНЫХ ИНТЕРФЕЙСОВ ПРИЛОЖЕНИЙ МОБИЛЬНЫХ УСТРОЙСТВ

К.В. Рубинов
Научный руководитель – д.т.н., профессор А.А. Шалыто

На основании опыта тестирования программных интерфейсов приложений для мобильных телефонов автором создан подход к их тестированию.

Введение

Мобильные телефоны получили широкое распространение, а разработка приложений для мобильных устройств является одной из самых прибыльных сфер создания программных продуктов. Новые функциональные возможности программ вводятся в мобильные телефоны часто и быстро.

Программные интерфейсы приложений для мобильных устройств реализуются в виде стандартов, которые поддерживают производители устройств и программного обеспечения. Соответствие стандартам контролируется и базируется на тестировании интерфейсов в соответствии со спецификациями.

В настоящее время не существует общего подхода к созданию тестов для программных интерфейсов – известные методики в своем большинстве основываются на интуиции и опыте инженеров по тестированию.

Автором предлагается подход к созданию тестов программных интерфейсов приложений для мобильных телефонов, основанный на спецификациях к интерфейсам и автоматном подходе [1]. При этом на основании спецификаций к интерфейсам описываются и создаются тестовые сценарии, а тестовое приложение создается на основе автоматного программирования.

1. Постановка задачи

Тестирование программного обеспечения (*software testing*) – деятельность, выполняемая для оценки и улучшения качества программного обеспечения. Эта деятельность в общем случае базируется на обнаружении дефектов и проблем в программных системах [2].

Процесс тестирования является неотъемлемой частью любого процесса разработки программного обеспечения и играет важную роль на всех этапах его создания. Обнаружение ошибок на ранних стадиях жизненного цикла разработки программного обеспечения позволяет снизить затраты на его создание в целом и уменьшить вероятность появления сбоев в дальнейшем.

Программный интерфейс приложения (Application Programming Interface, API) или интерфейс прикладного программирования – это набор функций, предоставляемый некоторой программной системой для использования ее функциональности в других прикладных программах.

API – это не конечный программный продукт, а промежуточное звено между различными программами или определенными уровнями программных систем. Как и любое звено, задействованное в программной среде, API должен выполнять свои функции в соответствии со спецификацией и предоставлять разработчику именно ту функциональность, для которой этот интерфейс предназначен. Проверку указанного соответствия можно выполнить с помощью функционального тестирования.[1]

[1] Функциональное тестирование – проверка соответствия системы предъявляемым к ней требованиям.

При создании программного продукта необходимо обеспечить качество всех его компонент. Поэтому необходимо проводить и тестирование API. Оно должно являться этапом разработки наравне с тестированием программного продукта в целом.

Тестирование API имеет свои особенности, которые будут изложены ниже.

2. Требования к программному интерфейсу приложения

Основным документом, на котором базируется тестирование API, является спецификация программной системы – набор предъявляемых к ней требований.

Для описания программного интерфейса приложения выделяют несколько классов требований:

- требования к данным программного интерфейса (Software Interface Data Requirements);
- функциональные (Functional Requirements) или поведенческие требования (Behavioral Requirements).

2.1. Требования к данным программного интерфейса

Требования к данным программного интерфейса имеют декларативный характер и вносят ограничения на синтаксис и формат используемых данных.

Приведем пример требований к синтаксису данных для определения протоколов.

Пример 1. Требование к синтаксису

"`http`*",* *"*`https`*" протоколы:* `HTTP Player`

Мультимедийный API должен поддерживать ввод локаторов для создания плееров из `HTTP/HTTPS` *соединения к медиа файлу в стандартном синтаксисе* `HTTP/HTTPS URL`*:*

```
http://<путь_к_медиа_файлу>
https://<путь_к_медиа_файлу>
```

В этом требовании задается синтаксис для мультимедийных локаторов: `http://<путь_к_медиа_файлу>` и `https://<путь_к_медиа_файлу>`.

2.2. Функциональные (поведенческие) требования

Функциональные требования в основном имеют условный (conditional) характер. Они определяют функциональность (поведение) программной системы, которая должна быть создана разработчиками.

Обычно в качестве ключевых слов в описании функциональных требований используются такие конструкции как «должен» (*shall*) и «если, то» (*if ..., then*).

Исходя из семантики методов, предоставляемых программным интерфейсом приложения, возможны следующие варианты функционирования:

- ввод данных;
- вывод данных;
- переход в состояние;
- получение события/сообщения;
- отправка события/сообщения.

Объекты функционирования и взаимодействия – методы, события, диалоги, исключения, устройства ввода информации и т.п.

Приведем пример функционального требования к методу, обеспечивающему ввод данных.

Пример 2. Ввод данных

```
public int setLevel(int level)
```

Этот метод должен устанавливать уровень звука для плеера на величину, указанную в параметре `level`*. Этот параметр является целым числом в диапазоне 0…100.*

Значение ноль должно устанавливать минимальное значение звука – тишина. Значение 100 должно устанавливать максимально возможный уровень звука.

Если параметр `level` *имеет отрицательное значение, то уровень звука должен быть установлен в ноль (silence).*

Если параметр `level` *имеет значение больше 100, то уровень звука должен быть установлен 100 (максимально возможный уровень звука).*

Возвращаемое значение должно представлять новое установленное значение уровня звука.

Когда вызов метода `setLevel()` *заканчивается изменением звука, событие* `VOLUME_CHANGED` *должно быть доставлено к зарегистрированному слушателю события* `PlayerListener`. *Параметр* `eventData` *должен быть* `VolumeControl`*-объектом. Приложение может использовать этот объект для дальнейшего изменения звука.*

В приведенном требовании ввод данных для метода `setLevel()` обеспечивается через входной параметр `level`.

Приведем также пример функционального требования к методу, обеспечивающему вывод данных.

Пример 3. Вывод данных

```
public int getState()
```

Данный метод должен возвращать текущее состояние плеера.

Возможные значения: `UNREALIZED, REALIZED, PREFETCHED, STARTED, CLOSED.`

Вывод данных обеспечивается через значение, возвращаемое методом `getState()`.

Пример функционального требования к методу, обеспечивающему переход в состояние.

Пример 4. Переход в состояние

```
public void stop()
```

Вызов данного метода для плеера в состоянии `STARTED` *должен остановить проигрывание медиа-файла.*

Когда выполнение метода `stop()` *заканчивается, плеер должен перейти в состояние* `REALIZED`.

Вызов данного метода на остановленном плеере (плеер в состоянии `UNREALIZED`*) будет проигнорирован.*

Если метод `stop()` *вызван, когда плеер находится в состоянии* `CLOSED`*, то должно появиться исключение* `IllegalStateException`.

Вызов метода `stop()`может обеспечить переход плеера в состояние `REALIZED`.

Пример функционального требования к методу, обеспечивающему получение/отправку события/сообщения.

Пример 5. Получение/отправка события/сообщения

Когда плеер достигает конца медиа-файла, событие `END_OF_MEDIA` *должно быть доставлено к зарегистрированному слушателю событий* `PlayerListener`. *Параметр* `eventData` *должен быть* `java.lang.Long` *объектом, указывающим время, когда плеер достиг конца медиа-файла.*

3. Техники тестирования

Классификация, описанная ниже, применима к процессу тестирования, который базируется на спецификациях программного интерфейса приложения. Техники, базирующиеся на спецификации (Specification-based techniques), разделяются на следующие виды.

1. ***Positive technique.*** Тесты строятся с ориентацией на использование тех величин, которые находятся в рамках специфицированных пределов значений. Техника предполагает получение заданных реакций тестируемой системы на допустимые («правильные») воздействия.
2. ***Negative technique.*** Тесты строятся с ориентацией на использование тех величин, которые выходят за рамки специфицированных пределов значений, не соответствуют определенному формату или используют искаженные данные. Техника предполагает получение заданных реакций тестируемой системы на недопустимые, некорректные («неправильные») воздействия.

3. ***Boundary technique.*** Тесты строятся с ориентацией на использование тех величин, которые определяют предельные или близкие к предельным характеристики тестируемой системы. Данную технику можно выделить как подкласс техники *positive*.
4. ***Interaction technique.*** Тесты строятся с ориентацией на проверку взаимодействия между программными компонентами/модулями, имеющими связи в виде совместных процессов, ресурсов или устройств.
5. ***Stress technique.*** Техника является развитием и обобщением четырех техник, описанных выше. Тесты строятся с ориентацией на повышение нагрузки на программную систему, вплоть до достижения предопределенных показателей загрузки и далее на всем протяжении повышения загрузки системы. Примером стрессового тестирования может служить повторяющийся вызов определенного метода с высокой частотой в условиях ограничения ресурсов.

4. Тестирование

4.1. Процедура выбора тестов

Для создания полного и эффективного в использовании тестового набора необходимо подобрать и применить максимально широкий спектр тестов к каждому требованию. Далее рассматриваются возможные варианты соответствия вариантов функционирования и техник тестирования для требований к данным программного интерфейса и функциональных требований. В таблице приведены семь групп функциональных требований и пять техник тестирования. Клетки таблицы с меткой «X» означают, что для определенной группы требований могут быть применимы определенные техники тестирования.

Используемые сокращения: **Positive – P; Negative – N; Boundary – B; Stress – S; Interaction – I.**

№ группы	Требование	Тест				
		P	**N**	**B**	**S**	**I**
1	Ограничение, определение формата (синтаксис)	X	X			
2	Ограничение, определение типа (String, Integer...)	X	X	X		
3	Ввод данных	X	X	X	X	X
4	Вывод данных	X			X	X
5	Переход в состояние	X	X		X	X
6	Получение события/сообщения	X			X	X
7	Отправка события/сообщения	X	X		X	X

Таблица. Соответствие функциональных требований и техник тестирования

Для групп 1 и 2 применимы **P**, **N** и **B** техники, так как они основаны на контроле данных и не предполагают проверку взаимодействий.

Для требований, предполагающих ввод данных, применимы все техники тестирования, так как ввод данных может вызывать весь спектр реакций и взаимодействий в тестируемой системе.

Для требований, описывающих вывод данных, по сравнению с вводом, исключаются **N** и **B** техники. Вывод данных – ответная реакция системы на определенное воздействие, следовательно, создать условия для применения **N** и **B** техник невозможно.

Требования, описывающие условия для перехода между состояниями, сходны с требованиями по вводу данных, за исключением того, что **B** техника неприменима в данном случае, так как обычно состояниям не предписываются предельные характеристики.

Требования, описывающие получение и отправку событий/сообщений, как правило, не предписывают событиям/сообщениям предельные характеристики.

Группы требований 1 и 2 могут являться общими для остальных групп и проверяться в процессе тестирования групп 3 – 7.

Итак, для выбора тестов необходимо ответить на следующие вопросы.

1. К какой группе относится требование?
2. Какие существуют компоненты, связи, процессы, ресурсы или устройства, смежные с тестируемыми?
3. Какие тесты удобней, легче и выгодней выполнять на используемом оборудовании (автоматические/ручные)?
4. Какое количество тестов даст реалистичные статистические результаты?

Дополнительно к описанным выше вопросам необходимо учесть, какие возможны и/или необходимы предусловия для тестирования.

Таким образом, проанализировав полученные данные и сделав обзор предъявляемых требований, можно приблизительно определить количество тестов и их специфику. Полученной информации должно быть достаточно для создания первоначальной версии плана тестирования.

Приведем пример выбора тестов для функционального требования.

Пример 6. Функциональное требование

```
public long setMediaTime(long newTime)
```

Данный метод должен устанавливать время проигрывания на величину, указанную в параметре `newTime`. *Этот параметр задает новое время в микросекундах.*

Возвращаемое значение должно показывать фактически установленное время. Оно может отличаться от величины `newTime`, *так как установка времени может быть не очень точна из-за ограничений платформы.*

Если параметр `newTime` *имеет отрицательное значение, тогда время проигрывания будет установлено в ноль (начало медиа-файла).*

Если параметр `newTime` *больше, чем продолжительность медиа-файла, тогда время проигрывания должно быть установлено на конец медиа-файла.*

Время проигрывания может быть установлено, если плеер находится в состояниях `REALIZED`, `STARTED`.

Если метод `setMediaTime()` *вызван, когда плеер находится в состояниях* `UNREALIZED` *или* `CLOSED`, *то должно появиться исключение* `IllegalStateException`.

Проанализировав требование, можно сделать следующие выводы:

- требование является функциональным;
- формат значений определен (`long`).

В данном примере требование содержит в себе:

- ввод данных через входной параметр `newTime` метода `setMediaTime`;
- вывод данных через возвращаемое значение метода `setMediaTime`;
- получение события/сообщения об исключительной ситуации.

Также в требовании есть зависимость от состояний плеера.

В соответствии с таблицей, приведенной выше, для тестирования примера необходимо использовать все перечисленные выше группы тестов.

1. *Positive*:
   1.1. Проверка установки значения `newTime` в пределах продолжительности медиа-файла в состояниях `REALIZED, STARTED`. Ожидается, что возвращаемое значение равно значению параметра `newTime` с определенной точностью.
   1.2. Проверка установки значения параметра `newTime` на отрицательное значение в состояниях `REALIZED, STARTED`. Ожидается, что возвращаемое значение равно нулю.
   1.3. Проверка установки значения `newTime` на значение, превышающее продолжительность медиа-файла в состояниях `REALIZED, STARTED`. Ожидается, что возвращаемое значение равно длительности медиа-файла.
2. *Negative*:
   2.1. Проверка установки значения newTime в состояниях UNREALIZED, CLOSED. Ожидается появление исключения IllegalStateException.
3. *Boundary*:
   3.1. Проверка установки значения параметра `newTime` на значения: ноль, `MIN_LONG`, `MAX_LONG`. Ожидается, что возвращаемое значение будет соответствовать спецификации, как и в *positive* группе.
4. *Stress*:
   4.1. Чередование вызовов метода с переходами плеера из состояния в состояние.
   4.2. Повторяющийся вызов метода.
5. *Interaction*:
   В данном примере не уточняется, какие компоненты взаимодействуют с окружением метода. Поэтому используется минимальный вид проверки зависимости от внешнего события – проверка вызова метода с последующим экстренным выходом из тестового приложения. Последующий запуск и выполнение тестового приложения должен пройти без исключительных ситуаций.

Так как реализация метода для различных форматов медиа-файлов может различаться, рекомендуется провести тестирование метода на каждом определенном спецификацией формате.

Целевое назначение метода можно сформулировать как «установка точки начала проигрывания медиа-файла». Таким образом, желательно, в дополнение к описанным тестам, иметь такой тест, в котором тестер будет вручную устанавливать значение точки начала проигрывания медиа-файла, и контролировать на слух, соответствует ли проигрываемый фрагмент ожидаемому.

Итак, выбор тестов – это задача, которая решается каждый раз сначала, когда создается новая функциональность. Программный интерфейс приложения предполагает разработку некоторой системы или программы на своей основе. Тестер должен учитывать это и предполагать возможные варианты использования будущей системы/программы. Описанная выше процедура выбора тестов в определенных случаях не может дать полного покрытия тестируемой функциональности из-за своей статичности и непрямой зависимости от возможных вариантов использования системы. Таким образом, помимо тестирования «в статике», должно производиться еще и тестирование «в динамике» – тестами должны быть пройдены хотя бы основные пути/варианты использования системы.

В следующем разделе описана процедура выбора тестов, которые учитывают динамические свойства системы и возможные варианты ее использования.

4.2. Сценарное тестирование

Сценарий – подмножество вариантов использования системы или приложения.

Тестовый сценарий – последовательность выполнения тестовых процедур.

При построении тестов на основе функциональных спецификаций в большинстве случаев информация о внутренней структуре системы неизвестна. Проектирование тестов происходит на основе принципа «черного ящика». Сценарное тестирование в данном случае может помочь в упрощении процесса разработки тестов и покрытия возможных вариантов использования, однако оно не гарантирует проверку всех внутренних состояний системы, так как ее внутренняя структура неизвестна.

Создание тестовых сценариев позволяет оптимизировать процесс разработки тестов и, следовательно, процесс тестирования. При ограничениях на временные и человеческие ресурсы в процессе создания основных наборов тестов, выполняемого без анализа взаимосвязей в требованиях, могут быть пропущены проверки определенных вариантов использования системы, а некоторая последовательность действий может остаться не покрытой тестированием.

Предлагается выделить два подхода к созданию тестовых сценариев. Первый – от требований к тестам, второй – от имеющихся тестов к новым тестам.

В первом случае выполняется исследование требований, в котором прослеживаются связи между требованиями различных типов. Выявляются отношения и наложения в методах и состояниях системы. Анализируются варианты использования системы и связи между ними.

Во втором случае выполняется анализ плана тестирования на предмет совмещения схожих групп тестов. Например, объединение *positive* и *boundary* групп. Производится анализ уже существующих тестов, выявляются схожие по последовательности выполнения операций фрагменты тестов. Предусловия выполнения тестов также следует рассмотреть.

Результатом этих действий должен быть набор из возможных последовательностей вариантов выполнения тестовых процедур, пригодных для создания тестового сценария.

4.2.1. Выбор тестовых сценариев

Как определено выше, тестовый сценарий – это последовательность выполнения тестовых процедур. Для создания последовательности первоначально необходимо определить минимальный объем процедур – наименьшие составные части, из которых будет строиться тестовый сценарий. Далее необходимо определить оптимальный размер сценария по отношению к системе и к планируемому тестовому набору. Слишком длинная последовательность в тестовом сценарии может вызвать неоправданно длинный процесс выполнения теста, что может помешать оптимальному выбору и исполнению тестов в дальнейшем. Слишком короткая последовательность может вызвать обратную ситуацию – свести к минимуму эффективность применения сценарного тестирования из-за малого отличия от единичных тестовых процедур. Таким образом, тестовый сценарий должен быть оптимален для тестируемой системы/программы.

Существуют следующие разновидности тестовых сценариев.

1. Зависимый тестовый сценарий – в случае нахождения тестом неисправности или ошибки в любом из тестовых случаев тестирование останавливается. Особенности – высокая приоритетность тестовых случаев и сильная зависимость между ними.
2. Независимый тестовый сценарий – в случае нахождения тестом неисправности или ошибки в любом из тестовых случаев тестирование продолжается до конца. Особенности – равнозначность в приоритетах тестовых случаев и слабая зависимость между ними.
3. Комбинированный тестовый сценарий – сочетаются первый и второй принципы. В тестовом сценарии выделяются основная (приоритетная) последовательность, ошибка в которой предполагает окончание тестирования, и второсте-

пенная последовательность, ошибка в которой может быть проигнорирована и учтена позднее при анализе результатов тестирования.

Таким образом, помимо функционального наполнения тестового сценария, имеются три параметра, которые необходимо определить для создания адекватной тестовой последовательности: минимальная единица последовательности, длина последовательности и тип логических связей внутри последовательности.

Каждый подтест (наименьшая составная часть теста) состоит из начального состояния, ввода и прогнозируемого итога [4]. В идеальном случае, тестовая последовательность должна быть составлена таким образом, чтобы итог и/или конечное состояние одного подтеста являлось вводом и/или начальным состоянием другого (последующего подтеста).

Перед разработчиком тестов стоят задачи выбора и сочетания возможных вариантов тестовых сценариев.

4.2.2. Анализ требований

На данном этапе разработчик тестов для того, чтобы извлечь из требований возможные варианты использования интерфейса приложения, должен «встать на место» того, кто будет использовать API, в общем случае, разработчика приложения.

Пусть, например, в требованиях имеется диаграмма (рис. 1), описывающая возможные состояния и переходы плеера.

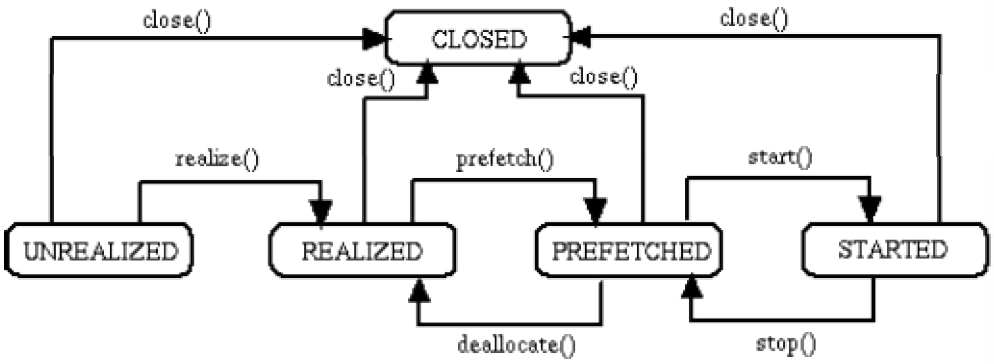


Рис.1. Состояния и переходы плеера

В этой диаграмме может быть выделена последовательность вызовов методов, позволяющая осуществить переходы между наибольшим числом состояний плеера:

```
realize() -> prefetch() -> start() -> close()
```

Можно назвать эту последовательность основной последовательностью проверки корректности переходов плеера из состояния в состояние. При дальнейшем тестировании ее следует считать приоритетной – при невозможности перехода из одного состояния в другое необходимо прекращать тестирование и считать результат теста отрицательным (failed).

Параллельно с анализом диаграммы изучаются другие требования к API. Среди них находятся такие требования, в которых описание методов имеет связь с состояниями плеера. Например, конкретный метод может быть вызван только в определенных состояниях плеера (пример 6).

Полученной информации достаточно, чтобы создать тестовый сценарий, содержащий комбинацию из переходов из состояния в состояние с вызовом методов в определенной последовательности, причем конечное состояние одного подтеста будет являться начальным состоянием другого.

Минимальной единицей сценария удобно сделать проверку тех методов, вызов которых возможен в одном состоянии (метод `setMediaTime` из примера 6).

Тестовый сценарий можно построить по комбинированному принципу: если переход в новое состояние невозможен – прерывается основная последовательность сценария, то тестирование прекращается, а если вызов конкретного метода в состоянии невозможен (не может быть выполнен определенный подтест) – тестирование продолжается.

На простейшем примере было показано, как анализ требований может быть использован для создания тестовых сценариев. К сожалению, любые требования не лишены недостатков и не могут на 100% описать реализуемую функциональность, а тем более возможные варианты ее использования. Это приводит к затруднениям при создании тестов для API, которые могут привести к неполному тестированию функциональности и другим сложностям.

4.3. Тестовое приложение (реализация тестовых сценариев)

Разработка тестов для API в итоге сводится к разработке приложения (каркаса), в которое интегрируются тестовые сценарии. Разработчик тестов должен знать язык программирования, в котором реализован API, и учитывать особенности языка при создании приложения.

Тестовые сценарии запускаются из меню приложения в автоматическом режиме или вручную. Результатом выполнения приложения является вердикт: тест прошел – положительный результат (passed), или тест не прошел – отрицательный результат (failed).

Общих рекомендаций при создании тестовых приложений для API не существует. Тесты должны быть спроектированы таким образом, чтобы после их выполнения можно было с точностью определить место ошибки. Также желательно обеспечить основные и необходимые характеристики тестов:

- универсальность;
- переносимость;
- удобство сопровождения;
- расширяемость.

Для того чтобы обеспечить все перечисленные характеристики и иметь регулярную структуру тестового приложения, автором предлагается применять автоматный подход для разработки и программирования тестового приложения.

Первоначально необходимо выделить и описать основные состояния и объекты тестового приложения. На рис. 2 представлена схема связей простейшего автомата с функциональностью достаточной для реализации теста, а на рис. 3 – его граф переходов.

Тестовое приложение может находиться в двух состояниях (рис. 2).

1. *Idle* – исходное состояние приложения с отображением статистики выполнения тестов.
2. *Test* – выполнение тестов.

В свою очередь, выполнение тестов *Test* происходит в три этапа.

1. Инициализация ресурсов для выполнения тестового сценария – *Init.*
2. Выполнение тестового сценария – *Run.*
3. Очистка выделенных ресурсов – *CleanUp.*

По результатам выполнения тестов обновляются статистические данные – количество положительных и отрицательных результатов из общего числа выполненных тестов, которые выводятся в состоянии *Idle.*

Использование автоматов и графов переходов при создании тестовых приложений дает возможность обеспечить прозрачность структуры и поведения тестового приложения для разработчика тестов.

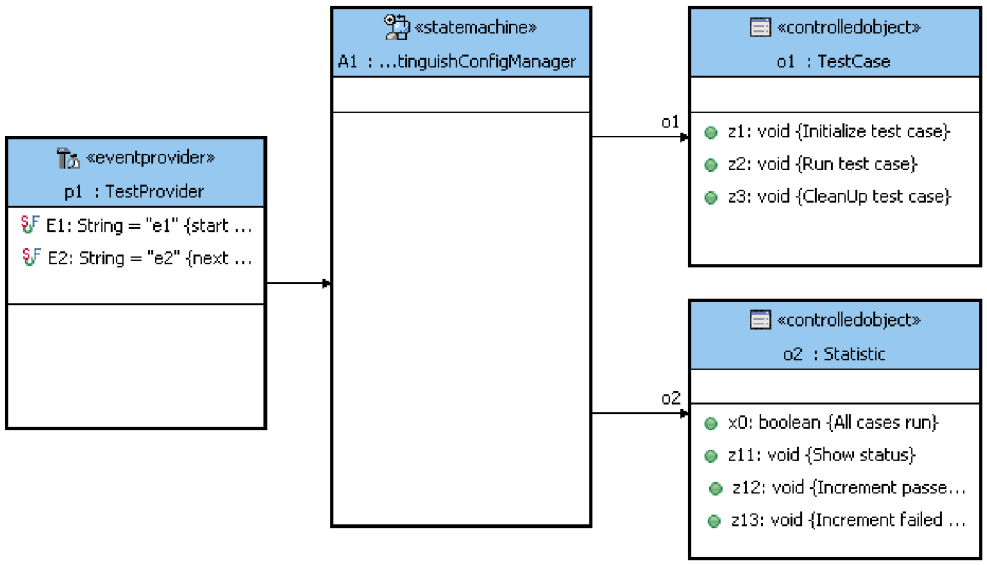


Рис. 2. Схема связей автомата

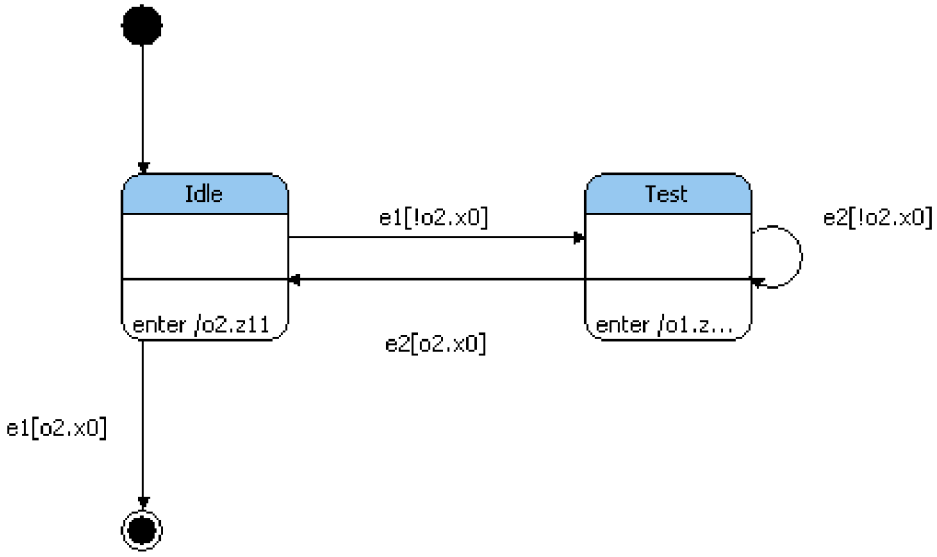


Рис. 3. Граф переходов автомата

Учитывая специфику мобильных телефонов и предъявляемых к ним требований, предложенный подход к созданию тестовых приложений позволяет обеспечить простоту внесения изменений в разработанное тестовое приложение, возможность повторного использования и расширения функциональности тестов.

Заключение

В работе изложены основные вопросы тестирования программного интерфейса приложения, произведен обзор требований программного интерфейса приложения и техник тестирования, описаны процедуры выбора тестов и создания тестовых сценариев, описан подход к созданию тестовых приложений для тестирования *API*.

Этот подход может быть использован при создании новых тестов, а также быть полезным при выявлении областей улучшения существующих тестов для программного интерфейса приложения.

Литература

1. Шалыто А.А. Технология автоматного программирования. // Мир ПК. 2003. №10. С. 74–78. http://is.ifmo.ru/works/tech_aut_prog/
2. Орлик С. Программная инженерия. Тестирование программного обеспечения (Software Testing) / Введение в программную инженерию и управление жизненным циклом ПО. 2004. http://sorlik.blogspot.com
3. Интерфейс программирования приложений. http://ru.wikipedia.org/wiki/API
4. Бейзер Б. Тестирование черного ящика. Технологии функционального тестирования программного обеспечения и систем. СПб: Питер, 2004. 318 с.

АНАЛИЗ АЛГОРИТМОВ ПОИСКА ПЛАГИАТА В ИСХОДНЫХ КОДАХ ПРОГРАММ

О.А. Евтифеева*, А.Л. Красс, М.А. Лакунин*, Е.А. Лысенко*, Р.Р. Счастливцев*
*** (Санкт-Петербургский государственный университет)**
Научный руководитель – д.т.н., профессор А.А. Шалыто

Статья содержит краткий обзор существующих алгоритмов поиска плагиата в исходных кодах программ/ Рассмотрены их особенности, преимущества и недостатки, приведены результаты тестирования наиболее популярных некоммерческих детекторов плагиата, использующих эти алгоритмы.

1. Введение

Плагиат в исходных кодах программ встречается как в коммерческой разработке программного обеспечения, так и в образовании. Сейчас, когда к соблюдению авторских прав в нашей стране стали относиться более внимательно, задача выявления плагиата стала актуальной: необходимы методы и средства, позволяющие автоматизировать этот процесс.

Формально определить понятие плагиата крайне сложно. Условимся под этим термином понимать случай, когда между исходными кодами двух программ есть существенная (на уровне языка программирования) общая часть. При этом производная программа получается из оригинальной несложными преобразованиями, цель которых – скрыть факт заимствования вставкой лишних операторов, изменением порядка следования в программе независимых операторов, разбиением одной функции на две, изменением имен переменных и так далее. Такое определение в большинстве случаев пригодно, ибо серьезные изменения исходного кода, сделанные для сокрытия плагиата вручную, крайне трудоемки, если же для этого используются автоматические средства (например, *обфускаторы*), то по виду исходного кода это легко определяется человеком.

В начале основной части этой статьи мы рассмотрим, в какое представление переводится исходный код программ в большинстве современных алгоритмов, далее будут кратко описаны наиболее интересные современные алгоритмы поиска плагиата в исходных текстах программ. В конце этой части мы приведем результаты тестирования соответствующих детекторов плагиата, а в заключении обсудим дальнейшие перспективы исследований в данной области.

2. Алгоритмы

2.1. Классификация

Существующие в данной области алгоритмы можно классифицировать следующим образом:

- текстовые алгоритмы,
- структурные алгоритмы,
- семантические алгоритмы.

Особенностью текстовых алгоритмов является то, что они представляют исходные коды программ в виде текста, а точнее, строки над алфавитом, символ которого соответствует определенному оператору или группе операторов языка программирования. Причем аргументы операторов (которые сами могут быть операторами) игнорируются, что делает многие элементарные действия по сокрытию плагиата (например, изменения имен переменным) бесполезными. Символ такого алфавита традиционно называют *токеном*.

Текстовые алгоритмы включают в себя как наиболее эффективные современные алгоритмы поиска плагиата в исходных кодах программ, так и самые старые алгоритмы. Примером последних может служить [1], где для проверки на плагиат производится подсчет некоторых характеристик программ (например, количество операторов ветвления в программе), а затем полученные векторы характеристик сравниваются между собой. Конечно, этот метод очень легко обойти, к тому же он находит плагиат во всей программе и не может быть эффективно адаптирован для нахождения относительно небольшого фрагмента плагиата в большом исходном коде.

Практически во всех алгоритмах текстовой группы считается, что если исходные коды программ считаются похожими (т.е. один с большой вероятностью содержит плагиат другого), то и на уровне вышеописанных строковых представлений у них присутствуют существенные общие части. Причем эти общие части не обязательно должны быть непрерывны (в противном случае алгоритм было бы легко обмануть). Одним из наиболее ранних алгоритмов, использующих этот подход, описывается в работе [2], но из-за эвристики, которой пользуется данный алгоритм, его легко обойти плагиатору. В следующей части мы кратко опишем современные алгоритмы данной области.

Структурные алгоритмы, что следует из названия, используют саму структуру программы, обычно это или *граф потока управления*, или *абстрактное синтаксическое дерево*. Вследствие такого представления алгоритм сводит на нет многие возможные действия плагиатора. Но все алгоритмы этого класса являются крайне трудоемкими, поэтому не используются на практике. Единственным исключением может служить алгоритм из статьи [3]. Он использует представление программы в виде абстрактного синтаксического дерева и имеет, по утверждению авторов, квадратичную сложность, хотя проверить это не представилось возможным, так как авторы скрывают некоторые детали алгоритма. Но даже этого недостаточно на практике для наиболее распространенного случая поиска плагиата, когда у нас есть исходный код программы (а возможно – несколько) и большая база оригиналов, с которыми мы хотим его сравнить.

Семантические алгоритмы во многом похожи на структурные и текстовые. Например, в статье [4] используется представление исходного кода программы в виде графа с вершинами двух типов. Одни строятся из последовательности операторов, которым назначается определенная семантика (например, математическое выражение, цикл), другие задают отношение, в котором состоят соседние с ней вершины (например, вхождение). Между двумя вершинами первого типа обычно стоит вершина второго.

2.2. Алгоритм, основанный на выравнивании строк

Пусть у нас есть два исходных текста программ, представим их в виде строк токенов s и t соответственно (возможно, различной длины). Теперь мы можем воспользоваться методом *локального выравнивания строк*, разработанным для определения схожести ДНК [5]. Выравнивание двух строк получается с помощью вставки в них пробелов таким образом, чтобы их длины стали одинаковыми. Заметим, что существует большое количество различных выравниваний двух строк. Будем называть строки, полученные после выравнивания s и t, соответственно, s' и t'. Рассмотрим $s`_i$ и $t`_i$, стоимость их совпадения m, если только одни из символов пропуск, то – g, стоимость несовпадения d, где m – положительно, d и g – неположительны. Цена выравнивания – это сумма индивидуальных стоимостей всех пар $s`_i$ и $t`_i$, наибольшее значение этой целевой функции для всех i и j ($i <= j <= |s'|=|t'|$) на строках $s'[i..j]$ и $t'[i..j]$ – величина выравнивания.

Оптимальное выравнивание между двумя строками – максимальное значение целевой функции среди всех выравниваний. Это значение может быть вычислено с помощью динамического программирования.

Применение этого алгоритма в нашем случае выглядит так: получаем токенизированное представление *s* и *t* двух программ, делим вторую строку *t* на подстроки (секции), каждая из которых представляет модуль (функцию или процедуру) исходной программы. Для каждой секции и *s* получаем значение оптимального локального выравнивания, комбинируем результаты. Это позволяет алгоритму корректно обрабатывать перестановки модулей исходной программы. С деталями алгоритма можно ознакомиться в статье [5].

2.3. Жадное строковое замощение

Рассмотрим эвристический алгоритм получения *жадного строкового замощения* (The Greedy String Tiling, см. [6]). Он получает на вход две строки символов над определенным алфавитом (у нас это множество допустимых токенов), а на выходе дает набор их общих непересекающихся подстрок, близкий к оптимальному.

Алгоритм использует две эвристики:

- считается, что более длинные последовательные совпадения лучше, чем набор меньших и непоследовательных, даже если сумма длин последних больше.
- алгоритм игнорирует совпадения, длины которых меньше определенного порога.

Обе эвристики способствуют потере оптимального замощения, но результаты работы алгоритма обычно достаточно близки к нему для обнаружения плагиата. К тому же вторая эвристика способствует фильтрации шумов, т.е. небольших участков кода, которые случайно совпали в проверяемых исходных кодах и не являются плагиатом.

Асимптотика худшего случая для этого алгоритма $O(n^3)$, но на практике она значительно ниже $O(n^2)$, где n – длина строки, в которую переводят программу, используя токенизированное представление. Этот алгоритм в настоящее время успешно используется в детекторе плагиата JPlag [7].

2.4. Колмогоровская сложность в задаче нахождения плагиата

В работе [8] используется расстояние между последовательностями, основанное на теории информации (an information based sequence distance):

$$d(x,y) = 1 - \frac{K(x) - K(x \mid y)}{K(xy)},$$

где $K(x)$ – колмогоровская сложность последовательности x. Она показывает, сколько информации содержит последовательность x. По определению, $K(x)$ – длина самой короткой программы, которая на пустой ввод печатает строку x. Аналогично, $K(x \mid y)$ – это длина самой короткой программы, которая на ввод строки y печатает строку x. Можно отметить, что если y не содержит никакой полезной информации для понимания строения x, то $K(x \mid y) = K(x)$. Тогда выражение $K(x) - K(x \mid y)$ можно трактовать как, сколько y «знает» о x. Более подробно про колмогоровскую сложность можно посмотреть в [9].

Основное отличие приведенной метрики от других, используемых в задачах определения плагиата, состоит в ее универсальности: две программы, близкие относительно любой другой метрики, будут близкими и относительно данной [10]. Теоретически детектор, основанный на такой метрике, невозможно обмануть.

Чем ближе функция расстояния $d(x,y)$ к 0, тем больше общей информации содержат две программы x и y. Но, как известно, колмогоровская сложность не вычислима [9]. В статье [8] приводится эвристическое приближение $d(x,y)$, основанное на применении алгоритма сжатия к токенизированному представлению программы:

$$d(x,y) \approx 1 - \frac{Comp(x) - Comp(x \mid y)}{Comp(xy)},$$

где *Comp*(.) (*Comp*(.|.)) – длина сжатой (условно) строки, полученной из исходной с помощью какого-либо алгоритма сжатия. В работе [8] специально был разработан алгоритм сжатия, который удовлетворяет следующим специфическим требованиям.

- От традиционных алгоритмов сжатия очень часто требуют возможность выполнения в реальном времени или с линейной асимптотикой. Для нас же главное – качество сжатия, а скорость менее важна. Наш алгоритм может работать за сверхлинейное время, но должен достигать наилучшей степени сжатия для используемого типа данных.
- Для получения $d(x,y)$ нужно вычислить $Comp(x)$, условное сжатие x по y ($Comp(x|y)$) и $Comp(xy)$. Обычно программа, которая является плагиатом другой, содержит длинные, приблизительно совпадающие с фрагментами оригинальной программы блоки. Они возникают, когда злоумышленник копирует часть кода оригинальной программы и делает в нем простые изменения. Требуемый алгоритм сжатия должен эффективно справляться с такими вещами, поэтому применяется алгоритм семейства LZ (Lempel-Ziv, см. [11]), использующий найденные неточные совпадения.

Рассмотрим непосредственно процесс сравнения двух программ. Сначала производится их токенизация, далее запускается алгоритм TokenCompress, основанный на алгоритме сжатия LZ (см. [11]). Первым делом он находит длиннейшую неточно повторяющуюся подстроку (approximately duplicated substring), заканчивающуюся в текущем символе; кодирует ее указателем на предыдущее размещение и сохраняет информацию о внесенных поправках. Реализация отличается от классического LZ-сжатия некоторыми особенностями, связанными со спецификой задачи: классический алгоритм семейства LZ с ограниченным буфером может пропустить некоторые длинные повторяющиеся подстроки из-за ограничения на размер словаря (или скользящего окна) во время кодирования. Это не страшно для кодирования обычных текстов, где нужно, прежде всего, экономить память и уменьшать время исполнения, ведь потери качества сжатия будут крайне малы. Но в силу специфики задачи и выбранного алгоритма ее решения, использующего неточные совпадения, потеря даже небольшого количества длинных повторяющихся подстрок неприемлема.

В оригинальной статье [8] не приводится алгоритм подсчета условного сжатия x по y ($Comp(x|y)$), но, скорее всего, предполагается, что в начале алгоритма инициализируется словарь (буфер) всеми подстроками строки y, а дальше алгоритм остается без изменений.

Классические алгоритмы семейства LZ не поддерживают неточных совпадений. Данный же алгоритм ищет неточно повторяющиеся подстроки и кодирует их несовпадения. TokenCompress использует пороговую функцию, чтобы определить, выгоднее ли кодировать несовпадения или лучше воспользоваться альтернативами (кодировать части отдельно или вообще не производить кодирования некоторых частей). Соответственно, некоторые неточные повторы могут быть объединены в один с небольшим количеством ошибок (несовпадений). Неточно совпавшие пары подстрок (т.е. неточные повторы) записываются в файл и позже могут быть просмотрены человеком. На основе работы этого алгоритма считается величина $d(x,y)$, которая затем используется для определения, насколько вероятно то, что одна из этих программ является плагиатом другой.

2.5. Метод отпечатков

2.5.1. Основная идея

В этом алгоритме мы представляем токенизированную программу в виде набора отпечатков (меток, fingerprints), так чтобы эти наборы для похожих программ пересека-

лись. Этот метод позволяет организовать эффективную проверку по базе данных. Метод отпечатков можно представить в виде четырех шагов.

1. Последовательно хэшируем[1] подстроки токенизированной программы P длины k (фиксированный параметр).
2. Выделяем некоторое подмножество их хэш-значений, хорошо характеризующее P. Проделываем те же шаги для токенизированных программ $T_1, T_2 \ldots T_n$ и помещаем их выбранные хэш-значения в хэш-таблицу.
3. С помощью хэш-таблицы (базы) получаем набор участков строки P, подозрительных на плагиат.
4. Анализируем полученные на предыдущем шаге данные и делаем выводы.

Очевидно, что содержательная часть этого алгоритма – выбор k и шага 2. Число k ограничивает длину наименьшей подстроки, с которой может работать данный алгоритм. Если в двух программах есть общая подстрока длиной, например, два символа, то не обязательно, что мы нашли плагиат. Скорее всего, это совпадение – просто случайность и обусловлено малой мощностью нашего алфавита. Т.е. значение k может способствовать игнорированию «шума», но при больших значениях k мы можем проигнорировать настоящий случай плагиата, поэтому это число нужно выбирать с осторожностью.

Пусть наша хэш-функция – это h, а $h_1, h_2 \ldots h_{|P|-k+1}$ – последовательность значений хэш-функций, полученных на шаге 1. Рассмотрим некоторые возможные реализации шага 2.

- Наивный подход заключается в выборе каждого i-ого из n хэш-значений, но он неустойчив к модификациям кода. (Если мы добавим в начало файла один лишний символ, то получим совершенно другое подмножество хэш-значений после выполнения алгоритма.)
- Мы можем назначать метками p минимальных хеш-значений (см. [13]), их количество для всех документов будет постоянно. С помощью этого метода нельзя найти частичные копии, но он хорошо работает на файлах примерно одного размера, находит похожие файлы, может применяться для классификации документов.
- Манбер [14] предложил выбирать в качестве меток только те хеш-значения, для которых $h \equiv 0 \bmod p$, так останется только n/p меток (объем идентификационного набора для разных файлов будет отличаться, сами метки будут зависеть от содержимого файла). Однако в этом случае расстояние между последовательно выбранными хеш-значениями не ограничено и может быть велико. В этом случае совпадения, оказавшиеся между метками, не будут учтены.
- Метод просеивания (winnowing) [15] не имеет этого недостатка. Алгоритм гарантирует, что если в двух файлах есть хотя бы одна достаточно длинная общая подстрока, то как минимум одна метка в их наборах совпадет.

2.5.2. Метод просеивания

Чтобы быть эффективнее других реализаций метода отпечатков, рассматриваемый алгоритм должен гарантировать следующее.

- Если у двух токенизированных программ есть общая подстрока длиной как минимум t, то она будет найдена. Отметим, что если мы находим любой общий фрагмент, длина которого не меньше шумового порога (k), и он полностью лежит в другом общем фрагменте, то мы можем обнаружить этот больший фрагмент, используя данные о позиции, найденных совпадений за линейное время.
- Общие подстроки короче шумового порога игнорируются.

[1] Обычно используют хэш-функцию из алгоритма Карпа-Рабина поиска подстроки в строке [12].

Последний пункт гарантируется величиной параметра *k*. Теперь разберемся с первым пунктом. Очевидно, что он будет соблюдаться, если из каждых $(t-k+1)$ хэш-значений от идущих непосредственно последовательно подстрок будет выбрана хотя бы одна в качестве метки. Будем продвигать окно размера $w=(t-k+1)$ вдоль последовательности $h_1...h_n$, на каждом шаге перемещаем окно на одну позицию вправо. Назначаем меткой минимальное h_j в окне; если таких элементов несколько, то назначается самый правый из них.

Рассмотрим пример. Строке *abrakadabra* соответствует последовательность хеш-значений:

12, 35, 78, 3, 26, 48, 55, 12, 35

Пусть $k=2, t=4$, следовательно $w=t-k+1=3$. Выпишем последовательно содержание окон, полученных алгоритмом, жирным шрифтом выделены те хэш-значения, которые мы будем назначать метками:

(**12**, 35, 78), (35, 78, **3**), (78, 3, 26), (3, 26, 48), (**26**, 48, 55), (48, 55, **12**), (55, 12, 35)

Заметим, что если в двух последовательных окнах хэш-значения от одной и той же подстроки являются минимальными, то меткой назначается только одна из них (нет смысла хранить две последовательные абсолютно одинаковые метки), поэтому получим такой набор меток:

12, 3, 26, 12

Нужно отметить, насколько компактен набор идентификационных меток строки.

Показателем эффективности алгоритма может служить плотность *d* – доля хэш-значений, выбранных алгоритмом в качестве меток, среди всех хэш-значений документа. Для метода просеивания

$$d=\frac{2}{w+1}.$$

(см. оригинальную статью [15].) Некоторые другие реализации метода отпечатков тоже могут быть изменены так, чтобы соблюдались описанные выше требования, но для них показатель *d* в этом случае будет значительно выше, полученного для метода просеивания.

Как мы видим, выбор метки определяется только содержимым окна, такой алгоритм называется *локальным*. Любой локальный алгоритм выбора меток корректен. Действительно, если в двух файлах есть достаточно большая общая подстрока, то будут и одинаковые окна, а значит, будут назначены одинаковые метки. По ним определим, что в файлах есть совпадения. Это доказывает корректность приведенного алгоритма.

3. Результаты тестирования

Мы провели тестирование четырех алгоритмов, используя существующие их реализации: алгоритм, основанный на выравнивании строк (SIM [5]), жадное строковое замощение (JPlag [7]), алгоритм, использующий приближение колмогоровской сложности (SID [8]), метод отпечатков (MOSS [15]). Отметим, что у каждого детектора есть своя метрика схожести исходных кодов. Они достаточно близки для выбранных детекторов, поэтому мы можем их использовать, не опасаясь за чистоту результата. Будем считать, что у каждого детектора есть какое-то пороговое значение этой метрики (она у всех считается в процентах: 100 % – полное совпадение, 0 % – с точки зрения алгоритма полностью различны), при превышении которого мы можем считать, что имеем дело с плагиатом. Это позволяет нам игнорировать разницу в метрике схожести, когда уже понятно, что детектор с большой вероятностью локализовал случай плагиата и поэтому увеличивает чистоту эксперимента. Мы использовали четыре коллекции тестов, рассмотрим каждую из них подробно.

Простая коллекция. Содержит настоящий случай плагиата для небольшой программы, плагиат практически не пытались скрыть. Все детекторы обнаружили плагиат.

Коллекция с олимпиады по программированию. Она построена на основе программ, написанных во время различных олимпиад по программированию, и представляет собой подмножество программ, выложенных в открытый доступ на сайте http://neerc.ifmo.ru. Алгоритмически решения задач в большинстве случаев достаточно схожи, реализации же достаточно сильно зависят от стиля программирования конкретного участника. Все детекторы, кроме SIM, справились с задачей, ни один не посчитал плагиатом независимо написанные программы. Но стоит отметить, что для менее специфических программ, содержащих больше шаблонных решений, могли возникнуть проблемы. Детектор SIM использует алгоритм, основанный на выравнивании строк, поэтому на небольших программах он бывает неоправданно подозрителен, в 14 % случаев SIM посчитал плагиатом программы, написанные независимо.

Тестовая коллекция детектора SID. Здесь мы имеем дело с четырьмя видами искусственных изменений кода: перегруппировка переменных, различные виды рефакторинга кода, перегруппировка методов, случайные вставки операторов. На первых трех видах изменений кода все детекторы справились идеально. Для четвертого вида изменений мы имеем более сложную картину. Для небольшого количества вставок, 20 на 100 строк кода или 300 на 1100 строк, все детекторы находят плагиат. Если же количество вставок достигает 50 на 100 строк кода или 400 на 1100 строк, то все детекторы, кроме SID, перестают находить плагиат, что неудивительно из-за построения этих алгоритмов. Отметим, что даже SID в одном случае не смог обнаружить плагиат, что закономерно: его алгоритм просто более устойчив к данному способу сокрытия плагиата.

4. Анализ результатов тестирования

Все детекторы показали приемлемый результат на тестах, приближенных к реальным случаям плагиата, где не использовались средства автоматизированного сокрытия плагиата (или ими пытались сгенерировать код, по которому человек не сможет быстро определить, было ли применено автоматизированное средство сокрытия плагиата). Все же в сложных случаях можно утверждать, что будет лидировать SID с минимальным преимуществом. При использовании автоматизированных средств сокрытия плагиата, которые применяют в достаточно большом количестве случайные вставки, SID будет вне конкуренции. А так как случайные вставки прекрасно имитируют любые изменения, которые затрагивают одну и изредка несколько соседних строк, то можно быть уверенным, что алгоритм, на котором основан SID, справится с любыми случаями плагиата, где нет значительного изменения структуры кода**.**

Но все описанные нами алгоритмы, кроме метода отпечатков, не могут эффективно работать с большой базой, так как требуют проведения сравнений образца с каждым элементом базы. А метод отпечатков проводит сравнение образца со всей базой целиком, причем скорость на практике почти линейно зависит от количества элементов базы, относительно которых с определенной вероятностью образец содержит плагиат.

Существует работа [16], в которой предлагается использовать кластеризацию элементов базы, объединяя в кластер очень похожие исходные коды, а потом выбрать (или построить) у каждого кластера своего представителя и проводить сравнение только с ним. Это приведет к ускорению проведения поиска плагиата относительно базы для любого алгоритма, но, на наш взгляд, такой подход не даст на практике достигнуть скорости работы с базой метода отпечатков, а также может способствовать некоторому ухудшению качества поиска при неправильном выборе алгоритма.

Можно обобщить вышеописанную идею и проводить кластеризацию алгоритмом с большей полнотой (отношение количества найденных случаев плагиата ко всем слу-

чаям плагиата), но при этом, возможно, неприемлемо низкой точностью (доля настоящих случаев плагиата в найденных алгоритмом) для самостоятельного использования, а внутри кластера применить алгоритм с большей точностью, причем, возможно, даже несколько алгоритмов, но неустойчивых к разным способам сокрытия плагиата. Наиболее интересно, что для уточнения результатов внутри кластера мы можем применить алгоритм с достаточно высокой трудоемкостью, т.е. вышеупомянутые структурные или семантические методы.

5. Заключение

В настоящий момент существует достаточно много текстовых алгоритмов поиска плагиата, наиболее практичные и распространенные из них мы описали в этой статье, они используют разные эвристики, но основные черты этих алгоритмов неизменны (см. п. 2.1). На основе полученных результатов тестирования мы можем предположить, что существующие алгоритмы почти достигли предела для этого подхода по полноте поиска, остается только увеличивать точность, но для этого можно использовать сторонние алгоритмы или, например, вышеописанный подход, использующий кластеризацию.

Для метода отпечатков и подобных ему в работе с базой мы можем предложить определенным образом задавать вес каждому фрагменту исходного кода, имеющего отображение в хэш-таблице, например, в зависимости от частоты его встречаемости в базе. Ведь чем чаще фрагмент появляется в базе, тем больше вероятность того, что это шаблонное решение, которое в большинстве случаев считать плагиатом не следует. С помощью машинного обучения мы можем корректировать веса фрагментов в зависимости от того, считает ли пользователь текущий фрагмент плагиатом. Эти модификации должны существенно повысить точность поиска.

Насколько нам известно, подобные системы еще не реализованы на практике, и, тем более, их эффективность не была исследована. Единственный текстовый алгоритм, при улучшении которого будет получен существенный результат – это алгоритм, использующий приближение колмогоровской сложности [8], так как он более устойчив к случайным вставкам, чем другие алгоритмы и не имеет относительно них других слабых сторон.

Большой потенциал мы видим в усовершенствовании алгоритма из статьи [3] и создании подобного ему, но сохраняющего семантику.

Мы планируем реализовать и провести тестирование комбинированных методов (см. выше). Нам особенно интересно использование в качестве кластеризующего алгоритма атрибутно-подсчитывающих [1] методов, обогащенных алгоритмом из статьи [16], а в качестве действующего внутри кластера – одного из структурных или семантических алгоритмов, например, [3] или [4]. Мы ожидаем существенное улучшение результатов при таком подходе.

Сейчас только метод отпечатков [15] и алгоритм, основанный на сравнении абстрактных синтаксических деревьев, из работы [3] позволяют организовать базу исходных кодов программ, относительно которой можно искать плагиат в любом другом исходном коде за приемлемое время, а не проверять на плагиат образец с каждым элементом базы. Это происходит из-за того, что эти алгоритмы используют хэш-таблицы. Но в работе [3] утверждается, что нужна особая нетрадиционная «хэш-функция», примеры которой не сообщаются, поэтому можно выразить сомнения о ее эффективном применении на практике.

6. Благодарности

Авторы благодарят Юрия Лифшица за общее руководство. Исследования были поддержаны корпорацией Intel и грантом РФФИ № 06-01-00584-а.

Литература

1. Faidhi J.A.W., Robinson S.K. An Empirical Approach for Detecting Program Similarity within a University Programming Environment. // Computers and Education. 1987. 11(1). P. 11–19.
2. Heckel P. A Technique for Isolating Differences Between Files. // Communications of the ACM 21(4). April 1978. P. 264–268.
3. Baxter I., Yahin A., Moura L., Anna M.S., Bier L. Clone Detection Using Abstract Syntax Trees. // Proceedings of ICSM. IEEE. 1998.
4. Mishne G., M. de Rijke. Source Code Retrieval using Conceptual Similarity // Proceedings RIAO. Vaucluse. 2004. P. 539–555.
5. Huang X., Hardison R.C., Miller W. A space-efficient algorithm for local similarities. // Computer Applications in the Biosciences 6. 1990. P. 373–381.
6. Wise M.J. String similarity via greedy string tiling and running Karp-Rabin matching. // Dept. of CS, University of Sydney. December 1993.
7. Prechelt L., Malpohl G., Philippsen M. JPlag: Finding plagiarisms among a set of programs. // Technical Report No. 1/00, University of Karlsruhe, Department of Informatics. March 2000.
8. Chen X., Francia B., Li M., McKinnon B., Seker A. Shared Information and Program Plagiarism Detection. // IEEE Trans. Information Theory. July 2004. P. 1545–1550.
9. Li M., Vitanyi P. An introduction to Kolmogorov complexity and its applications. 2nd Ed., Springer, New York. 1997.
10. Li M., Chen X., Li X., Ma B., Vitanyi P. The similarity metric. // Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms. 2003. P. 863–872.
11. Ziv J., Lempel A. A universal algorithm for sequential data compression // IEEE Transactions on Information Theory 23. 1977. P. 337–343.
12. Karp R.M., Rabin M.O. Efficient randomized pattern-matching algorithms // IBM J. of Research and Development. 1987. 31(2). P. 249–260.
13. Heintze N. Scalable document fingerprinting. // In 1996 USENIX Workshop on Electronic Commerce, 1996.
14. Manber U. Finding similar files in a large file system. // Proceedings of the USENIX Winter 1994 Technical Conference. San Francisco. 1994. P. 1–10.
15. Aiken A., Schleimer S., Wikerson D. Winnowing: local algorithms for document fingerprinting. // Proceedings of ACMSIGMOD Int. Conference on Management of Data. San Diego. 2003. P. 76–85. ACM Press. New York, USA. 2003.
16. Moussiades L.M., Vakali A. PDetect: A Clustering Approach for Detecting Plagiarism in Source Code Datasets. // The Computer Journal Advance Access. June 24, 2005.

АНАЛИЗ ИЗМЕНЕНИЙ ПРОГРАММНОГО КОДА МЕТОДОМ КЛАСТЕРИЗАЦИИ МЕТРИК

Е.Г. Князев, Д.Г. Шопырин
Научный руководитель – к.т.н., доцент М.В. Тарасюк

В работе представлен метод анализа метрик изменений программного кода, основанный на многомерной статистической кластеризации. Метод позволяет производить классификацию изменений по типам путем интерпретации экспертом кластеров наборов метрик. Метод упрощает процедуру просмотра кода (code-review), создает предпосылки для улучшения процесса разработки программного обеспечения и делает возможным контроль типов изменений, вносимых на различных стадиях разработки.

1. Введение

1.1. Введение в MSR и DataMining

Добыча данных из программных репозиториев (mining software repositories, MSR) [1] заключается в предоставлении средств обработки неупорядоченной информации, содержащейся в программном репозитории, выделении значимых показателей из исходных исторических данных модификации программы. Добыча данных из программных репозиториев (хранилищ исходного кода) совмещает в себе методы традиционной добычи данных с методами анализа исходного кода программ и позволяет, среди прочего, добиться улучшения процесса разработки программы на основе анализа истории ее изменений в прошлом.

Традиционная *добыча данных* (data mining) – это технология анализа хранилищ данных, базирующаяся на методах и инструментах поддержки принятия решений [2].

Задачи, решаемые MSR, делятся на два круга: *поиск ассоциативных правил* и *отыскание общих характеристик изменений*. Задача поиска ассоциативных правил применяется в MSR для предсказания последствий произведенных изменений. Например, как изменится количество ошибок в программе при внесении определенного изменения, какие модификации будут произведены в дальнейшем и т. д. Часто изменения могут быть не полностью документированы, и тогда результаты работы методов могут оказаться неточными [3, 4]. Задача отыскания общих характеристик изменений освещена, например, в работе [5]. В частности, в ней ставится вопрос, каков наиболее общий тип изменения при исправлении ошибки. В настоящей работе предлагается метод, который, как и [5], можно отнести к задаче отыскания общих характеристик изменений MSR.

1.2. Задача классификации изменений

Многие современные системы контроля версий хранят историю исходного кода в терминах добавления и удаления строк. Однако для проведения глубокого анализа изменений кода требуется информация о семантике произведенных модификаций [6]. Для решения проблемы в данной работе ставится задача классификации изменений программного кода. Метод классификации изменений должен автоматически производить выделение семантически различных изменений на основе значений их метрик. В данной работе не предлагается ставить четкие критерии для каждого типа изменения, наоборот, задача ставится максимально широко, так как предполагается использование методов добычи данных при ее решении. Такой подход позволит адаптивно анализировать исходные данные, выделяя в числе общих типов, также специфичные для конкретного процесса разработки шаблоны.

1.3. Краткий список существующих методов

Среди существующих методов анализа программных репозиториев выделяются несколько групп [7].

- *Методы анализа служебной информации ревизий* репозиториев – метод поиска семантических зависимостей [8, 9], анализ «запросов на изменение» [10].

- *Эвристические методы* – метод предсказания изменений [3], метод автоматический классификации изменений [11], [12] посредством анализа комментариев.
- *Методы анализа синтаксиса* изменений – метод эвристического сравнения синтаксических деревьев версий [5], метод анализа разницы версий [13] при помощи встраиваемых в исходный код тегов.
- Методы, основанные на *добыче данных* – метод обнаружения совместно появляющихся изменений [4].

Более подробно эти методы будут рассмотрены ниже.

1.4. Предложенный метод

В данной работе предложен новый метод, основанный на кластеризации метрик изменений исходного кода. При помощи метода *k* средних МакКуина [2, 14] производится разбиение множества изменений на заданное число кластеров, каждый из которых соответствует определенному типу изменений.

1.5. Преимущества метода кластеризации метрик изменений

Преимущества предложенного метода состоят в следующем.

- *Объективность*: для анализа используется исходный код, а не комментарий, сопровождающий изменение. Оценка по исходному коду адекватна, в отличие от классификации комментариев к изменениям, ведь комментарии могут не в полной мере соответствовать характеру изменений [11].
- *Настраиваемость*: для оценки изменений выбирается множество метрик программного кода. В зависимости от того, по каким аспектам изменений предполагается группировка, выбираются те или иные метрики [15, 16] (см. п. 3.4).
- *Адаптивность*: при кластеризации задается лишь результирующее количество групп, следовательно, для каждого отдельно взятого исходного кода будут выделяться специфичные множества изменений, которые будут интерпретированы экспертом как те или иные группы изменений (см. п. 3.3).

1.6. Практическое применение

Представленный в работе метод успешно опробован на практике. Реализован программный инструмент кластеризации ревизий репозитория системы контроля версий *subversion* [17]. Метрики исходного кода, написанного на любом из языков *C, C++, C#, Java*, вычислялись при помощи инструмента *MSquared Resource Standard Metrics* [18]. Разработанный инструмент доказал эффективность практического применения метода кластеризации метрик изменений на проекте *Navi-Manager* [19].

1.7. Краткое содержание следующих разделов

В разделе 2 статьи приводится краткий обзор упомянутых выше методов MSR, решающих задачи классификации изменений и другие близкие задачи путем анализа изменений исходного кода. В разделе 3 описывается предложенный метод классификации изменений. В разделе 4 рассматривается пример практического применения инструмента классификации изменений при разработке программного продукта Navi-Manager – системы управления флотом.

2. Предшествующие исследования

Среди методов, применяемых в рамках MSR, можно выделить [7]:

- методы разбора информации, хранящейся системой контроля версий;
- эвристические методы;
- анализ синтаксиса изменений;
- методы добычи данных.

2.1. Методы разбора информации, хранящейся системой контроля версий

Метод, представленный в работах [8, 9], основан на анализе свойств изменений, хранящихся системой контроля версий для истории исходного кода. Производится поиск общих семантических зависимостей между классами с точки зрения добавления или модификации определенного класса. Классы, модифицируемые одним и тем же автором приблизительно в одно и то же время, предполагаются зависимыми.

В работе [10] изучаются изменения на уровне исходных файлов. Последовательные изменения группируются в так называемые «запросы на изменение». Исследуются вопросы: является ли целью «запроса на изменение» добавление новой функциональности или исправление ошибки, различны ли «запросы на изменение» на различных стадиях эволюции.

Методы, основанные на разборе служебной информации из системы контроля версий, не обладают достаточной мощностью. Исходный код и содержание комментария в таких методах, как правило, не анализируются. При данном подходе невозможно сделать выводов относительно типов произведенных изменений, их структуры. Но использование служебной информации системы контроля версий в комбинации с методами анализа исходного кода приводит к повышению качества результатов.

2.4. Эвристические методы

Анализ информации об изменениях системы контроля версий может быть улучшен с применением эвристики на основе информации, выделенной из исходного кода или модели исходного кода. Множество эвристик, основанных на идентификаторе разработчика, внесшего изменение, отношении вызова-использования-определения в коде, распределении кода по файлам предложено в работе [3]. Эвристики используются для предсказания вхождений – кандидатов на изменение – исходя из предположения, что изменено заданное вхождение.

В работе [11] предложен метод автоматической классификации, основанный на анализе комментариев к изменениям кода. В комментарии производится поиск слов, специфичных для каждого из типов изменений. Если вхождение слова найдено, изменение классифицируется как принадлежащее группе, описываемой этим словом. Например, слова «исправлено», «ошибка» характеризуют группу исправлений ошибок, а слова «добавлено», «реализовано» – группу добавления новой функциональности. Далее сравниваются результаты автоматической и экспертной оценки для тестового набора данных. При таком подходе согласованность результатов достигает 70 %.

Метод [11] основывается на анализе комментариев, однако иногда разработчики не всегда исчерпывающе и корректно описывают все произведенные модификации кода. Вообще, содержание комментария субъективно. Текст, написанный человеком, часто может быть интерпретирован различным образом. Поэтому на практике не всегда корректно судить о содержании изменений, основываясь лишь на комментариях к ним.

2.7. Методы анализа синтаксиса изменений

В работе [5] при помощи эвристического алгоритма сравнения синтаксических деревьев, построенных для последовательных версий исходных файлов, строится синтаксическая разница между версиями. Затем вычисляется, сколько функций и вызовов было добавлено, модифицировано, сколько условных выражений было изменено и т.д.

В работе [20] анализируется разница версий с помощью тегов, предварительно встроенных в код. Благодаря тегам обнаруживаются добавленные, измененные и удаленные синтаксические вхождения и вычисляются типы синтаксических изменений. В результате работы метода даются ответы на вопросы: добавились ли новые методы в определенный класс, есть ли изменения в директивах препроцессора, было ли модифицировано условие любого условного оператора и т.д.

Методы анализа синтаксиса изменений в большинстве своем сложны, зависят от конкретного языка программирования. Применение таких методов в чистом виде для анализа типов изменений оправданно только для простых изменений. В случае если структура изменения сложна, метод не позволит дать однозначный ответ на вопрос о его типе.

2.10. Методы, основанные на добыче данных

Методы добычи данных предоставляют множество технологий, имеющих потенциальное применение в MSR. В работе [4] ставится цель обнаружить совместно появляющиеся в программной системе изменения: когда отдельное вхождение было модифицировано (например, функция с именем *A*), какие еще функции будут также модифицированы (например, функции с именами *B, C*).

3. Метод кластеризации метрик изменений

В данной работе решается задача классификации *изменений программного кода* методом *кластеризации метрик изменений*. Согласно классификации методов MSR из [7] (см. п. 2), предложенный подход принадлежит к методам добычи данных.

3.1. Формализация задачи

Изменение программного обеспечения δ трактуется в работе как отображение множества исходных данных S в другое – множество модифицированных данных S^*:

$\delta : \mathrm{S} \xrightarrow{\delta} S^*$.

Изменения происходят по инициативе разработчика. В данной работе рассматриваются только изменения *исходного кода* программного обеспечения.

Согласно обозначениям систем контроля версий каждому состоянию сопоставляется неотрицательное целое число, *ревизия или версия исходных данных r*:

$\delta_r : \mathrm{S_r} \xrightarrow{\delta r} S_{r+1}$.

В ходе разработки, тестирования и поддержки программного обеспечения исходный код часто модифицируется согласно нескольким *шаблонам* (*паттернам*), например, таким как *реализация новой функциональности, рефакторинг, исправление ошибки* [21]. На практике принадлежность изменения к одному из шаблонов устанавливается путем просмотра содержания изменения. Эта задача трудоемка и требует высокой квалификации разработчика, так как нет четких критериев оценки типа изменения. В данной работе предлагается выделять типы изменений при помощи кластеризации их метрик.

Метрика программного обеспечения (software metric) – это мера *M*, позволяющая получить численное значение некоторого свойства программного обеспечения *S* или его спецификаций [15, 16]. Например, количество строк исходного файла, цикломатическая сложность, количество ошибок на строку кода, количество классов и интерфейсов, связность и другие.

Метрика изменения программного обеспечения определяется здесь по аналогии – это мера *M`*, характеризующая *изменение* δ программного обеспечения. Например, количество модифицированных строк кода, изменение цикломатической сложности, изменение количества классов и другие. Для аддитивных метрик справедливо соотношение:

$$M`\delta_r = M`(S_r, S_{r+1}) = MS_{r+1} - MS_r, \quad (1)$$

здесь *M* – метрика исходного кода, соответствующая метрике изменения *M`*, S_r – исходный код до изменения δ_r, S_{r+1} – исходный код после этого изменения.

Задача кластеризации метрик изменений состоит в построении множества *C*:

$$C = \{c_1, c_2, \ldots, c_n\},\ ci = \{\delta_r, \delta_k \mid M_D(\delta_r, \delta_k) < D_{min}\}, \quad (2)$$

здесь M_D – мера близости между объектами, называемая расстоянием, D_{min} – величина, определяющая меру близости для включения объектов в один кластер. M_D определяется следующим образом:

$M_D(\delta_x, \delta_y) = \rho<M`_1, M`_2, \ldots M`_n>(\delta_x, \delta_y) = \rho(<m_{1x}, m_{2x}, \ldots m_{nx}>, <m_{1y}, m_{2y}, \ldots m_{ny}>)$, (3)

где ρ – мера в пространстве R^n, $M`_i$ – метрика изменения, $m_{ix}$, $m_{iy}$ – значения метрик $M`_i$ изменений δ_x, δ_y соответственно. Применительно к задаче кластеризации метрик изменений ρ – мера в пространстве кластеризации R^n; $M`_i$ – отдельное измерение пространства кластеризации; m_{ix}, m_{iy} – координаты точек в пространстве кластеризации. В качестве меры точек ρ в пространстве R^m в работе выбирается мера Евклида:

$$\rho(< x_1, x_2, \ldots, x_n >, < y_1, y_2, \ldots, y_n >) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots + (x_n - y_n)^2} .$$

3.2. Описание метода

В работе для решения задачи кластеризации применяется метод k средних Мак-Куина [2, 14]. Пусть известно число кластеров n, выбраны метрики для оценивания изменений при кластеризации, а мера расстояния ρ между точками пространства кластеризации R^n принята евклидовой. Алгоритм кластеризации изменений следующий:

(1) Произвести начальное разбиение множества объектов $\{\delta_r\}$ случайным образом:

$C^0 = \{c_1, c_2, \ldots, c_n\}$, $c_i = \{\delta_r \mid \delta_r \notin c_j, j \neq i\}$.

(2) Принять номер итерации $l = 1$.

(3) Определить центры кластеров cc_i по формуле:

$$cc_i = \frac{\sum_r [\delta_r \in c_i] M`(\delta_r)}{\sum_r [\delta_r \in c_i]} .$$

(4) Обновить множества распределения объектов по кластерам $C^l = \{c_i\}$:

$c_i = \{\delta_r \mid M_D(\delta_r, cc_i) = min\ M_D(\delta_r, cc_k)\}$.

(5) Проверить условие: $||C^l - C^{l-1}|| < \varepsilon$,

где ε – точность разбиения. Если условие выполнено, то завершить процесс, иначе перейти к шагу (3) с номером итерации $l = l + 1$.

Вышеописанное позволяет автоматически разбить множество изменений на кластеры. В каждый кластер группируются наиболее схожие друг с другом изменения.

3.3. Экспертная интерпретация изменений

В данном разделе под интерпретацией понимается отображение множества кластеров C, полученных при помощи метода Мак-Куина, во множество типов изменений из постановки задачи:

$$I^n : C \xrightarrow{I^n} \{t_1, t_2, \ldots, t_n\},$$

где $\{t_j\}$ – множество типов изменений, вводящихся экспертом. Например, исправление ошибки, реализация новой функциональности, удаление функциональности.

Задача поиска интерпретации I^n в данной работе отдается эксперту. Эта задача значительно более проста, чем исходная, так как на практике имеет смысл различать лишь небольшое число типов изменений. Определение количества кластеров и интерпретация каждого кластера производится экспертом на основе выборочного анализа изменений, принадлежащих каждому кластеру. Анализируется изменение исходного кода и комментарий, сопровождающий изменение. В результате устанавливается, какому из типов t_j соответствует данный кластер c_i.

В случае если в один кластер попадают изменения различных типов, возможна одна из ситуаций.

- *Неверно выбрано количество кластеров для разбиения*. Количество кластеров выбрано меньше числа реально присутствующих типов изменений. В этом случае следует увеличить количество кластеров n при выполнении кластеризации.
- *В исходных данных присутствуют смешанные изменения*. На практике возможны изменения, содержащие в себе несколько типов модификаций. Однозначная классификация таких изменений затруднительна.

- *Набор метрик кластеризации неполон*. При кластеризации не учитывается один из значимых параметров изменений, который должен влиять на изменение. Необходимо сконструировать соответствующую метрику, опытным путем проверить ее влияние на результат разбиения и включить в набор метрик кластеризации.
- *Типы изменений неотличимы с точки зрения программного кода*. Существуют типы изменений, которые невозможно отличить путем анализа программного кода. Необходима дополнительная информация для принятия решения. Такие типы следует либо рассматривать как один, либо ввести в процесс кластеризации дополнительные (не основанные на программном коде) метрики. Задача типизации таких изменений находится за рамками данной работы.

Необходимо заметить, что процесс интерпретации кластеров изменений может быть недетерминированным, так как выполняется человеком.

3.4. Выбор метрик для кластеризации

В формуле (3) п. 3.1 для меры близости между объектами M_D, используемой в методе кластеризации, присутствует набор метрик изменений $M`_i$. Выбор этих метрик должен быть обоснован интересующими эксперта аспектами классификации. Селекция конкретных метрик осуществляется путем анализа их влияния на результат кластеризации. Процесс выбора метрик $M`_i$ для кластеризации заданного набора типов изменений $\{t_j\}$ состоит в следующем.

(1) Формируется обучающая выборка – статистическая выборка изменений S, по которой будет строиться набор метрик.
(2) Экспертом классифицируются изменения обучающей выборки S по типам $\{t_j\}$.
(3) Выбирается произвольная метрика изменения $M`^*$, предположительно связанная с различением набора типов $\{t_j\}$ изменений.
(4) Задается уровень значимости статистического критерия. Для данного уровня значимости проверяется статистическая гипотеза об отсутствии связи типов выбранных изменений и значения метрики $M`^*$ для данных изменений. Если гипотеза не подтверждается, тогда метрика $M`^*$ включается в набор кластеризации.
(5) Производится кластеризация изменений выборки S с набором выбранных метрик.
(6) Для заданного критического уровня значимости производится проверка статистической гипотезы об отсутствии связи автоматической и ручной классификации [22]. Если гипотеза подтверждается, то продолжить алгоритм с пункта (3), иначе конец алгоритма.

После окончания работы алгоритма заданный набор метрик используется для кластеризации генеральной совокупности изменений.

4. Пример практического применения метода автоматической классификации

В качестве иллюстрации работы метода кластеризации метрик изменений на практике в текущем разделе приводятся результаты применения метода на проекте Navi-Manager [19]. Navi-Manager – это система управления флотом, состоящая из серверной и клиентской частей. Система преимущественно реализована на языке C#. Общее количество строк кода – более 100 000.

Все оценки и выводы делаются на основании тестового набора. Все действия производятся согласно алгоритму из п. 3.4.

4.1. Выбор метрик

(1) В данном разделе анализируется набор из 29 случайным образом выбранных изменений, произведенных в проекте Navi-Manager в течение 6 месяцев разработки.
(2) Изначально экспертом выбирается следующий набор типов изменений: добавление новой функциональности, удаление функциональности, исправление ошибки, рефакторинг, косметические изменения.

(3) Выбираются следующие метрики, предположительно связанные с различением данных типов изменений:

- eLOC (effective lines of code) – эффективное количество строк кода;
- CC (cycolmatic complexity) – цикломатическая сложность;
- IC (interface complexity) – интерфейсная сложность (общее количество параметров во всех методах);
- C/S (classes/structs) – количество классов и структур [23].

Метрики изменений строятся как аддитивные при помощи формулы (1).

(4) Уровень значимости статистического критерия выбирается равным 0.05.

(5) В табл. 1 приведены сравнительные результаты экспертной и автоматической классификации для выбранного набора изменений.

Ревиз.	Эксп.	Класт.	eLOC	CC	IC	C/S	Комментарий
12883	удал.	1	-112	-28	-39	-8	Перенос файлов между проектами, удаление ненужных.
16742	реф.	2	-4	5	18	0	Переделан на общую логику процессор InmarsatD+.
16743	реф.	2	-9	3	8	0	Обобщена обработка GlobeWireless.
17135	реф.	2	-6	-1	0	0	Обобщена логика FindOrCreateUserSource.
17404	реф.	2	-11	0	-6	0	Переведены службы и тесты на использование контекстной сессии.
17551	реф.	2	-9	-2	-5	-1	Убран дублирующийся класс DictionaryHelper.
18048	реф.	2	-6	-2	0	1	Рефакторинг работы с интервалами.
18142	реф.	2	-13	5	13	0	Реализована по-новому отрисовка меток.
18281	удал.	3	-48	-9	-6	-1	Уменьшено дублирование.
12955	нов.	4	3	0	0	0	Отчеты по сетям дополнены полями.
12956	косм.	4	1	0	0	0	Косметика.
16515	косм.	4	0	0	0	0	Убран реализованный TODO.
17613	испр.	4	1	0	0	0	Увеличен таймаут PositionDiscarder до 5 минут.
17778	испр.	4	2	0	0	0	Правка неточностей и опечаток.
17929	испр.	4	4	0	0	0	Исправлено несколько логических ошибок в работе TrackAgent.
17970	косм.	4	0	0	0	0	Переименован ITrack-Fetch.Interval в DestInterval.
18187	косм.	4	0	0	0	0	В CompoundTrack добавлен TODO.
18273	испр.	4	2	1	3	0	Поправлена ошибка при проверке флагов рисования элемента меню.

Таблица 1. Результаты автоматической классификации и экспертной интерпретации изменений из проекта Navi-Manager

18368	косм.	4	0	0	0	0	Убран бесполезный комментарий.
18382	нов.	4	2	0	0	0	При загрузке устанавливаются сохраненные значения для свойств trackAgent.
18431	испр.	4	3	1	0	0	Треки перестали отключаться при изменении промежутка времени в Playback.
15115	испр.	5	10	1	1	0	Поправлена установка vessel.LastReportTime.
15191	нов.	5	13	1	0	0	1. Добавлены блоки try..catch в методы OnStart, OnStop сервиса.
16580	нов.	5	12	4	0	0	Добавлен перенос max LastReportTime, LastSourceId в логику слияния кораблей.
16899	нов.	5	7	2	2	0	Добавлена инверсия цвета текста в колонку L в Vessel List View.
18390	нов.	5	15	6	3	0	Проверяется правильность ввода значений TrackMarksPeriod.
16337	нов.	6	26	6	5	0	Точечные объекты теперь экспортируются/импортируются в AddInfo как Text.
18421	нов.	6	22	0	0	0	На главную форму добавлено меню погоды.
16443	нов.	7	33	4	5	0	В диалог опций добавлена новая опция.

Таблица 1 (продолжение). Результаты автоматической классификации и экспертной интерпретации изменений из проекта Navi-Manager

В колонках слева направо представлены: ревизия изменения в системе контроля версий, результат экспертной классификации, результат автоматической кластеризации на основе метрик изменений, значения метрики изменения эффективного количества строк кода, изменения цикломатической сложности, изменения интерфейсной сложности, изменения количества классов и структур, комментарий к изменению.

Количество выбранных кластеров в таблице – 7, набор интерпретируемых типов изменений – новая функциональность («нов.»), удаление функциональности («удал.»), исправление ошибки («испр.»), рефакторинг («реф.»), косметическое изменение («косм.»). В качестве пояснения следует заметить, что число кластеров выбрано большим, чем количество типов изменений, так как при интерпретации несколько кластеров относятся к одному типу (например, кластеры 1 и 3, кластеры 6 и 7).

(6) В табл. 2 приведены результаты проверки статистической гипотезы об отсутствии связи автоматической и экспертной классификации. Дополнительно приведены оценки зависимости отдельных метрик изменений и экспертной классификации.

Автоматический метод Экспертный метод	eLOC	CC	IC	C/S	Кластеризация на основе метрик
Экспертная классификация	0,493511 0,003259	0,343601 0,034002	0,224454 0,120888	0,379072 0,021282	0,588552 0,000392

Таблица 2. Связь метрик с экспертной классификацией. Приведены значения коэффициента Пирсона и уровни значимости

Статистическая гипотеза об отсутствии связи между экспертной классификацией и кластеризацией на основе выбранных метрик таким образом отвергается (уровень значимости ~ 0,0004 ниже заданного критического уровня 0,05, величина коэффициента Пирсона больше 0,5: ~ 0,59). Следовательно, нет необходимости продолжать выбор метрик для получения значимых результатов кластеризации на основе выбранных метрик.

4.2. Улучшение результатов

Статистические гипотезы об отсутствии связи между отдельно взятыми метриками и экспертной классификацией также отвергаются во всех случаях, кроме метрики IC. Подтвержденная гипотеза об отсутствии связи между значениями метрики IC и экспертной классификацией говорит о том, что метрика IC не влияет на результат, и ее можно исключить. Возможен также анализ корреляционной связи событий появления отдельных групп и кластеризации на основе выбранных метрик. В табл. 3 приведены результаты вычисления таких величин.

Автоматический метод Экспертный метод	Кластеризация на основе метрик
Новая функциональность	0,668393 0,000037
Рефакторинг	-0,66147 0,000047
Исправление ошибки	0,110010 0,284986
Экспертная классификация	0,588552 0,000392

Таблица 3. Связь экспертных типов с автоматической классификацией. Приведены значения коэффициента Пирсона и уровня значимости

Статистическая гипотеза о независимости появления экспертного типа в данной таблице и результата автоматической классификации в случае типа «исправление ошибки» подтверждается. Следовательно, выбранный набор метрик не в состоянии различать тип изменения «исправление ошибки». Для обнаружения этого типа изменений потребуется дополнительная информация, например, из системы учета ошибок.

Для кластера 4 (см. табл. 1) не наблюдается хорошего согласования экспертной оценки с автоматической классификацией. Этот кластер содержит модификации, отнесенные экспертом к различным типам: новая функциональность, рефакторинг, исправление ошибки, косметические изменения.

После отказа от отдельного класса исправления ошибок и от метрики IC и последующей кластеризации получены результаты, представленные в табл. 4.

Рев.	Эксп.	Класт.	eLOC	CC	C/S	Комментарий
12956	косм.	1	1	0	0	Косметика.
16515	косм.	1	0	0	0	Убран реализованный TODO.
17613	нов.	1	1	0	0	Увеличен таймаут PositionDiscarder до 5 минут.
17970	косм.	1	0	0	0	Переименован ITrackFetch.Interval в DestInterval.
18187	косм.	1	0	0	0	В CompoundTrack добавлен TODO.
18368	косм.	1	0	0	0	Убран бесполезный комментарий.
17135	реф.	2	-6	-1	0	Обобщена логика FindOrCreateUserSource.
17551	реф.	2	-9	-2	-1	Убран дублирующийся класс DictionaryHelper.
18048	реф.	2	-6	-2	1	Рефакторинг работы с интервалами.
16743	реф.	3	-9	3	0	Обобщена обработка GlobeWireless.
18142	реф.	3	-13	5	0	Реализована по-новому отрисовка меток.
12955	нов.	4	3	0	0	Отчеты по сетям дополнены полями.
17778	нов.	4	2	0	0	Правка неточностей и опечаток.
17929	нов.	4	4	0	0	Исправлено несколько логических ошибок в работе TrackAgent.
18273	нов.	4	2	1	0	Поправлена ошибка при проверке флагов рисования элемента меню.
18382	нов.	4	2	0	0	При загрузке устанавливаются сохраненные значения для свойств trackAgent.
18431	нов.	4	3	1	0	Треки перестали отключаться при изменении промежутка времени в Playback.
15115	нов.	5	10	1	0	Поправлена установка vessel.LastReportTime.
15191	нов.	5	13	1	0	Добавлены блоки try..catch в методы OnStart, OnStop сервиса.
16580	нов.	5	12	4	0	Добавлен перенос max LastReportTime, LastSourceId в логику слияния кораблей.
16899	нов.	5	7	2	0	Добавлена инверсия цвета текста в колонку L в Vessel List View.
18390	нов.	5	15	6	0	Проверяется правильность ввода значений TrackMarksPeriod.
16337	нов.	6	26	6	0	Точечные объекты экспортируются/импортируются в AddInfo как Text.
18421	нов.	6	22	0	0	На главную форму добавлено меню погоды.
16742	реф.	7	-4	5	0	Переделан на общую логику процессор InmarsatD+.

Таблица 4. Результаты автоматической классификации и экспертной интерпретации изменений из проекта Navi-Manager

Состав колонок в табл. 4 аналогичен составу колонок табл. 1.

Количество выбранных кластеров – 7, интерпретируемые типы изменений – новая функциональность или правка ошибок («нов.»), удаление функциональности («удал.»), рефакторинг («реф.»), косметическое изменение («косм.»).

Проверка статистической гипотезы (без учета дробления кластеров для одного типа изменения) доказала, что качество кластеризации при введении новой метрики значительно улучшено: коэффициент Пирсона = 0,89, $P = 8{,}3*10^{-10}$. Согласованность экспертных данных с данными кластеризации *на данном примере* составляет 96 %.

5. Заключение

В данный момент методы добычи данных из хранилищ исходного кода слабо проработаны. Имеет место эпизодичность их практического внедрения в связи с недолгой историей исследований в этой области. Целью разработки нового метода было внести упорядоченность в работу с историей изменений программного кода.

Предложенный в работе метод предполагает участие эксперта в процессе кластеризации. Задача определения количества кластеров решается опытным путем, а именно, выполнением алгоритма кластеризации над исходными данными с несколькими значениями *n* (см. п. 3.2). Проблема выбора числа групп для кластеризации описана, например, в [2].

Как уже упоминалось в п. 1, преимуществом предложенного в работе метода, отличающим его от других методов (см. п. 2), является адаптивность, достигаемая за счет выбора метрик, соответствующих требуемому разбиению (см. п. 3.4).

Классификация может применяться для поиска дестабилизирующих программную систему изменений, оценки эффективности реализации проекта и других задач. Укажем варианты использования автоматической классификации изменений.

- *Упрощение процедуры просмотра кода*. Поиск добавленной возможности проще производить по списку изменений, добавляющих новую функциональность.
- *Возможность запрета изменений на определенных этапах разработки*. На этапе тестирования продукта может быть запрещена реализация новой функциональности.
- *Контроль процесса разработки программного обеспечения при помощи анализа соотношений вносимых изменений*. Появляется возможность оценивать скорость и качество разработки по соотношению внесенных исправлений ошибок, реализации новой функциональности и произведенных рефакторингов.

Практическое применение метода выявило проблему *смешанных изменений* (см. п. 3.3), сочетающих в себе разнородные модификации кода. Настоящим методом не всегда возможна корректная классификация таких изменений. Надо заметить, что наличие смешанных изменений на практике нежелательно и даже вредно. При их наличии усложняется процедура просмотра кода и другая работа с историей программного продукта. Обнаружение смешанных изменений в процессе кластеризации – предмет дальнейших исследований.

6. Литература

1. Zimmermann T. Knowledge Collaboration by Mining Software Repositories // Proceedings of the 2nd International Workshop on Supporting Knowledge Collaboration in Software Development (KCSD 2006). Tokyo, Japan, 2006, P. 64–65.
2. Баргесян А.А., Куприянов М.С., Степаненко В.В., Холод И.И. Технологии анализа данных: Data Mining, Visual Mining, Text Mining, OLAP. 2-е издание. СПб: БХВ-Петербург, 2007. 375 с.

3. Hassan A.E., Holt R.C. Predicting Change Propagation in Software Systems // Proceedings of 20th IEEE International Conference on Software Maintenance (ICSM'04). Chicago, Illinois, September 11–14, 2004. P. 284–293.
4. Zimmermann T., Wei.gerber P., Diehl S., Zeller A. Mining Version Histories to Guide Software Changes // Proceedings of 26th International Conference on Software Engineering (ICSE'04). Edinburgh, Scotland, United Kingdom, May 23–28, 2004. P. 563–572.
5. Raghavan S., Rohana R., Podgurski A., Augustine V. Dex: A Semantic-Graph Differencing Tool for Studying Changes in Large Code Bases // Proceedings of 20th IEEE International Conference on Software Maintenance (ICSM'04). Chicago, Illinois, September 11–14, 2004. P. 188–197.
6. Fluri B., Gall H. Classifying Change Types for Qualifying Change Couplings // Proceedings of the International Conference on Program Comprehension (ICPC). Athens, Greece. June 2006, P. 35–45.
7. Kagdi H., Collard M., Maletic J. Towards a Taxonomy of Approaches for Mining of Source Code Repositories // ACM SIGSOFT Software Engineering Notes, Proceedings of the 2005 international workshop on Mining software repositories MSR '05. St. Louis, Missouri, 2005. P. 1–5.
8. Gall H., Hajek K., Jazayeri M. Detection of Logical Coupling Based on Product Release History // Proceedings of 14th IEEE International Conference on Software Maintenance (ICSM'98). Bethesda, Maryland. March 16–19, 1998. P. 190–198.
9. Gall H., Jazayeri M., Krajewski J. CVS Release History Data for Detecting Logical Couplings // Proceedings of Sixth International Workshop on Principles of Software Evolution (IWPSE'03). Helsinki, Finland. September 01–02, 2003. P. 13–23.
10. German, D. M. An Empirical Study of Fine-Grained Software Modifications // Proceedings of 20th IEEE International Conference on Software Maintenance (ICSM'04). Chicago, Illinois. September 11–14, 2004. P. 316–325.
11. Hassan A.E., Holt. R.C. Source Control Change Messages: How Are They Used And What Do They Mean? 2004. Draft Available Online: http://www.ece.uvic.ca/~ahmed/home/pubs/CVSSurvey.pdf
12. Mockus A., Votta L.G. Identifying reasons for software change using historic databases // Proceedings of the International Conference on Software Maintenance (ICSM). San Jose, California, 2000. P. 120–130.
13. Maletic J.I., Collard M.L. Supporting Source Code Difference Analysis // Proceedings of IEEE International Conference on Software Maintenance (ICSM'04). Chicago, Illinois. September 11–17, 2004. P. 210–219.
14. Мандель И.Д. Кластерный анализ. М.: Финансы и статистика, 1988. 176 с.
15. Орлов С.А. Технологии разработки программного обеспечения: Учебник для вузов. 3-е изд. СПб: Питер. 2004. 528 с.
16. Липаев В.В. Выбор и оценивание характеристик качества программных средств: Методы и стандарты. М.: Синтег. 2001. 224 с.
17. http://subversion.tigris.org/
18. http://msquaredtechnologies.com/m2rsm/index.htm
19. http://www.transas.com/telematics/products/navi-manager/default.asp
20. Collard M.L. Meta-Differencing: An Infrastructure for Source Code Difference Analysis. Kent State University, Kent, Ohio USA. Ph.D. Dissertation Thesis, 2004.
21. Фаулер М. Рефакторинг: улучшение существующего кода. СПб: Символ-Плюс, 2003. 432 с.
22. Вадзинский Р.Н. Справочник по вероятностным распределениям. СПб: Наука. 2001. 294 с.
23. http://msquaredtechnologies.com/m2rsm/docs/rsm_metrics_narration.htm

СОЗДАНИЕ АВТОМАТНЫХ ВЕБ-ПРИЛОЖЕНИЙ

С.А. Сытник
Научный руководитель – д.т.н., профессор В.Г. Парфенов

В статье приводится пример простого веб-приложения, в котором история переходов между веб-страницами является важной. Указываются недостатки традиционного подхода к его реализации, предлагается автоматный подход.

Введение

Веб-приложения, являясь поставщиками различных сервисов, богатыми источниками информации и удобными средствами коммуникации, прочно вошли в современную жизнь. С ними работают как на производительных рабочих станциях, так и на мобильных телефонах.

Создавая веб-приложение, разработчик пользуется большой свободой в выборе способа его реализации. Это имеет много положительных сторон. Однако, когда приложение становится достаточно объемным, с большим количеством переходов между страницами, возникают сложности различного характера. Одна из них – будем называть ее «проблема контекста» – играет большую роль в том случае, когда поведение приложения зависит от пути перемещения пользователя по веб-страницам – другими словами, когда важна история переходов.

Эта проблема может быть решена при применении традиционного подхода к разработке веб-приложения, однако мы покажем, что структура приложения при этом может быть довольно сложной.

В настоящей статье предлагается автоматный подход к разработке, основанный на SWITCH-технологии [1], разрешающий проблему контекста. Программы, написанные при помощи автоматного подхода, сосредотачивают логику в едином месте программы, хорошо читаемы и легко модифицируемы. Этим можно объяснить то обстоятельство, что автоматный подход становится все более популярным как при программировании в общем, так и при веб-программировании. Автоматный подход, например, описан при разработке javascript-компонентов в работе Э. Принга [2].

В направлении использования автоматного подхода к разработке веб-приложений было проведено некоторое исследование [3]. При этом для реализации веб-приложений использовалось программное средство UniMod [4], в основе которого также лежит SWITCH-технология.

1. Проблема контекста в веб-приложениях

Действия, совершаемые веб-сервером при обработке запроса, очевидно, зависят от того, какая веб-страница запрашивается и какие параметры запроса ему были переданы. В общем случае действия зависят также от состояния запрашиваемой страницы и от истории предшествующих переходов по другим страницам до попадания на данную страницу:

$$actions = f(request, page, state, history),$$

где *actions* – совершаемые действия, f – некоторая зависимость, *request* – совокупность параметров запроса, *page* – текущая страница, за *state* мы обозначили ее состояние, а *history* – предшествующая история переходов по другим веб-страницам.

При программировании веб-приложения эту зависимость приходится учитывать. Будем называть это задачей учета контекста. При этом под контекстом условимся понимать совокупность четырех обозначенных сущностей (*request*, *page*, *state* и *history*), от которых зависят выполняемые при поступлении запроса действия:

$$actions = f(context),$$

context = (*request*, *page*, *state*, *history*).

Задача учета контекста может возникнуть в следующих трех случаях.

1) Переход на данную страницу возможен с некоторых других (двух или более) страниц, например, с целью ее повторного использования.
2) Возможен переход с некоторой страницы на ту же самую страницу, например, в результате возникшей ошибки ввода пользователя.
3) В зависимости от истории переходов до попадания на данную страницу требуется совершить некоторый переход (например, возврат на «вызывающую» страницу) и, возможно, некоторые различные действия.

В каждом из этих случаев веб-серверу может потребоваться совершить некоторые действия (рис. 1).

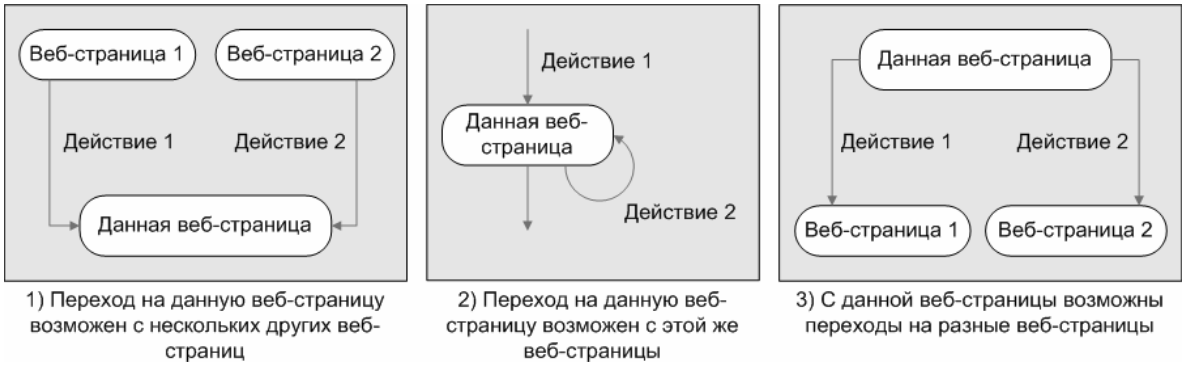


Рис. 1. Случаи возникновения задачи учета контекста

Проблема контекста заключается в том, что не существует метода программирования веб-приложений, описывающего, как можно удобно решить задачу учета контекста. Другими словами, неясно, каким образом можно учесть внутреннее состояние страницы и предшествующую историю посещения других страниц, чтобы совершить нужные действия. Вследствие этого задача контекста традиционно решается бессистемным введением различных «флагов». Например, если логика приложения должна «понять», с какой страницы совершен переход на данную страницу (случай 1, рис. 1), а потом использовать эту информацию для выбора перехода (случай 3, рис. 1), то программист должен предусмотреть для этого специальный «флаг», указывающий, с какой веб-страницы произошел переход.

2. Автоматный подход к реализации

Использование «флагов» совершенно не формализовано и может привести к многочисленным логическим ошибкам при проектировании приложения, в особенности при его усложнении. Попытаемся избавиться от них. Это представляется весьма актуальным, так как логика даже простых веб-приложений обычно не выглядит тривиальной.

Используя SWITCH-технологию, мы можем представить веб-приложение в виде конечного автомата, где каждой странице соответствует одно состояние, имеющее внутри себя выходное воздействие – формирование и выдачу веб-страницы.

Событиями (e) будут являться ввод пользователем адреса в строке веб-браузера и нажатия им на ссылки и кнопки на веб-страницах. При запросе к веб-серверу также будут передаваться другие параметры, используемые при вычислении входных переменных (x) и выходных воздействий (z). Это автоматически позволит нам избавиться от «флагов», так как эту задачу возьмут на себя состояния и переходы между ними.

Для сохранения состояния автомата (y) воспользуемся возможностью хранить данные текущей сессии пользователя на сервере. Эта возможность поддерживается

всеми современными средами разработки, в том числе Java Servlets или PHP [5, 6]. Однако при необходимости (к примеру, если мы работаем на кластере из нескольких серверов, когда сессия недоступна), ее можно эмулировать самостоятельно через параметры запроса.

Структуру и направления обмена данными для автоматного подхода описывает следующая схема (рис. 2).

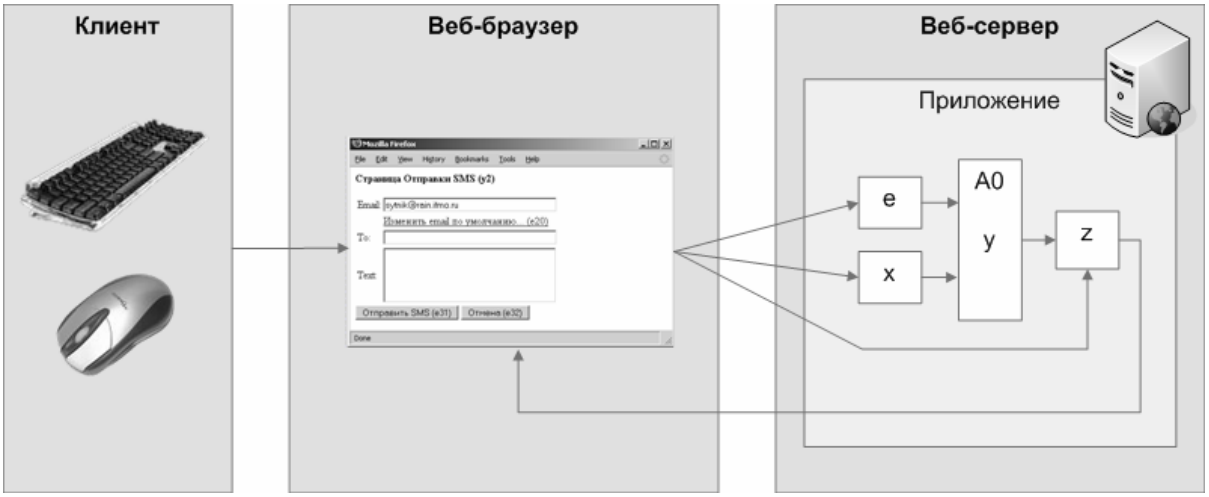


Рис. 2 Структурно-функциональная схема работы автоматного веб-приложения

Для того чтобы иметь возможность разрешать проблему контекста, мы можем воспользоваться следующими приемами:

1. разделить состояние автомата, соответствующее одной веб-странице, на несколько подсостояний; при этом изначальное состояние становится гиперсостоянием в общим выходным воздействием (вывод страницы);
2. явно выделить условия переходов, зависящих от контекста, в отдельные переходы между состояниями конечного автомата.

Проиллюстрируем применение автоматного подхода к разработке веб-приложения на простом примере.

3. Пример

Пусть требуется разработать веб-приложение, реализующее функциональность отправки SMS-сообщений. Для того чтобы отправить сообщение, пользователю необходимо указать номер телефона получателя и ввести текст сообщения. Также потребуем от пользователя указания email-адреса отправителя. Ввиду того, что этот email-адрес может изменяться достаточно редко, следует предусмотреть возможность его редактирования и запоминания для последующей подстановки «по умолчанию» в форму отправки сообщения. Для удобства пользователя следует предоставить эту возможность как независимо, так и в процессе отправки SMS-сообщения.

Диаграмма использования этого приложения имеет довольно простой вид (рис. 3). Следуя этой диаграмме, спроектируем приложение таким образом, чтобы оно состояло из трех веб-страниц, схематично представленных на рис. 4. Назначение этих страниц заключается в следующем.

1) «Главная страница» предоставляет пользователю возможность перейти либо к вводу email по умолчанию с последующим возвратом на главную страницу, либо к отправке SMS-сообщения.

2) Страница «настройки» отображает текущий «email по умолчанию» и разрешает его отредактировать.
3) При попадании на страницу «отправка SMS» текущий «email по умолчанию» автоматически подставляется в поле «email». Затем пользователь может изменить «email», ввести номер отправителя «to», текст сообщения «text» и отправить SMS. Также имеется возможность воспользоваться ссылкой «изменить email по умолчанию…». При этом пользователь попадет на страницу настроек, где он сможет, изменив email по умолчанию, вернуться к странице отправки SMS и продолжить редактирование (при этом «email» должен измениться на новое значение «email по умолчанию» автоматически).

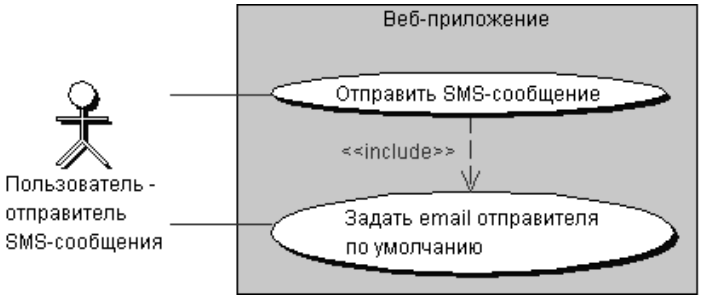


Рис. 3. Диаграмма использования веб-приложения

Рис. 4. Схематичное представление веб-страниц приложения

На рис. 5 представлена схема связей автомата, а на рис. 6 – диаграмма переходов автомата. Код программы строится по диаграмме переходов формально и изоморфно.

Выделим достоинства и недостатки автоматного подхода.

Достоинства приведенного подхода:

1) Структура программного кода, построенного по диаграмме переходов, обладает достаточно хорошей читаемостью, отсутствием дублирования логики в разных местах программы; при этом добавление новых функций не требует большого изменения в коде программы, а реализация поддержке сложной условная логики имеет достаточно простой вид. Такой код хорошо поддается поддержке и модификации.
2) Отсутствие некоторых проблем, присущих реализации при помощи традиционного подхода, а именно:
    a. отсутствие «флагов»;
    b. все переходы явно указывают, с какой страницы на какую происходит переход и какие действия при этом совершаются. Это упрощает добавление новых страниц по сравнению с традиционным подходом;

c. логика, а, следовательно, и установка выходных данных, необходимых для формирования страницы, собрана в одном месте, что упрощает восприятие программы.

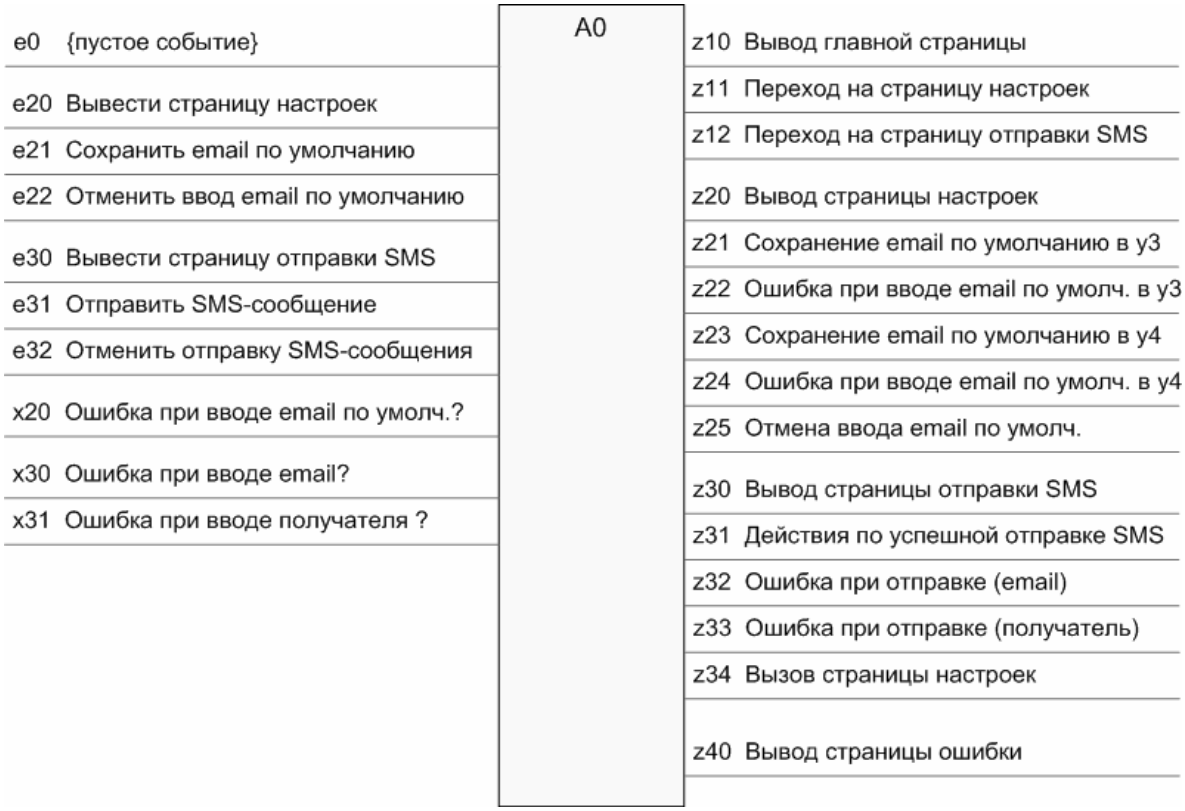


Рис. 5. Схема связей

3) Программа реализована в строгом соответствии со SWITCH-технологией, а значит, к ней может быть применен весь построенный для нее аппарат, как существующий в настоящее время, так и, возможно, полученный в будущем. Этому способствовал тот факт, что мы избежали использования так называемого автомата с глубокой историей, для которого такой аппарат фактически отсутствует.
4) Предложенная SWITCH-технологией нотация гиперсостояний нашла свое новое применение при описании веб-страниц. Одной веб-странице может соответствовать несколько состояний, но, если мы внесем их в общее для них гиперсостояние, соответствующее всей веб-странице, это облегчит понимание диаграммы переходов автомата. Это немаловажно, учитывая взаимнообратное соответствие диаграммы переходов и кода программы. Хотя гиперсостояние не имеет ни одного перехода, оно, тем не менее, выполняет не только функцию группировки состояний, но и содержит логически общие выходные воздействия. Таким общим воздействием является запрос на формирование веб-страницы.

Недостатки подхода:

1) Нет реентерабельности страниц. Хотя опыт показывает, что в веб-приложениях «вызов» веб-страниц используется ограниченно, а реентерабельность страниц требуется еще гораздо реже, тем не менее, это является недостатком. Вероятно, этот недостаток может быть разрешен при помощи конечного автомата со стековой памятью, однако рассмотрение этой возможности выходит за рамки настоящей статьи.
2) Для длинных последовательностей страниц, учитывающих историю предыдущих переходов, число дополнительных состояний для одной и той же страницы может достигать больших значений.

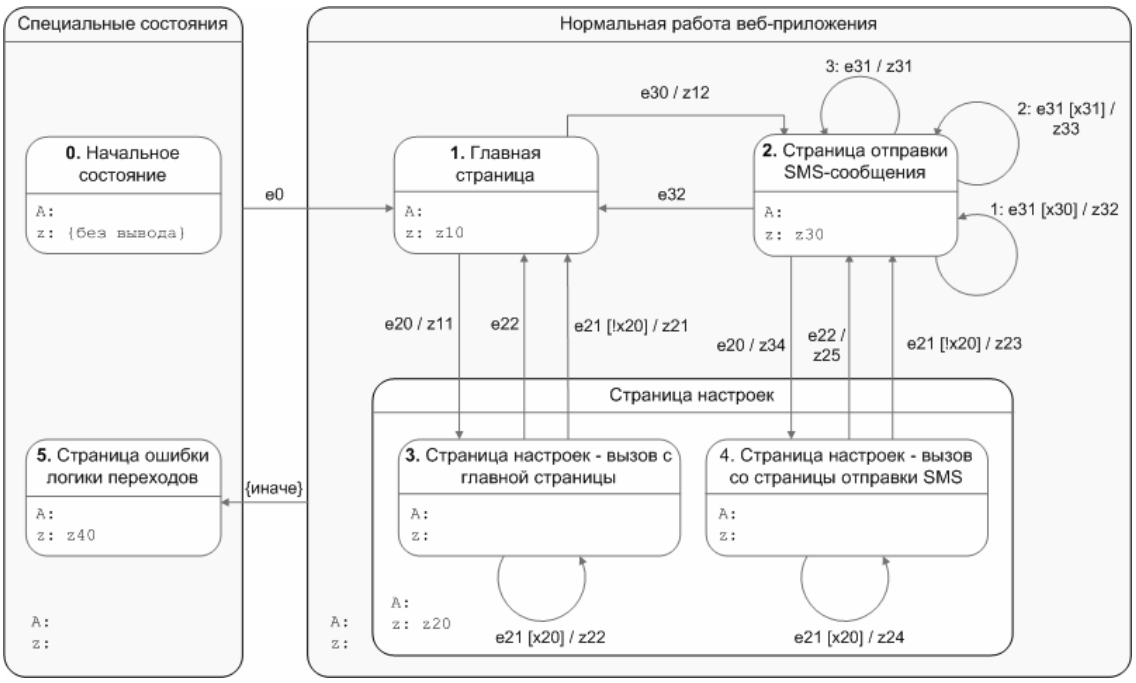


Рис. 6. Диаграмма переходов

Заключение

В настоящей работе был предложен автоматный подход к их разработке, базирующийся на SWITCH-технологии. При этом была затронута проблема контекста. Было дано определение этой проблеме, показано, когда она возникает, каким образом разрешается, а также даны соответствующие комментарии.

Приведенный подход имеет достаточно сильные ограничения, среди которых: отсутствие поддержки реентерабельности веб-страниц, многооконности и асинхронных запросов (AJAX [7]), что требует проведения дальнейшего исследования в этом направлении. Не менее актуально исследование в направлении преобразования веб-приложений, реализованных с применением традиционного подхода, в автоматные веб-приложения, которое также проводится автором.

В заключение следует отметить, что автоматный подход к разработке веб-приложений, и, в частности, предложенный способ решения на базе него проблемы контекста, представляется достаточно удобным.

Литература

1. Шалыто А. SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука. 1998. http://is.ifmo.ru
2. Pring E. Finite state machines in JavaScript, Part 1: Design a widget. http://www-128.ibm.com/developerworks/web/library/wa-finitemach1/
3. Методические рекомендации и указания по разработке, базирующиеся на принципах автоматного программирования, интернет-системы. http://is.ifmo.ru/science/ME-Internet.pdf
4. Гуров В., Мазин М. UniMod. Веб-сайт проекта UniMod. http://unimod.sourceforge.net/
5. Hunter J. Java Servlet Programming. O'Reilly, 1998
6. Веб-сайт проекта PHP. http://www.php.org
7. Технология AJAX. http://ru.wikipedia.org/wiki/AJAX

ИСПОЛЬЗОВАНИЕ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ ДЛЯ РЕШЕНИЯ ЗАДАЧИ ОБ «УМНОМ МУРАВЬЕ»

П.Г. Лобанов
Научный руководитель – д.т.н., профессор А.А. Шалыто

Предложен новый генетический алгоритм для решения задачи об «Умном муравье». Выполнены вычислительные эксперименты, демонстрирующие эффективность этого алгоритма по сравнению с известным.

Введение

В последнее время все шире применяется автоматное программирование, в рамках которого поведение программ описывается с помощью конечных детерминированных автоматов (в дальнейшем автоматов [1]).

Для многих задач автоматы удается строить эвристически, однако существуют задачи, для которых такое построение затруднительно. Известны задачи (итерированная дилемма узников [2], задача о «флибах» [3]), в которых применение генетических алгоритмов позволяет автоматизировать построение конечного автомата. Это один из первых шагов к автоматическому построению программ.

В данной работе рассматривается одна из таких задач – задача об «Умном муравье» [4]. Муравей должен за ограниченное количество ходов съесть всю еду, расположенную определенным образом на поле. Требуется построить конечный автомат, моделирующий поведение муравья. В работе [5] приводится генетический алгоритм для поиска такого автомата. Однако время поиска автомата довольно велико. В данной работе предлагается ряд методов, позволяющих ускорить поиск автомата и уменьшить количество состояний в нем.

1. Задача «умного муравья»

На тороидальном поле размером 32×32 вдоль определенной ломаной линии («тропы») расположено 89 ячеек с едой. Расположение ячеек с едой на «тропе» фиксировано. На рис. 1 приведено пример «тропы», называемый в литературе «тропой Джона Мьюра» (John Muir Trail) [5]. Черным цветом изображены ячейки с едой. Пустые ячейки на «тропе» изображены серым цветом. Серые ячейки невидимы для муравья.

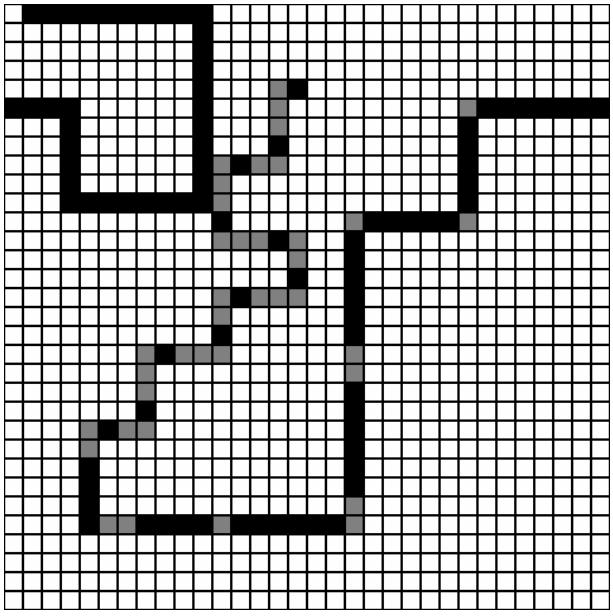


Рис. 1. Тропа Джона Мьюра (John Muir Trail)

Перед первым шагом муравей находится в верхней левой ячейке и смотрит направо. У него есть сенсор, который позволяет ему определить, есть ли еда в ячейке, находящейся непосредственно перед ним. За ход муравей может сделать одно из следующих трех действий – подвинуться вперед, повернуться налево, повернуться направо. Чтобы съесть еду в ячейке, муравью необходимо попасть в нее. После этого ячейка с едой считается пустой и становится для муравья неотличимой от изначально пустых ячеек. Муравей должен съесть всю еду за 200 ходов.

Сенсор муравья определяет значение битовой переменной (ноль – еды в ячейке перед муравьем нет, единица – еда в ячейке перед муравьем есть). Значение этой переменной используется в качестве входного воздействия для конечного автомата, моделирующего поведения муравья. После получения входного воздействия автомат генерирует выходную переменную (действие, которое может осуществить муравей – **вперед, налево** или **направо**) и переходит в новое состояние.

Пример автомата с четырьмя состояниями, моделирующего поведение муравья, построенный в [4] эвристически, приведен на рис 2. Однако муравей с таким поведением задачу не решает, так как за 200 ходов съедает только 42 единицы еды.

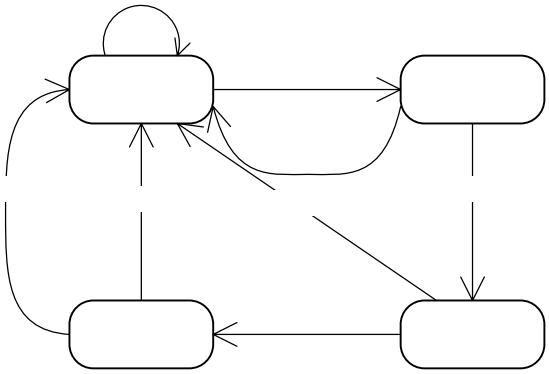


Рис. 2. Автомат с четырьмя состояниями, моделирующий поведение «Умного муравья»

2. Известный генетический алгоритм

Конечный автомат, моделирующий поведение муравья, является решением задачи об «Умном муравье». Чем больше ячеек с едой съедает муравей за 200 ходов, тем лучше решение (в качестве целевой функции [4, 5] используется количество съеденных единиц еды). Автомат задается набором состояний, пронумерованных от нуля. Из каждого состояния выходит две дуги, так как возможно два входных воздействия – ноль и единица. Дуга определяет номер состояния, в которое перейдет автомат при переходе по ней, и значение выходной переменной, генерируемой им при переходе (**вперед, налево** или **направо**).

2.1. Стратегия отбора

Как и в [5], в данной работе используется стратегия отбора особей для следующего поколения – отбор отсечением. При использовании такого отбора родительские решения выбираются из группы лучших решений текущего поколения. Размер этой группы решений (порог отсечения) обычно составляет от 1/100 до 1/3 от размера поколения. Все решения из группы лучших решений имеют одинаковые шансы быть выбранными в качестве родительских решений.

Приведем описание генетического алгоритма, использующего отбор отсечением.

1. Текущее поколение решений заполняется случайными решениями.
2. Из текущего поколения решений отбирается группа лучших решений.

3. Формируется новое поколение решений:
   a. Создается пустое новое поколение решение.
   b. Из группы лучших решений случайным образом выбирается пара решений.
   c. Формируется новое решение с помощью применения оператора скрещивания [4] к двум выбранным решениям.
   d. К новому решению применяются операторы мутации [4].
   e. Новое решение добавляется в новое поколение решений.
   f. Если размер нового поколения решений меньше размера текущего поколения переходим к пункту b.
4. Новое поколение становится текущим.
5. Если число созданных поколений меньше заданного пользователем, переходим к п. 2.

2.2. Оператор скрещивания

В данной работе, как и в работе [5], используется *n*–точечный оператор скрещивания. Приведем описание алгоритма работы этого оператора:

1. В качестве нового решения берется копия первого из выбранных решений.
2. Осуществляется цикл по всем состояниям нового решения. Для каждого состояния:
   a. Выполняется цикл по всем дугам состояния. Для каждой дуги:
      i. Случайным образом определяется, требуется ли изменить номер состояния, в которое переходит автомат по дуге. Если это требуется, то он изменятся на номер состояния из соответствующей дуги второго решения.
      ii. Случайным образом определяется, требуется ли изменить значение выходной переменной, генерируемое при переходе автомата по дуге. Если это требуется, то значение выходной переменной изменяется на значение выходной переменной, полученной из соответствующей дуги второго решения.

Для *n*–точечного оператора скрещивания задается вероятность его применения к каждому элементу автомата.

2.3. Оператор мутации

В данной работе, как и в работе [5] используется *n*–точечный оператор мутации. Приведем описание алгоритма этого оператора:

1. Осуществляется цикл по всем состояниям автомата:
   a. Выполняется цикл по всем дугам переходов в состоянии:
      i. Случайным образом определяется необходимость изменение индекса состояния, в которое автомат переходит по дуге.
      ii. Если требуется изменить индекс состояния, то он изменяется на индекс состояния, выбранный случайным образом.
      iii. Случайным образом определятся необходимость изменения значения выходной переменной, генерируемой автоматом при переходе по дуге.
      iv. Если требуется изменить значение выходной переменной, то оно изменяется на одно из возможных значений, выбранное случайным образом.

Для *n*–точечного оператора мутации задается вероятность его применения к элементу автомата. В отличие от обычного оператора мутации, такой оператор может изменить сразу несколько элементов в автомате.

3. Предложенные модификации алгоритма

3.1. Сортировка состояний в порядке использования

В большинстве решений, которые перебирает генетический алгоритм, автомат в процессе моделирования работы муравья не попадает в часть состояний, количество которых в данном алгоритме задается исходно. На поведение муравья влияют только те

состояния, из которых осуществляется переход по дуге хотя бы на одном из ходов моделирования. Далее такие состояния будем называть **используемыми состояниями**, а все остальные состояния – **неиспользуемыми состояниями**.

Приведем алгоритм сортировки состояний в порядке их использования:

1. Создается пустой словарь пар номеров: [«старый номер состояния» – «новый номер состояния»].
2. Моделируется поведение муравья. Перед каждым переходом по дуге выполняем следующее:
   - если в словаре нет пары, в которой первый элемент равен текущему номеру состояния, то в него добавляется пара [текущий номер состояния – количество пар в словаре].
3. Выполняется цикл по всем состояниям автомата. Для каждого состояния:
   - если в словаре нет пары, в которой первый элемент равен номеру состояния, то в него добавляется пара [номер состояния – количество пар в словаре].
4. Согласно словарю, изменяется порядок состояний и номера состояний в дугах переходов.

Состояния, для которых в словарь добавляются пары в пункте 2 – это и есть **используемые состояния**. Состояния, для которых в словарь добавляются пары в пункте 3 – это **неиспользуемые состояния**.

Используя этот алгоритм, несложно построить автомат, имеющий такое же поведение, как и автомат, найденный с помощью генетического алгоритма, но имеющий меньшее количество состояний. Для этого в автомате, состояния которого отсортированы в порядке использования, достаточно удалить все **неиспользуемые состояния** и заменить переходы в них переходами в состояние с номером ноль.

3.2. Всегда вперед, если впереди еда

После изучения автоматов, которые полностью решают задачу об умном муравье, было замечено, что эти муравьи всегда идут вперед, если перед ними еда. Чтобы достичь такого поведения, был добавлен еще один оператор мутации. Этот оператор изменяет значение выходной переменной на **«вперед»** для всех дуг переходов, помеченных входным воздействием единица (в ячейке перед муравьем еда)

3.3. Уменьшение количества состояний

В настоящей работе изменена целевая функция из [5] таким образом, что наиболее приспособленными оказываются те решения, у которых меньше **используемых состояний**. Для этого в целевую функцию (количество съеденных единиц еды) добавляется соотношение:

$$\frac{1}{1+useful\_state\_count},$$

где *useful_state_count* – число **используемых состояний**. Так как такая добавка всегда меньше единицы, то наиболее приспособленными будут те муравьи, которые съедают больше еды за 200 ходов. Однако, если количество съеденной пищи одинаково, то предпочтение будет отдаваться автоматам с меньшим числом **используемых состояний**, а как показано в разд. 3.1, число состояний в автомате может быть уменьшено до числа **используемых состояний**.

4. Эксперименты

Для проведения экспериментов была написана программа на языке C#, исходные коды которой будут приведены на сайте http://is.ifmo.ru в разделе «Статьи». Программа позволяет задавать размер и максимальное число поколений, устанавливать вероятно-

сти применения для операторов мутации и скрещивания. Также в ней можно изменить порог для стратегии отбора отсечением. При проведении экспериментов при необходимости можно изменять расположение ячеек с едой.

Все эксперименты, о которых идет речь ниже, проводились при размере поколения 10000 и пороге отсечения 2000. Вероятность применения оператора скрещивания к элементу автомата взята равной 0.04, а вероятность применения оператора мутации – 0.02. Число состояний автомата было ограничено двадцатью. Эксперименты проводились на ПК с процессором AMD Athlon 3800+.

В табл. 1 приведены результаты экспериментов, в которых использовалась целевая функция из работы [5]. Число поколений в экспериментах, в которых использовались предложенные в работе операторы мутации, было выбрано равным 100. Число поколений в экспериментах только с *n*–точечным оператором мутации не превышало двухсот. Каждый эксперимент без дополнительных операторов мутации длился примерно 310 с. (время построения двухсот поколений). Продолжительность каждого эксперимента с дополнительными операторами мутации составила около 270 с. (время построения ста поколений).

В столбце «Номер поколения» таблицы приведен номер поколения, в котором был найден лучший автомат в эксперименте. В столбце «Результат» приведено количество ячеек с едой, съеденных лучшим автоматом в эксперименте. Число **используемых состояний** лучшего автомата, полученного в эксперименте, приведено в столбце «Число используемых состояний». В столбце «Время поиска» приводится время в секундах, прошедшее с начала эксперимента, до построения поколения, в котором был найден лучший автомат в эксперименте.

	Номер эксперимента	Номер поколения	Результат	Число используемых состояний	Время поиска
Без дополнительных операторов мутации	1	137	87	15	225
	2	180	88	15	281
	3	73	88	12	137
	4	72	86	12	114
	5	68	89	12	114
С дополнительными операторами мутации	1	47	89	12	154
	2	49	89	11	160
	3	56	89	13	182
	4	51	89	12	145
	5	67	89	13	189

Таблица 1. Результаты экспериментов с использованием стандартной целевой функции

Как следует из приведенных данных, использование передоложенных операторов мутации позволило при всех запусках найти автомат, решающий поставленную задачу. Известный алгоритм с этим не справился за вдвое большее число поколений в четырех экспериментах из пяти. Это, по всей видимости, связано с тем, что в данных экспериментах размер поколения был значительно меньше размера поколения, использующегося в работе [5].

В табл. 2 приведены результаты экспериментов, проводившихся с использованием предложенной в работе целевой функцией. Число поколений в экспериментах проводившихся, с предложенными в работе операторами мутации, было ограничено сотней. Число поколений в экспериментах, в которых использовался только *n*–точечный оператор мутации, не превышало двухсот. Лучшим результатом считался автомат, который съедает максимальное количество ячеек с едой и имеет при этом минимальное

количество **используемых состояний**. Каждый эксперимент без дополнительных операторов мутации длился примерно 320 с (время построения двухсот поколений). Продолжительность каждого эксперимента с дополнительными операторами мутации составила около 290 с (время построения ста поколений).

	Номер эксперимента	Номер поколения	Результат	Число используемых состояний	Время поиска
Без дополнительных операторов мутации	1	143	89	11	237
	2	144	89	9	263
	3	138	88	10	259
	4	99	89	13	152
	5	158	89	10	252
С дополнительными операторами мутации	1	61	89	10	161
	2	85	89	10	263
	3	70	89	10	227
	4	63	89	10	176
	5	96	89	9	269

Таблица 2. Результаты экспериментов с использованием предложенной целевой функции

Как следует из приведенных в табл. 2 данных, изменение целевой функции позволило получить автоматы с меньшим числом **используемых состояний** по сравнению с предыдущим случаем. Необходимо отметить, что алгоритм, использующий дополнительные операторы мутации, работает более стабильно и с измененной целевой функцией.

Сравнив данные, приведенные в табл. 1 и 2, отметим, что изменение целевой функции не только не ухудшило сходимость алгоритма, но даже улучшило ее.

В табл. 3 приведен автомат, который решает задачу об «Умном муравье» для «тропы Джона Мьюра» и имеет минимальной число **используемых состояний** из построенных автоматов.

	Перед муравьем нет еды		В ячейке перед муравьем еда	
Номер состояния	Номер следующего состояния	Выходная переменная	Номер следующего состояния	Выходная переменная
0	6	**направо**	1	**вперед**
1	2	**вперед**	2	**вперед**
2	8	**вперед**	3	**вперед**
3	2	**вперед**	4	**вперед**
4	6	**направо**	5	**вперед**
5	4	**налево**	3	**вперед**
6	7	**направо**	1	**вперед**
7	2	**налево**	1	**вперед**
8	0	**налево**	4	**вперед**

Таблица 3. Таблица переходов и выходов для лучшего из найденных автоматов

Отметим, что число состояний в автомате, приведенном в таблице 3 (девять состояний) меньше числа состояний в автоматах, полученных в работах [5] и [6] (одиннадцать состояний), которые, кроме того, содержат опечатки.

Заключение

В работе предложены два новых оператора мутации и целевая функция для генетического алгоритма из работы [6]. Применение предложенных методов позволило сократить время поиска автомата, решающего задачу об «Умном муравье» для «тропы Джона Мьюра».

Написана программа, позволяющая производить эксперименты как с предложенными методами, так и без них. Приведенные экспериментальные данные показывают эффективность предложенных методов по сравнению с известным.

Предложенная в работе целевая функция позволила уменьшить количество состояний у автоматов, получаемых с помощью генетического алгоритма. Это также подтверждается экспериментальными данными.

В заключение отметим, что задача поиска муравья с наименьшим количеством состояний не ставилась. При этом отметим, что известен муравей, решающий задачу, который имеет восемь состояний [7].

Литература

1. Шалыто А.А. Технология автоматного программирования. / Труды первой Всероссийской конференции «Методы и средства обработки информации». М.: МГУ. 2003. http://is.ifmo.ru/works/tech_aut_prog
2. Mitchell M. An Introduction to Genetic Algorithms. MIT Press. Cambridge. MA, 1996.
3. Лобанов П.Г., Шалыто А.А. Использование генетических алгоритмов для автоматического построения конечных автоматов в задаче о «флибах». / Материалы 1-й Российской мультиконференции по проблемам управления. Сборник докладов 4-й Всероссийской научной конференции «Управление и информационные технологии» (УИТ-2006). СПбГЭТУ «ЛЭТИ», 2006. С. 144–149. http://is.ifmo.ru/works/_flib.pdf
4. Koza J.R. Genetic programming. On the Programming of Computers by Means of Natural Selection. The MIT Press, 1998.
5. Jefferson D., Collins R., Cooper C., Dyer M., Flowers M., Korf R., Taylor C., Wang A. The Genesys System: Evolution as a Theme in Artificial Life. // In Langton et al. (ALife II). 1992. www.cs.ucla.edu/~dyer/Papers/AlifeTracker/Alife91Jefferson.html
6. Angeline P.J., Pollack J. Evolutionary Module Acquisition. // Proceedings of the Second Annual Conference on Evolutionary Programming. 2003. http://www.demo.cs.brandeis.edu/papers/ep93.pdf
7. Chambers L. Practical handbook of genetic algorithms, Boca Raton, CRC Press, 1995.

ОБ ОСОБЕННОСТЯХ РАЗРАБОТКИ КОМПЛЕКСА ВИРТУАЛЬНЫХ ЛАБОРАТОРНЫХ РАБОТ ПО ЭКОЛОГИЧЕСКОЙ ТЕМАТИКЕ

Д.П. Баранова
Научный руководитель – к.т.н., доцент Н.И. Керро

В статье обосновывается актуальность проблемы разработки комплекса виртуальных лабораторных работ по экологической тематике, предложен общий алгоритм такой виртуальной лабораторной работы, который успешно реализован автором в рамках компьютерной программы, затем апробирован в условиях реального учебного процесса.

Введение

В настоящее время в связи с тем, что в учебный процесс в вузе активно внедряются информационные технологии, меняются как стиль, так и методы обучения. Применительно к условиям реального учебного процесса эти методы постоянно совершенствуются, причем с учетом специфических особенностей конкретной учебной дисциплины. В конечном счете эти усовершенствования сводятся к тому, что появляются новые эффективные методики обучения, ориентированные, как правило, на более активное использование информационных технологий в процессе обучения. В полной мере это относится и к дисциплинам экологического профиля.

Известно, что информатизация общества является основной отличительной чертой современного этапа его развития. В этих условиях многие традиционные методы преподавания стали отходить на задний план, а на смену им пришли более прогрессивные методики, основанные на имитации диалога обучающегося и преподавателя, который, как правило, реализуется при непосредственном использовании ими компьютера. Следует отметить, что диалог преподавателя с учеником можно организовать и другим способом, например, в ходе проведения деловой игры, организация которой требует от преподавателя не только досконального знания предмета, но и хороших коммуникативных навыков, а также умения грамотно и четко организовать саму процедуру деловой игры. В этом плане участие преподавателя в диалоге со студентами, организованном с использованием компьютера, существенно упрощает задачу преподавателя.

Что касается информатизации процесса обучения, то с позиций современной дидактики он чаще всего трактуется как использование вычислительной техники и связанных с ней информационных технологий в процессе обучения, а также как использование этой техники в качестве средства управления познавательной деятельностью учащихся.

Постановка задачи

Применение информационных и коммуникационных технологий в высшей школе традиционно развивается по двум основным направлениям. Первое направление предполагает использование возможностей этих технологий для непосредственного включения в процесс образования лиц, для которых в силу тех или иных причин возможна и доступна только дистанционная форма обучения. Второе направление замыкается на использование информационных технологий для изменения содержательной составляющей процесса обучения путем обновления и совершенствования его методической базы и создания и внедрения на этой основе новых более прогрессивных методик.

Поскольку первое направление развивается сейчас достаточно успешно, а дистанционные курсы лекций по различным дисциплинам созданы в большинстве учебных заведений, то наибольший интерес для исследователя на настоящий момент представ-

ляет второе направление. Возможен также некий синтез этих направлений в рамках одного проекта, например, нацеленного на создание современных виртуальных учебно-методических комплексов (ВУМК). Считается, что такой комплекс может использоваться как при дистанционной форме обучения, так и традиционной очной. В состав такого комплекса входят теоретические материалы, лабораторные работы и практикумы, а также средства для контроля знаний обучаемых, например, в форме тестовых вопросов. Объединенный общей проблематикой, например, экологической, этот комплекс может достаточно успешно использоваться в реальном учебном процессе, поскольку его применение позволит не только более эффективно организовать работу студентов, но и значительно сократить нагрузку на преподавателя по проверке их знаний [1].

Важным элементом виртуального учебного методического комплекса являются виртуальные лабораторные работы (ВЛР) или практикум. Следует сказать, что на настоящий момент применительно к дисциплинам экологического профиля ВЛР практически отсутствуют. Этим обстоятельством и объясняется актуальность работы автора, целью которой являлась разработка стандартного алгоритма ВЛР, реализация его в рамках существующих расчетных заданий по тематике «Охрана и рациональное использование водных ресурсов» с последующей апробацией созданной ВЛР условиях реального учебного процесса.

Разработке настоящей ВЛР предшествовал процесс сбора и анализа информации об аналогичных проработках, существующих на настоящий момент в рамках других дисциплин, а также поиск материалов по экологической тематике, которые могли бы быть заложены в основу расчетной части ВЛР. В ходе анализа и сбора информации выяснилось, что виртуальные лабораторные работы созданы и используются в учебном процессе в Томском государственном университете, в Санкт-Петербургском государственном университете низкотемпературных и пищевых технологий и других учебных заведениях. Однако их тематика не имеет отношения к экологическому направлению. Анализ этих проработок позволил сделать вывод о том, что для создания таких педагогических программных продуктов, как ВЛР, как правило, используют следующие технические методы и средства: прямое программирование, пакеты профессиональных программ, специализированные авторские инструментальные среды.

В данном случае автором был выбран путь прямого программирования, основным достоинством которого являются широкие возможности по реализации стилей педагогических программных средств. Речь идет об организации интерфейса и структуры представления учебного материала, возможностей использования цветовой палитры и подачи материала. Однако прямое программирование имеет также и недостатки. К ним, в первую очередь, относятся: сложность модификации и сопровождения готового продукта, трудность стыковки модулей программы, созданных на разных языках, большая трудоемкость самого процесса программирования [2].

Анализ известных на настоящий момент методических проработок по содержательной части лабораторных работ по экологической тематике показал, что только незначительное количество из них содержат как расчетную, так и графическую часть и поэтому могут достаточно эффективно использоваться в практике обучения студентов технических вузов дисциплинам экологического цикла. Наибольший интерес в этом плане представляют лабораторные работы методического пособия [3]. Одна из лабораторных работ этого пособия была взята автором за основу при разработке ВЛР.

Поскольку основная цель работы автора состояла в том, чтобы разработать алгоритм лабораторной работы в рамках единого комплекса виртуальных лабораторных работ для обучения студентов дисциплине «Прикладная экология», а также реализовать его на практике, то для достижения поставленной цели были решены следующие задачи:

- разработана общая структура виртуальной лабораторной работы;
- разработан интерфейс виртуальной лабораторной работы;

- написан код программы для организации взаимодействия и работы пользователя с графическим материалом;
- разработана база данных для хранения результатов работы и исходных данных;
- осуществлено внедрение разработанной виртуальной лабораторной работы в учебный процесс.

Особенности разработанной программы

Общая структура ВЛР представлена на рис. 1.

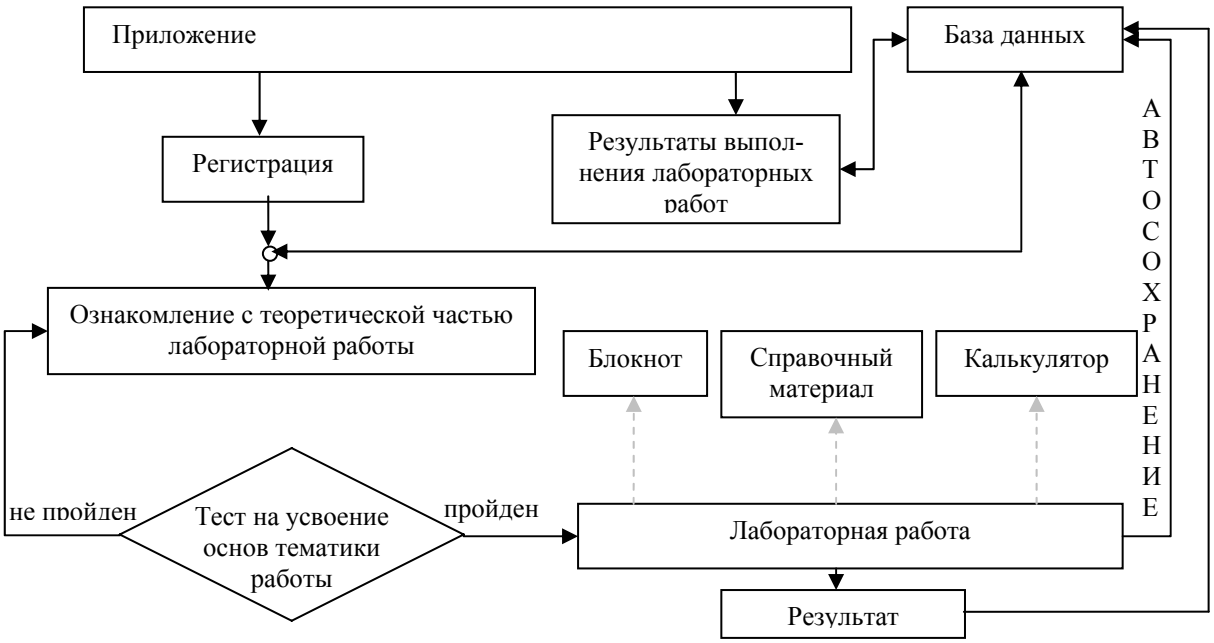


Рис. 1. Общая структура виртуальной работы

Виртуальная лабораторная работа реализована на базе Windows-приложения с использованием базы данных MySQL. При выполнении этой работы учащимся необходимо освоить методику расчета основных параметров процесса очистки сточной воды. Для этого им необходимо последовательно в диалоговом режиме решить следующие задачи:

- изучить теоретическую часть лабораторной работы о процессе очистки воды озонированием;
- пройти тест на знание теоретической части, успешное прохождение которого является допуском к практической части лабораторной работы;
- выполнить последовательно несколько расчетных заданий.

Ниже перечислены основные особенности ВЛР и кратко изложен механизм использования ее студентом.

- В качестве допуска к лабораторной работе используется тест. Время на выполнения теста ограничено. Вопросы и варианты ответов в тесте представляются для студента случайным образом.
- Лабораторная работа выполняется в четыре этапа. После завершения очередного этапа компьютерная программа проверяет результаты вычислений студента, если они попадают в допустимый промежуток результатов вычисленных программой, тогда учащийся переходит к следующему этапу. Данная проверка осуществляется во избежание использования некорректных данных, полученных вследствие ошибок студента, на последующих этапах работы.

- После выполнения студентом задания каждого этапа происходит автоматическое сохранение полученных им результатов, что позволяет студенту, не успевшему по какой-либо причине вовремя завершить работу, закончить ее тогда, когда у него будет такая возможность.
- В помощь студенту при проведении им вычислений в программе реализованы встроенный блокнот, калькулятор (с дополнительными возможностями) и приведен пример выполнения расчетов в лабораторной работе. Все расчеты, произведенные студентом на калькуляторе, автоматически переносятся в лабораторную работу.
- Алгоритм программы позволяет преподавателю контролировать процесс выполнения лабораторной работы со своего рабочего места, оснащенного компьютером в ходе просмотра полного отчета о выполненной студентом работе, в котором указываются все допущенные студентом ошибки. Результаты выполнения лабораторной работы доступны только преподавателю, что исключает возможность внесения студентом корректив в отчет. Ограничение прав доступа к результатам реализованы средствами СУБД MySQL.
- В списке результатов реализована гибкая система фильтрации, позволяющая осуществить выборку из базы данных отчетов студентов как по фамилии, так и по дате, номеру группы, результатам лабораторной работы, а также номеру варианта.
- Программа предусматривает хранение исходных данных и результатов выполненных студентами работ в базе данных MySql.
- Преподавателю предоставлена возможность просмотра результатов выполненных студентами работ с любого компьютера, подключенного к локальной сети и имеющего доступ к MySql;
- Программа реагирует на введенные пользователем данные и в зависимости от ситуации организует диалог с пользователем.

Внедрение виртуальной лабораторной работы в учебный процесс и результаты ее апробации

Результаты апробации и внедрения виртуальной лабораторной работы на занятиях в группе студентов естественнонаучного факультета СПбГУ ИТМО по теме «Методы очистки сточных вод» показали, что студенты за отведенное на выполнение работы время успешно справились с изучением теоретической части, тестированием и выполнили все расчетные задания, при этом количество ошибок, допущенных студентами в расчетах, не превысило допустимого. Программа обработала данные по результатам выполнения работы, которые затем были представлены в виде сводного перечня индивидуальных отчетов студентов, содержащего сведения о допущенных ими ошибках.

В ходе апробации ВЛР было проведено дополнительное тестирование студентов на предмет того, насколько доступен теоретический материал, изложенный в вводной части виртуальной лабораторной работы, и какие сложности возникали у них при выполнении этой работы.

После обработки результатов тестирования были получены следующие результаты:

- интерфейс программы не вызвал затруднений у 70 %; остальные 30 % испытывали их поначалу, но потом адаптировались;
- теоретический материал лабораторной работы не вызвал каких-либо дополнительных вопросов у студентов;
- основные трудности, которые испытывали студенты при выполнении лабораторной работы, связаны с вычислениями;

- с тестированием после изучения теории вопроса очистки сточных вод справилось большинство студентов;
- все учащиеся отметили, что им было интересно работать с данной лабораторной работой;
- 92 % учащихся считают необходимым использование подобных виртуальных лабораторных работ в обучении, так как они вносят разнообразие в учебный процесс, облегчают труд преподавателей и способствуют совершенствованию практических навыков у студентов.

Таким образом, апробация разработанной автором ВЛР в условиях реального учебного процесса подтвердила предположение о том, что ее общий алгоритм может быть достаточно успешно реализован применительно к другим виртуальным лабораторным работам, внедрение которых в учебный процесс несомненно позволит получить нужный эффект вследствие улучшения методики обучения дисциплинам экологического цикла.

Заключение

В настоящее время в связи с активным внедрением информационных технологий в процесс обучения большое внимание уделяется созданию ВУМК, в состав которых, как правило, помимо теоретической части, тестовых и контрольных вопросов входят также практические задания, например в форме ВЛР. Однако в составе комплексов по экологическим дисциплинам ВЛР практически отсутствуют. В рамках данной работы автор постарался восполнить этот пробел и создал виртуальную лабораторную работу «Исследование процесса очистки воды при окислении загрязняющих веществ», которая, судя по ее апробации в реальном учебном процессе, с успехом может быть использована в качестве одного из основных элементов обучающего комплекса по экологии.

К несомненным достоинствам разработанной ВЛР относятся:

- сокращение нагрузки на преподавателя по проверке знаний студентов и результатов выполнения работы;
- внесение разнообразия в учебный процесс;
- возможность ее использования системе дистанционного обучения.

По общему алгоритму этой работы могут быть реализованные и другие виртуальные лабораторные работы аналогичной тематики. Это говорит о том, что данное направление исследования является актуальным, а результаты проработок в настоящее время могут быть востребованы со стороны учебных заведений самого широкого профиля.

Литература

1. Захарова И.Г. Информационные технологии в образовании. М.: ACADEMIA, 2003. 189 с.
2. Горлушкина Н.Н. Педагогические программные средства: Учебное пособие / Под. ред. проф. М.И. Потеева. СПб.: СПбГИТМО (ТУ), 2002. 152 с.
3. Латухов С.В., Решняк В.И., Поляков А.В., Моисеев А.Ю., Праслов Л.Д., Шуринова О.В. Экологическая безопасность водных объектов. Учебно-методическое пособие СПб: ГМА им. С.О. Макарова, 2005. 85 с.
4. Веллинг Л., Томсон Л. MySQL. Учебное пособие.: Пер. с англ. М.: Издательский дом «Вильямс», 2005. 304 с.

ОСНОВНЫЕ МАТЕМАТИЧЕСКИЕ МОДЕЛИ И МЕТОДЫ РАСЧЕТА ФУНКЦИОНАЛЬНОЙ НАДЕЖНОСТИ КОРПОРАТИВНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

А.В. Мейко (Казанский государственный технический университет им. А.Н. Туполева)
Научный руководитель – д.т.н., профессор В.С. Моисеев (Казанский государственный технический университет им. А.Н. Туполева)

Предлагаются математические модели и методы расчета функциональной надежности для применения при проектировании современных корпоративных информационных систем. Обсуждаются оригинальные методы оценки чувствительности показателей надежности таких систем.

Введение

В настоящее время автоматизированные системы управления (АСУ) вступили в новую стадию своего развития, которая определяется широким внедрением в практику сложных территориально-распределенных корпоративных информационных систем (КИС). Появление, развитие и распространение сложных систем, компонентами которых являются технические средства, программное обеспечение, человек, требуют новых подходов к оценке их надежности и безопасности. Одним из важнейших вопросов теории и практики надежности является математическое моделирование функционирования систем, разработка методов и алгоритмов расчета, анализа и прогнозирования их надежности. Сложность решения задачи расчета надежности систем управления обусловлена отсутствием универсальных методов, учитывающих неполноту и неоднородность исходной информации о надежности элементов систем.

Анализ существующих методов и моделей расчета надежности сложных информационных систем выявил основные недостатки этих методов, а именно отсутствие методик расчета такого важного их свойства, как функциональная надежность, отсутствие учета структур и особенностей используемых технических и программных средств системы, а также сложный математический аппарат для расчета показателей надежности. Последнее не позволяет использовать его в составе инженерных методик расчета надежности корпоративных информационных систем. В ГОСТе по надежности АСУ [1] также приведен лишь список показателей надежности, но сами методы расчета надежности КИС отсутствуют. Поэтому разработка достаточно новых простых математических методов расчета и анализа надежности функционирования компонент КИС является актуальной [2].

В данной работе делается попытка ликвидации этого недостатка: разработаны инженерные математические модели оценки надежности функциональных компонентов КИС, а также приведены методы оценки чувствительности показателей надежности таких систем.

Постановка задачи

Корпоративные информационные системы – это интегрированные системы управления территориально распределенной корпорацией, основанные на углубленном анализе данных, широком использовании систем информационной поддержки принятия решений, электронных документообороте и делопроизводстве. КИС призваны объединять стратегию управления (бизнес-стратегию) предприятием и передовые информационные технологии [3].

Основным понятием в работе является понятие «функциональной надежности КИС», под которым в данной работе понимается свойство системы, обеспечивающее

требуемый уровень работоспособности рассматриваемой КИС при выполнении возложенных на нее функций.

В работе [4] показано, что любая информационная система S может быть описана отношением вида: $S \subseteq T \times P \times D \times U$, где T – множество технических средств системы; P – множество программ системы; D – множество данных, используемых при ее функционировании; U – множество пользователей системы.

Корпоративные системы охватывают, как правило, всю финансово-хозяйственную и производственную деятельность предприятия, в том числе имеющего филиалы и дочерние фирмы, входящего в холдинговые компании и концерны. В частности, типовыми функциями КИС являются автоматизация документооборота, моделирование бизнес-процессов, устранение внутрифирменных барьеров, обеспечение доступа в международные информационные сети. Пусть F – множество функций, реализованных в составе КИС (планирование, учет, финансы и т.п.). Тогда реализацию некоторой функции $f \in F$ можно рассматривать как некоторое подотношение $S_f \subset S$ такое, что $S_f \subset T \times P \times D \times U$, $f \in F$. При этом предполагается, что $\bigcup_{f \in F} S_f = S$.

Требуется разработать достаточно простые математические модели для оценки надежности функционирования этих компонент КИС.

Отметим, что в существующих работах по надежности информационных (автоматизированных) систем предлагаются методы отдельного расчета показателей надежности отмеченных выше компонентов, хотя в процессе их функционирования все эти компоненты тесно связаны. Для разработки методик расчета показателей функциональной надежности КИС введем в рассмотрение два вида элементов КИС: аппаратно-программные и функциональные элементы КИС.

Под аппаратно-программными элементами (АПЭ) КИС будем понимать технические средства КИС и системное программное обеспечение, необходимое для их функционирования. Примерами таких элементов являются телекоммуникационное оборудование, серверы, персональные компьютеры и т.п.

Функциональные элементы (ФЭ) КИС должны включать в себя алгоритмы соответствующих деловых (бизнес-) процессов, диалоговые комплексы программ, необходимые для их работы данные и эксплуатирующих их пользователей.

Множество ФЭ КИС должно полностью отражать множество функций F рассматриваемой системы.

Динамические модели расчета надежности аппаратно-программных элементов корпоративных информационных систем

Рассмотрим предварительные схемы расчета надежности АПЭ КИС.

Введем следующие состояния АПЭ КИС: E_0 – АПЭ работоспособен; E_1 – отказ и восстановление программной части АПЭ; E_2 – отказ и восстановление технической части АПЭ; E_3 – одновременный отказ и восстановление обеих частей АПЭ. Взаимосвязь рассматриваемых состояний АПЭ представлена на рис. 1, где λ_i и μ_i – соответственно интенсивности отказа и восстановления i-й части АПЭ, $i = \overline{(1,3)}$.

Обозначим через $p_i(t)$ – вероятность того, что процесс функционирования АПЭ в момент времени t находится в состоянии E_i, $i = \overline{(0,3)}$. Тогда уравнения, описывающие динамику изменения этих вероятностей, записываются как [5]:

$$\frac{\mathrm{d}p_0}{\mathrm{d}t} = -(\lambda_1 + \lambda_2 + \lambda_3)p_0 + \mu_1 p_1 + \mu_2 p_2 + \mu_3 p_3; \qquad (1)$$

$$\frac{dp_1}{dt} = \lambda_1 p_0 - \mu_1 p_1;$$

$$\frac{dp_2}{dt} = \lambda_2 p_0 - \mu_2 p_2;\ \frac{dp_3}{dt} = \lambda_3 p_0 - \mu_3 p_3;$$

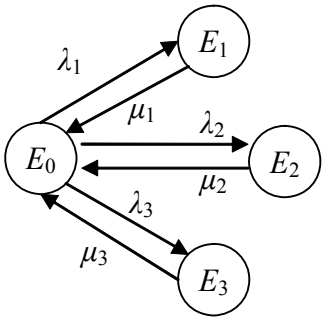


Рис. 1. Взаимосвязь состояний АПЭ

Кроме того, необходимо выполнение условия:

$$p_0(t) + p_1(t) + p_2(t) + p_3(t) = 1, \qquad (2)$$

так как события, состоящие в том, что в момент времени t система находится в состояниях E_0, E_1, E_2, E_3, несовместны и образуют полную группу.

Начальные условия для дифференциальных уравнений (1) имеют вид:

$$p_0(0)=1,\ p_1(0)=0,\ p_2(0)=0,\ p_3(0)=0. \qquad (3)$$

При использовании математической модели надежности АПЭ КИС вида (1)–(3) изменение вероятности его безотказной работы определяется функцией $p_0(t)$. Вероятность отказа АПЭ к моменту времени t определяется как $q(t)=1-p_0(t)$. Вероятность безотказной работы АПЭ определяется функцией вида $P_{АПЭ}(t)= p_0(t)$.

Далее нас в большей степени будут интересовать вероятности $p_0(t)$, $p_1(t)$, $p_2(t)$, так как событие E_3 происходит крайне редко и практически не влияет на надежность АПЭ.

Модели оценки надежности функциональных элементов корпоративных информационных систем

Рассмотрим модель оценки надежности функционального элемента (ФЭ) КИС, реализованного в виде диалогового комплекса программ (КП). Граф связи состояний процесса его функционирования приведен на рис. 2.

Построим модель оценки надежности работы пользователя КИС. В процессе его работы выделим следующие состояния: E_0 – запуск КП; E_1 – пользователь осуществляет ввод данных; E_2 – пользователь проводит контроль вводимых данных; E_3 – пользователь обнаружил ошибку и проводит ее исправление; E_4 – пользователь не обнаружил пропущенную ошибку; E_5 – выполнение функций, реализованных в составе КП; E_6 – отказ и восстановление КП; E_7 – контроль результатов работы (выходных данных) КП; E_8 – завершение работы КП.

Математическая модель оценки надежности функционирования диалогового КП будет выглядеть так:

$$\frac{dp_0}{dt} = -\lambda_{01} p_0;\quad \frac{dp_1}{dt} = -(\lambda_{12} + \lambda_{15}) p_1 + \lambda_{01} p_0 + \lambda_{31} p_3 + \lambda_{41} p_4;$$

$$\frac{dp_2}{dt} = -(\lambda_{23} + \lambda_{24})p_2 + \lambda_{12}p_1 + \lambda_{72}p_7;\ \frac{dp_3}{dt} = -\lambda_{31}p_3 + \lambda_{23}p_2;$$

$$\frac{dp_4}{dt} = -\lambda_{41}p_4 + \lambda_{24}p_2;\ \frac{dp_5}{dt} = -(\lambda_{56} + \lambda_{57})p_5 + \lambda_{15}p_1 + \lambda_{65}p_6;$$

$$\frac{dp_6}{dt} = -\lambda_{65}p_6 + \lambda_{56}p_5;\ \frac{dp_7}{dt} = -(\lambda_{72} + \lambda_{78})p_7 + \lambda_{57}p_5;$$

$$\frac{dp_8}{dt} = \lambda_{78}p_7; \qquad \sum_{i=0}^{8} p_i(t) = 1.$$

Начальные условия имеют вид: $p_0(0)=1,\ p_i(0)=0,\ i=(\overline{1,8})$.

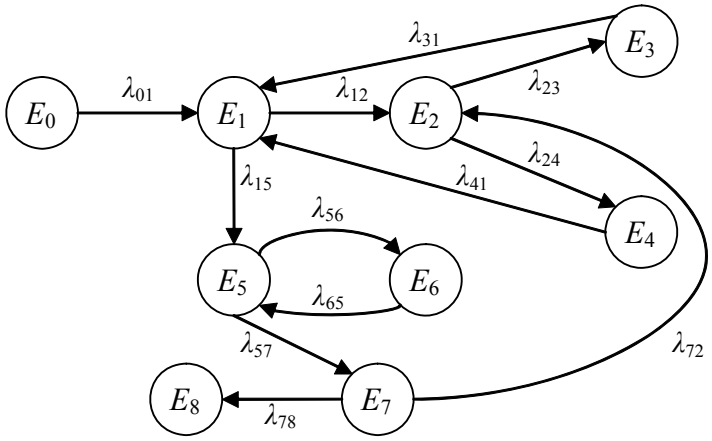


Рис. 2. Граф связи состояний процесса функционирования ФЭ

Пусть τ_f – заданное значение затрат времени на решение задачи с помощью КП. Тогда надежность реализации функции $f \in F$ системы с помощью рассматриваемого КП определим как вероятность правильности полученных к моменту времени τ_f результатов, которая вычисляется как $P_{КП}(\tau_f) = p_8(\tau_f)$.

Заметим, что данные методы применимы для отказов, подчиняющихся экспоненциальному закону распределения. Поэтому данные по отказам и восстановлениям подверглись статистической обработке, в результате гипотеза о экспоненциальном законе распределения подтвердилась в 83 % случаев [6].

Модели оценки надежности баз данных корпоративных информационных систем

База данных (БД) является важнейшим компонентом КИС, от надежного функционирования которого зависит эффективное выполнение практически всех функций системы [7].

Наиболее простая модель для оценки надежности работы корпоративной базы данных (БД) может быть представлена как восстанавливаемая система с ограниченным объемом запасных частей (в данном случае – БД-реплик).

Пусть рассматриваемая система корпоративных БД с «теплым» резервированием подвергается пуассоновскому потоку отказов с интенсивностью λ. В системе имеется N локальных копий, т.е. реплик БД, а также файлов вносимых изменений (локальных журналов транзакций). В самом простом случае восстановление реплик осуществляется с помощью эталонной версии БД, а их дальнейшая актуализация производится на основе соответствующего локального журнала изменений.

Будем считать, что затраты времени на восстановление некоторой реплики БД с помощью эталонной версии гораздо выше рассматриваемого интервала функциониро-

вания, поскольку требуется осуществить пересылку достаточно больших объемов данных (полной копии эталонной БД) и произвести их актуализацию. В этом случае надежность БД определяется вероятностью того, что имеющегося числа ее реплик хватит для обеспечения устойчивой работы системы на интервале времени (0,*t*), поскольку при отказе какой-то реплики БД запросы будут просто переадресованы к одной из оставшихся в рабочем состоянии реплик.

Надежность корпоративной БД целесообразно оценивать как вероятность того, что на интервале (0, *t*) число отказов реплик БД будет не больше (*N*-1), которая вычисляется по формуле вида

$$P_{БД}(t) = \sum_{k=0}^{N-1} P_k(t) = e^{-\lambda t} \sum_{k=0}^{N-1} \frac{(\lambda t)^k}{k!}. \qquad (4)$$

Эта величина может быть принята в качестве оценки вероятности безотказной работы системы с «теплым» резервированием, состоящей из *N* реплицированных корпоративных БД, к моменту времени *t*.

Обозначим через $P_{БД}^{треб}$ требуемый уровень надежности БД в составе КИС. Тогда требуемое число $N_{треб}$ реплик БД, необходимых для ее работы с уровнем надежности $P_{БД}^{треб}$ за время τ, определяется с учетом выражения (4) по следующей формуле:

$$N_{треб} = \arg \min_{N=1,2,\ldots} \{\sum_{k=0}^{N-1} \frac{(\lambda\tau)^k}{k!} \geq P_{БД}^{треб} e^{\lambda\tau}\}.$$

Рассмотрим более сложную, но достаточно реальную ситуацию, когда в территориально-распределенных узлах обработки данных КИС осуществляется локальное резервное копирование состояний БД-реплик через некоторые априорно заданные интервалы времени. Предполагается, что последовательность снятия резервных копий (т.е. контрольных точек) в целом для всей системы синхронизирована. В этом случае каждая из реплик БД восстанавливается со своей последней локальной РК, и далее производится ее актуализация в соответствии с локальным журналом изменений.

Дальнейшее построение моделей, предназначенных для оценки характеристик надежности рассматриваемой системы, будем осуществлять на основе формализма конечных цепей Маркова [8].

Введем в рассмотрение следующие состояния системы: E_0 – в системе функционируют и доступны все реплики корпоративной БД; E_1 – одна из реплик находится в состоянии отказа (неработоспособна), производится ее восстановление с использованием соответствующей локальной РК; E_2 – две реплики в состоянии отказа; и далее E_N – все имеющиеся реплики корпоративной БД находятся в состоянии отказа, производится их восстановление. Именно в этом последнем случае можно говорить об отказе всей корпоративной БД, поскольку ни одна из реплик не будет работоспособной и доступной для работы пользователей.

По-прежнему будем считать, что система подвергается пуассоновскому потоку отказов с интенсивностью λ, а восстановление каждой из реплик осуществляется с интенсивностью μ. Граф связи состояний системы в виде марковской цепи представлен на рис. 3.

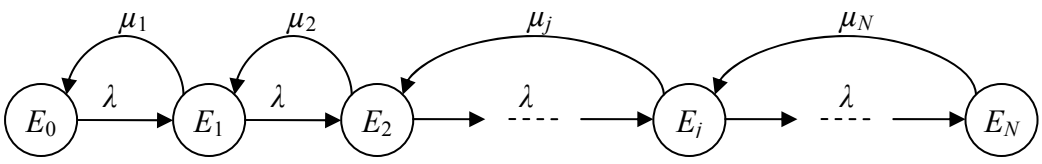


Рис. 3. Граф связи состояний системы в виде марковской цепи

Подобная схема переходов состояний сложной системы получила известность как марковская модель «гибели-размножения» [5]. Введем также в рассмотрение приведенную интенсивность потока отказов, которая определяется как $\rho = \frac{\lambda}{\mu}$. Тогда базовая модель расчета надежности корпоративной БД для графа, представленного на рис. 3, определяется через так называемые финальные вероятности состояний системы на основании формул Эрланга [4] (с учетом того, что $\mu_1 = \mu$; $\mu_2 = 2\mu$; $\mu_3 = 3\mu$ … $\mu_N = N\mu$) в следующем виде:

$$p_0 = (1 + \rho + \frac{\rho^2}{2!} + \frac{\rho^3}{3!} + ... + \frac{\rho^N}{N!})^{-1};$$

$$p_1 = \rho p_0;\ p_2 = \frac{\rho^2}{2!} p_0;\ p_3 = \frac{\rho^3}{3!} p_0;\ \ldots;\ p_N = \frac{\rho^N}{N!} p_0;\ \sum_{i=0}^{N} p_i = 1. \quad (5)$$

Отсюда с использованием системы уравнений (5), которая описывает финальные вероятности всех возможных состояний системы в рамках данной модели, искомая вероятность безотказной работы рассмотренной схемы функционирования корпоративной БД вычисляется как

$$Q_{БД} = 1 - p_N = 1 - \frac{\rho^N}{N!} p_0.$$

Таким образом, продемонстрирован подход, позволяющий с помощью несложных выкладок на основе экспертных оценок интенсивностей потоков отказов и восстановления [6] получить достаточно адекватные оценки таких важных показателей надежности КИС, как вероятность безотказной работы, вероятность работы с производительностью не ниже заданной и т.д.

Расчет надежности вспомогательных функций корпоративной информационной системы

Приведенные модели и методы рассмотрим на примере расчета функциональной надежности подсистемы учета и анализа долгосрочных вложений, являющейся основной компонентой существующих КИС. Для данной подсистемы рассчитывались такие показатели надежности, как коэффициент готовности к выполнению *i*-й функции и вероятность безотказного выполнения *i*-й функции.

Рассмотрим методику расчета на примере функций автоматизированной загрузки данных о долгосрочных вложениях организации, ее дочерних и зависимых обществ. Схема подключения аппаратных средств для данной функции представлена на рис. 4.

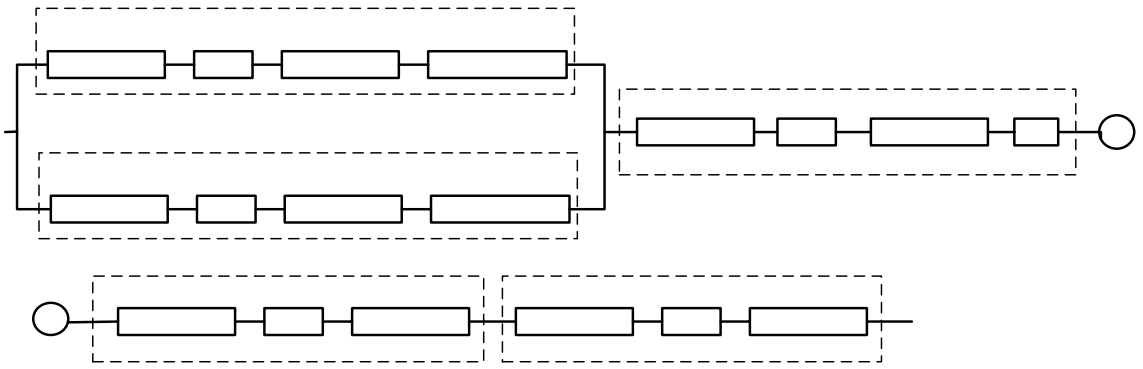


Рис. 4. Схема подключения аппаратных средств

Введем следующие состояния подсистемы, реализующей эту функцию, с точки зрения аппаратно-программных средств: E_0 – подсистема работоспособна; E_1 – отказ автоматизированных рабочих мест; E_2 – отказ сервера БД Администрации; E_3 – отказ сервера электронной почты (внутр.); E_4 – отказ сервера электронной почты (внеш.). Взаимосвязь выделенных состояний описывается ориентированным графом, представленным на рис. 5, где $\lambda_{iАП}$ и $\mu_{iАП}$ – соответственно интенсивности отказа и восстановления i-й части АПЭ, $i = (\overline{1,4})$.

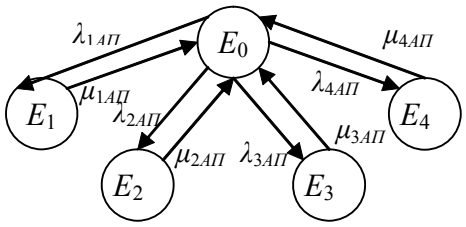


Рис. 5. Взаимосвязь состояний подсистемы учета и анализа долгосрочных вложений

Система дифференциальных уравнений, описывающих работу данной подсистемы, выглядит следующим образом [5]:

$$\frac{\mathrm{d}p_0}{\mathrm{d}t} = -(\lambda_{1АП} + \lambda_{2АП} + \lambda_{3АП} + \lambda_{4АП})p_0 + \mu_{1АП}p_1 + \mu_{2АП}p_2 + \mu_{3АП}p_3 + \mu_{4АП}p_4;$$

$$\frac{\mathrm{d}p_1}{\mathrm{d}t} = \lambda_{1АП}p_0 - \mu_{1АП}p_1; \frac{\mathrm{d}p_2}{\mathrm{d}t} = \lambda_{2АП}p_0 - \mu_{2АП}p_2;$$

$$\frac{\mathrm{d}p_3}{\mathrm{d}t} = \lambda_{3АП}p_0 - \mu_{3АП}p_3; \quad \frac{\mathrm{d}p_4}{\mathrm{d}t} = \lambda_{4АП}p_0 - \mu_{4АП}p_4;$$

$$p_0 + p_1 + p_2 + p_3 + p_4 = 1.$$

Начальные условия: $p_0(0)=1$, $p_i(0)=0$, $i=(\overline{1,4})$.

Рассчитаем вероятность безотказной работы этих функций с точки зрения надежности программного обеспечения и СУБД без учета аппаратных средств. Схема взаимодействия ПО и СУБД представлена на рис. 6.

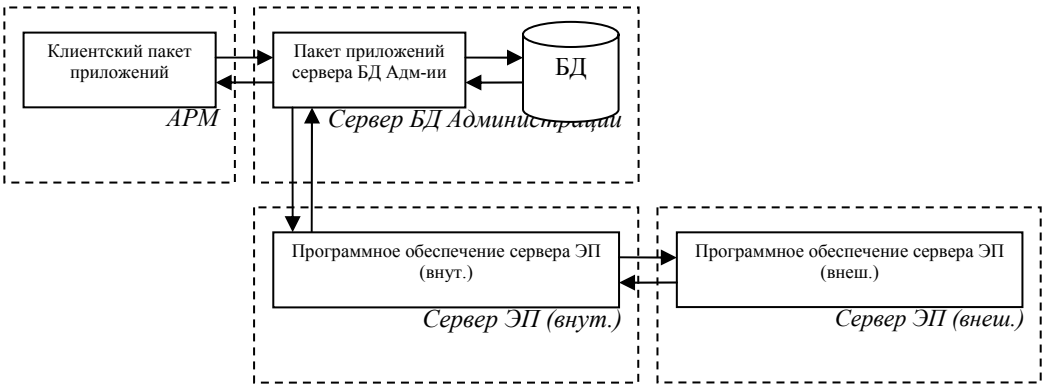


Рис. 6. Схема взаимодействия ПО и СУБД

С точки зрения ПО и СУБД выделяем следующие возможные состояния подсистемы: E_0 – подсистема работоспособна; E_1 – отказ клиентского приложения автоматизированных рабочих мест; E_2 – отказ пакета приложений сервера БД Администрации; E_3 – отказ СУБД Администрации; E_4 – отказ ПО сервера электронной почты (внутр.); E_5 – отказ ПО сервера электронной почты (внеш.). Взаимосвязь выделенных состояний описывается ориентированным графом, представленным на рис. 7, $\lambda_{iПО}$ и $\mu_{iПО}$ – соответственно интенсивности отказа и восстановления i-й части ПО, $i = (\overline{1,5})$.

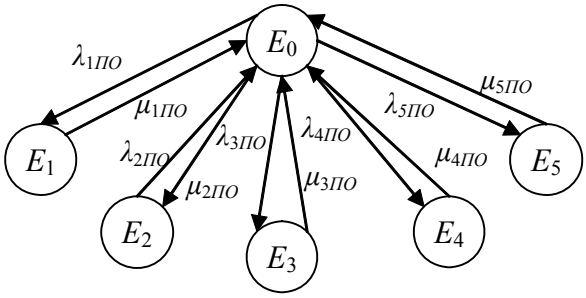


Рис. 7. Взаимосвязь выделенных состояний

Система дифференциальных уравнений, описывающих работу данной подсистемы, выглядит следующим образом [4]:

$$\frac{\mathrm{d}p_0}{\mathrm{d}t} = -(\lambda_{1ПО} + \lambda_{2ПО} + \lambda_{3ПО} + \lambda_{4ПО} + \lambda_{5ПО})p_0 + \mu_{1ПО}p_1 + \mu_{2ПО}p_2 + \mu_{3ПО}p_3 + \mu_{4ПО}p_4 + \mu_{5ПО}p_5;$$

$$\frac{\mathrm{d}p_1}{\mathrm{d}t} = \lambda_{1ПО}p_0 - \mu_{1ПО}p_1;\ \ \frac{\mathrm{d}p_2}{\mathrm{d}t} = \lambda_{2ПО}p_0 - \mu_{2ПО}p_2;\ \frac{\mathrm{d}p_3}{\mathrm{d}t} = \lambda_{3ПО}p_0 - \mu_{3ПО}p_3;$$

$$\frac{\mathrm{d}p_4}{\mathrm{d}t} = \lambda_{4ПО}p_0 - \mu_{4ПО}p_4;\ \frac{\mathrm{d}p_5}{\mathrm{d}t} = \lambda_{5ПО}p_0 - \mu_{5ПО}p_5;$$

$$p_0 + p_1 + p_2 + p_3 + p_4 + p_5 = 1.$$

Начальные условия: $p_0(0)=1$, $p_i(0)=0$,, $i=(\overline{1,5})$.

Задаваясь числовыми значениями интенсивностей отказа и восстановления [6], получим вероятность безотказной работы подсистемы при выполнении рассматриваемых функций с учетом аппаратно-программных средств, программного обеспечения и СУБД $p=p_{АП}\cdot p_{4ПО}=0{,}9993\cdot 0{,}9995=0{,}9988$.

Оценка надежности работы пользователя

Необходимо учитывать, что надежность выполнения отдельных функций, таких как функции ввода и редактирования, зависит не только от надежности работы аппаратно-программных и средств и программного обеспечения, но и от надежности работы пользователя.

Надежность работы пользователя определяется вероятностью ошибки. Значение вероятностей допущения ошибок, используемые при расчете, предлагается оценивать с помощью следующего подхода [9]. Для каждого n и заданного β можно построить область, внутри которой значение вероятности p совместимо с наблюденным в опыте значением частоты p^*. В нашем случае n – количество вводимых или редактируемых данных, k – число допущенных ошибок, β – доверительная вероятность, p – вероятность допущения ошибок в вводимых или редактируемых данных, p^* – частота допущения ошибок, $p^*=\frac{k}{n}$.

Выберем в качестве интервала p_1^*, p_2^* самый малый интервал, вероятность попадания левее которого и правее которого будет больше $\frac{\alpha}{2}$. Разрешая уравнение

$\sum_{m=k}^{n} C_n^m p_1^m (1-p_1)^{n-m} = \frac{\alpha}{2}$, можно найти нижнюю границу p_1 «доверительной области»; а из уравнения $\sum_{m=0}^{k} C_n^m p_2^m (1-p_2)^{n-m} = \frac{\alpha}{2}$ можно найти верхнюю границу p_2.

Но мы предполагаем, что данные вводят и редактируют квалифицированные операторы, которые допускают незначительное количество ошибок. Необходимо оценить надежность работы пользователя, т.е. вероятность допущения им ошибок в вводимых или редактируемых данных.

Пусть производится ввод или редактирование n данных, ни в одном случае не допущено ошибки. Задана доверительная вероятность β; требуется построить доверительный интервал для вероятности p события A, точнее, найти его верхнюю границу p_2, так как нижняя граница p_1, естественно, равна нулю. Верхняя граница определяется по формуле $p_2 = 1 - \sqrt[n]{1-\beta}$.

Зная вероятности безотказной работы каждой вспомогательной функции с учетом работы аппаратно-программных средств, программного обеспечения, СУБД и пользователя, можем посчитать вероятность безотказной работы всей функциональной подсистемы, реализующей данные вспомогательные функции.

Оценки чувствительности показателей надежности информационных систем

Основной задачей теории чувствительности динамических систем [10] является оценка степени влияния изменения значений начальных условий и параметров системы на протекание во времени процессов, описываемых системой дифференциальных уравнений вида:

$$\frac{dy}{dt} = \varphi(t, y, a), \tag{6}$$

$$y(t_0) = y_0. \tag{7}$$

Здесь $y(t)$ – вектор характеристик моделируемого процесса; t_0, y_0 – начальные условия процесса; a – вектор параметров, влияющих на его протекание. Методы этой теории позволяют построить векторную и матричные функции: $\xi(t) = \left[\frac{dy}{dt_0}(t)\right]$, $\eta(t) = \left[\frac{\mathrm{d}y}{\mathrm{d}y_0}(t)\right]$, $\gamma(t) = \left[\frac{\mathrm{d}y}{\mathrm{d}a}(t)\right]$, называемые функциями чувствительности решения задачи Коши (6), (7) вида

$$y = y(t, t_0, y_0, a) \tag{8}$$

к изменению значений параметра t_0, а также векторов y_0 и a. Здесь выражение (8) представляет собой общую форму записи решения задачи Коши (6), (7) [11].

В теории надежности [5] аналогом модели (6), (7) является система линейных дифференциальных уравнений:

$$\frac{dp_i}{dt} = \sum_{j=1}^{n} a_{ij} p_j, \; i=\overline{1,n}, \; t \geq 0 \tag{9}$$

с начальными условиями

$$p_i(0) = p_{i0}, \; i=\overline{1,n}. \tag{10}$$

Кроме этого, в модель оценки надежности системы включается условие вида

$$\sum_{i=1}^{n} p_i(t) = 1 . \qquad (11)$$

Здесь функции $p_i(t)$ описывают вероятности того, что в момент времени t рассматриваемая система находится в i-м состоянии (работоспособна, ремонтируется и т.п.). Элементы матрицы $A=[a_{ij}]_{n\times n}$ зависят определенным образом, обусловленным решаемой задачей, от векторов интенсивностей отказов $\lambda=(\lambda_1, \lambda_2, \dots, \lambda_N)$ и восстановлений $\mu=(\mu_1, \mu_2, \dots, \mu_N)$ отдельных элементов из числа N элементов рассматриваемой системы. Правило построения системы вида (9) описано в упомянутой выше работе [5] и в ряде других работ, посвященных теории надежности.

В этих работах на основе модели (9)–(11) решается задача определения коэффициента готовности системы $K_Г = P_c(\infty)$ или ее функции надежности $P_c(t)$ при фиксированном значении времени t.

Отметим, что в работе [12] рассматривались модели оценки чувствительности показателей надежности неремонтируемых изделий, информационных систем с учетом их ремонта, аппаратно-программных средств системы и прикладного программного обеспечения к ошибкам пользователя. Были получены соотношения, с помощью которых оценивается влияние изменения различных факторов и начального состояния системы на показатели надежности информационных систем. Это позволяет определить способы повышения надежности информационных систем.

Заключение

В работе были сформулирована постановка задачи расчета функциональной надежности КИС, разработаны инженерные математические модели оценки надежности аппаратно-программных средств и функциональных элементов КИС.

В качестве объекта исследований при разработке методики расчета функциональной надежности была выбрана одна из основных подсистем существующих КИС. Разработана методика вычисления основных показателей надежности рассматриваемой функциональной подсистемы. Приведенные алгоритмы расчета надежности были реализованы программно. В связи с отсутствием статистических данных по отказам компонент КИС разработаны методики сбора и обработки экспертной информации для вычисления интенсивностей отказов аппаратно-программных средств КИС.

Были освещены методы оценки чувствительности показателей надежности КИС.

Литература

1. ГОСТ 24.701-86 Надежность автоматизированных систем управления. Общие положения. М.: изд-во стандартов, 1986. 17 с.
2. Мейко А.В. Методика расчета функциональной надежности корпоративной информационной системы. / Микроэлектроника и информатика – 2005. 12-я Всероссийская межвузовская научно-техническая конференция студентов и аспирантов: Тезисы докладов. М.: МИЭТ, 2005. С. 276.
3. Бройдо В.Л. Вычислительные системы, сети и телекоммуникации. СПб: Питер, 2003. 688 с.
4. Кук Д., Бейз Г. Компьютерная математика. М.: Наука, 1990. 383 с.
5. Вентцель Е.С. Исследование операций. М.: Советское радио, 1972. 55 с.
6. Мейко А.В. Статистические и экспертные оценки показателей надежности компонент корпоративных информационных систем. VIII Королевские чтения: Всероссийская молодежная научная конференция, Самара, 4–6 октября 2005 года: Тезисы

докладов. Самара: Издательство Самарского государственного аэрокосмического университета имени академика С.П. Королева, 2005. С. 320.
7. Зиновьев П.А., Моисеев В.С., Мейко А.В. Модели для оценки надежности архитектурных решений корпоративных систем хранения данных. / Исследования по информатике. Выпуск 9. Научно-практическое издание. Институт проблем информатики АН РТ, Казань: Отечество, 2005. С. 93–102.
8. Кемени Дж., Снелл Дж. Конечные цепи Маркова. М.: Наука, 1970.
9. Вентцель Е.С. Теория вероятностей. М: Наука, 1969. 572 с.
10. Надежность автоматизированных систем управления / Под ред. Я.А. Хетагурова. М.: Высш. школа, 1988.
11. Понтрягин Л.С. Обыкновенные дифференциальные уравнения: Учебник для университетов. 4 изд. М.: Наука, 1974.
12. Моисеев В.С., Мейко А.В., Зиновьев П.А. К задаче оценки чувствительности показателей надежности информационных систем. / Исследования по информатике. Выпуск 10. Научно-практическое издание. Институт проблем информатики АН РТ, Казань: Отечество, 2006. С. 51–64.

ПРОЕКТИРОВАНИЕ И ТЕХНИЧЕСКАЯ РЕАЛИЗАЦИЯ ЭЛЕКТРОННОГО УЧЕБНИКА С ИСПОЛЬЗОВАНИЕМ PHP/MYSQL

Т.В. Мицкевич, А.А. Ржеутская
(Белорусский государственный технологический университет)
Научный руководитель – к.ф.-м.н., доцент Н.И. Гурин
(Белорусский государственный технологический университет)

В статье описано проектирование структуры электронного учебника и технический способ его реализации. В структуру учебника заложены функции виртуального преподавателя. Техническая реализация основана на странице-шаблоне, где динамические элементы формируются при помощи языка программирования PHP и базы данных MySQL.

Введение

Мультимедиа-технологии прочно вошли в обучающий процесс специалистов технических и гуманитарных специальностей. Электронные учебные средства на сегодняшний день представлены в широком ассортименте и разрабатываются как вузами, так и коммерческими предприятиями с использованием различных программных оболочек и языков программирования. Они представляют собой как простые HTML и PDF-документы [1], так и сложные обучающие системы, созданные по принципу работы экспертной системы [2]. Однако вопросы внедрения в электронный учебник виртуального преподавателя, способы технической реализации эффекта присутствия преподавателя, эффективного контроля пошаговой работы студента остаются актуальными. Поэтому мы предлагаем структуру электронного учебника, которая бы включала в себя функции виртуального преподавателя.

1. Проектирование структуры электронного учебника

В мультимедийном учебнике должны быть реализованы следующие функции:

- регистрация студента;
- индивидуализация обучения студента;
- накопление информации о студенте;
- анализ его действий во время изучения материала;
- промежуточный и итоговый контроль.

Кроме того, в него должны быть вложены функции преподавателя: четко ставить задачу обучения, предоставлять возможность работы с материалом, использовать имитационное моделирование, разъяснять трудные моменты, подсказывать, проводить промежуточный контроль, анализировать работу в целом с материалом и практическими заданиями и, в конечном итоге, ставить оценку, учитывая полную работу студента с учебником.

Таким образом, необходимо создать систему, структура которой должна обеспечить выполнение всех вышеуказанных функций и требований.

В настоящее время существуют два направления систем организации электронного обучения:

- коммерческие («Битрикс: Управление сайтом», «NetCat», «inDynamic 2.3», «Amiro.CMS», Система «Прометей» и др.);
- свободно распространяемые («ATutor», «Claroline», «Dokeos», «LAMS», «Moodle», «OLAT», «OpenACS», «Sakai» и др.).

Системы с открытым кодом позволяют решать распространенные задачи, в том числе коммерческие, у пользователей есть возможность доработки и адаптации конкретной системы к своим требованиям. Но практически все такие системы

имеют ошибки, которые достаточно сложно исправить, не говоря о сложности дополнения их другими функциями, которые должны соответствовать нашим требованиям. Поэтому было принято решение разработать собственную систему, в которую заложены функции, указанные выше.

Структура разрабатываемого учебника была разделена на области пользователя и области администратора (преподавателя).

Область пользователя включает авторизацию, страницы с основным текстом, страницы с дополнительным текстом (оглавление, введение, справочный аппарат, страницы с реакцией виртуального преподавателя), мультимедиа-компоненты, систему тестирования.

Область администратора позволяет преподавателю проводить анализ работы студента с учебником (анализ статистических данных). Кроме того, в этой области можно редактировать все компоненты учебника.

Каждая страница учебника имеет единые элементы оформления. Поэтому было принято решение разработать одну страницу-шаблон, которая включает статические, относительно динамические и динамические области.

Предусмотрены следующие статические элементы:

- навигационная панель (на которой расположены кнопки «Вперед», «Назад»);
- основная панель (на которой расположены кнопки «Оглавление», «Глоссарий», «Справка», «Выход» со ссылкой на соответствующие дополнительные страницы);

Динамические элементы – следующие:

- область информации о текущем пользователе;
- динамически меняющийся заголовок с названием текущего раздела и подраздела учебника;
- текстовая область.

Весь текст учебника разделен на смысловые страницы. Смысловая страница учебника – это логически и информационно завершенный фрагмент текста основного учебного материала, который в зависимости от своей семантики может занимать одну или несколько экранных страниц (во втором случае используется область с прокруткой).

Предусмотрены также такие кнопки, которые активны или неактивны в зависимости от того, в каком разделе находится студент. Это кнопки со следующими функциями:

- постановка задачи – кнопка работает в пределах одного и того же раздела и предназначена для введения студента в учебный материал;
- вопрос преподавателю – потенциально возможный вопрос преподавателю по учебному материалу. Преподаватели хорошо знают, какие потенциальные вопросы могут задать студенты по тому или иному материалу. При нажатии на данную кнопку появляется ответами виртуального преподавателя;
- обучающий тест – тест с мгновенной ответной реакцией, т.е. он имеет кнопку «Проверить», при нажатии которой студент сразу узнает, правильно или неправильно он ответил на вопрос;
- реакция преподавателя – предполагает какие-либо действия виртуального преподавателя в зависимости от семантики текстовой страницы. Это может быть реализовано в разных формах, например, в виде сообщения «Важно обратить внимание», «Вы должны обязательно запомнить», «Попробуйте решить задачу». Формы представления такой «виртуальной реакции» и способы ее включения в электронный учебник – предмет исследования.

В структуру учебника будет внедрен блок статистики, который будет собирать информацию о работе студента: даты и время работы, результаты прохождения промежуточных тестов, ведение журнала работы.

Многофункциональная динамическая обучающая система должна учитывать различные способы работы с ней, поэтому в структуре такого учебника предусмотрены режимы: «Просмотр» и «Изучение».

Постоянное «сопровождение» и контроль студента во время изучения материала нецелесообразно, особенно когда студент в первый раз бегло просматривает материал в целом или, напротив, просматривает его перед тем, как пройти тестирование, чтобы освежить материал в памяти. Поэтому добавлен режим «Просмотр», в котором отключены все динамические кнопки, ведение статистики, работа виртуального преподавателя.

Режим «Изучение» (или полнофункциональный режим) предусматривает активизацию всех динамических элементов.

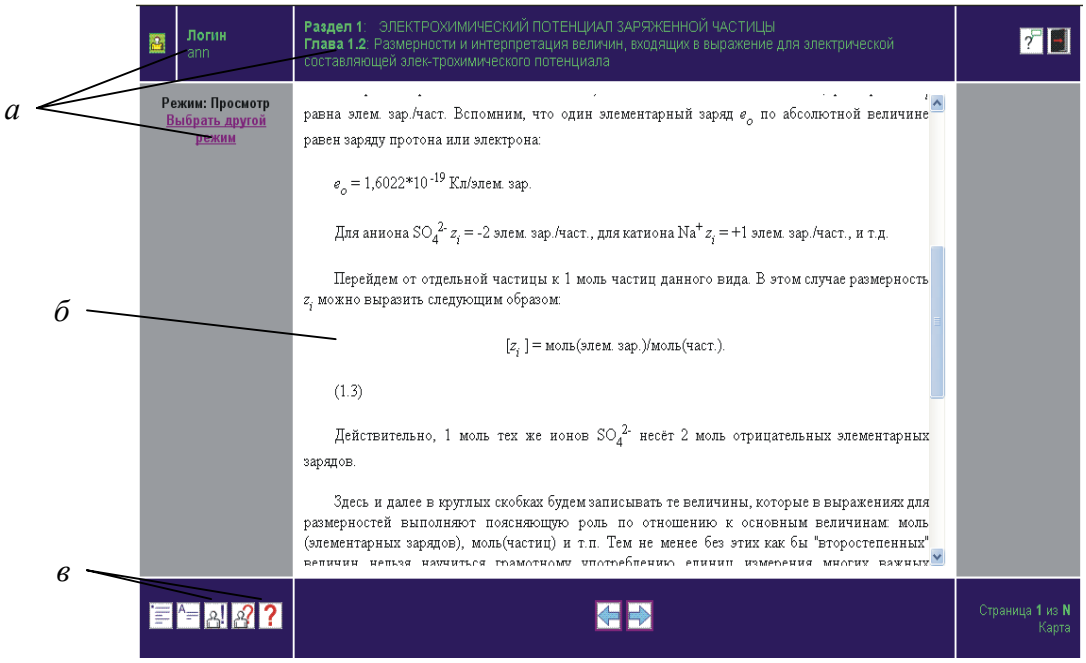


Рис.1. Схема дизайна страницы-шаблона электронного учебника: *а* – динамические элементы; *б* – смысловая страница; *в* – кнопки с реакцией преподавателя

Область администратора позволяет управлять процессом обучением и редактировать компоненты электронного учебника.

На данном этапе работы над электронным учебником по электрохимии в области администратора доступны следующие функции:

- редактирование информации о пользователях;
- редактирование текстов;
- редактирование названия глав и разделов;
- редактирование обучающих тестовых заданий;
- редактирование контрольных тестов.

2. Техническая реализация

При создании электронного учебника использовались следующие средства:

- HTML-редактор Dreamweaver;
- язык программирования PHP;
- базы данных и язык обработки запросов MySQL.

Для создания мультимедиа-компонентов использовались:

- Flash и язык программирования ActionScript (анимация процессов);
- Pinnacle Studio (видеомонтаж).

Перед программной частью электронного учебника ставятся следующие задачи:

- интерактивность;
- модульность;
- возможность внесения изменений (административная часть);
- мотивация студента на самообразование;
- легкость использования (как прикладная часть, так и административная);
- нетребовательность к ресурсам ПК.

Все данные электронного учебника хранятся в базе данных (MySQL). Основной каркас системы содержит следующие уровни: уровень базы данных, уровень языка MySQL, на котором происходит общение с базой данных, уровень php, обрабатывающий результаты запросов и генерирующий готовые страницы, уровень шаблонов, организующий представление данных.

Разрабатываемая нами CMS (Content management system – система управления содержимым) имеет модульную структуру. Каждому модулю соответствует один или несколько файлов php, который организовывает работу с базой данных и может подключать другие модули и шаблоны.

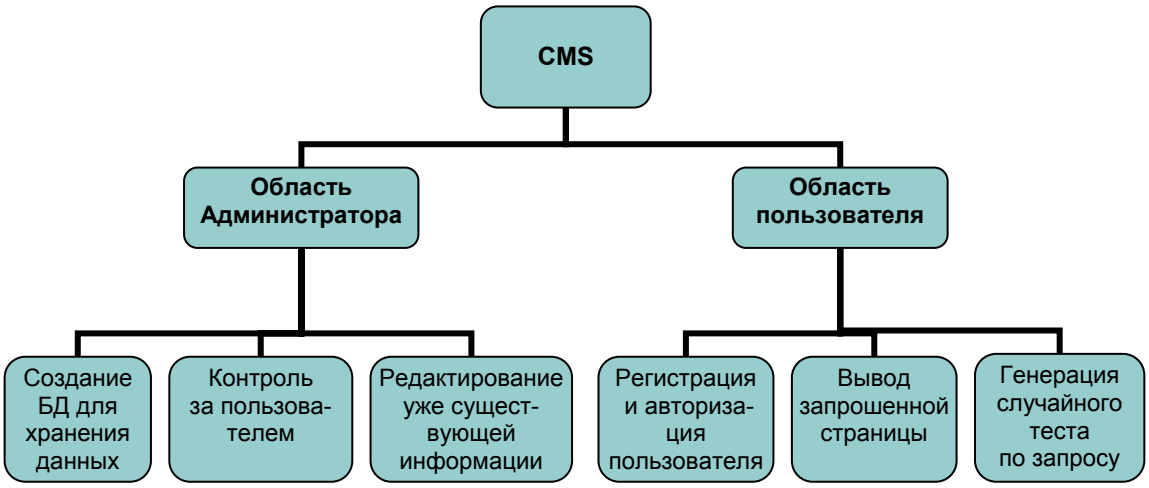


Рис.2. Функции областей администратора и пользователя электронного учебника

2.1. Модули области администратора

AdminIndex. Выводит на экран список из всех существующих модулей с возможностью перехода к работе с ними.

*Install***.** Создает в указанной базе данных необходимые для работы CMS таблицы. Сообщает пользователю, если такие таблицы уже существуют.

UserEdit. У администраторов есть возможность просмотреть и отредактировать информацию о пользователе. В этом модуле осуществляется активация учетной записи пользователя, если запрещена авторегистрация. Любую учетную запись пользователя можно временно деактивировать и удалить.

TextEdit. Администраторы могут добавлять, редактировать и удалять все тексты книги.

TitlesEdit. Позволяет администратору добавлять, редактировать и удалять названия глав и разделов.

TestEdit. С помощью этого модуля происходит добавление в базу данных тестов для самоконтроля (один вопрос – один тест). Поддерживаются флеш-тесты.

2.2. Модули области пользователя

UserReg. Позволяет пользователю послать запрос на авторизацию (если запрещена авторегистрация) или зарегистрироваться (если разрешена авторегистрация). Доступен с главной страницы.

Contents. Оглавление запрашивается в базе данных – выводятся все названия глав и разделов, содержащиеся в соответствующих таблица БД. Ссылки на страницы генерируются автоматически.

Test. Вывод случайного теста из БД. Определяет количество тестов в базе, генерирует случайный номер теста и выводит этот тест на экран пользователя.

Parser. Вывод на страницу-шаблон динамической информации из разных таблиц базы данных учебника. Обработчик получает запрос в виде системы переменных, содержащей требуемый номер страницы, запрашивает необходимую информацию из БД: название раздела, главы, содержание страницы, номер страницы. Имя авторизованного пользователя и режим работы берутся из сессионных переменных. Вся эта информация передается в виде переменных шаблону, который и представляет страницу пользователю.

Заключение

Разработанные модули позволяют создать современный мультимедийный учебник. В данный момент ведется работа над электронным учебником по электрохимии.

Предложенная технология создания электронного учебника с внедрением функций виртуального преподавателя может быть успешно применена для организации обучающего процесса в Белорусском государственном технологическом университете.

В дальнейшей работе планируется создание модулей «Виртуальный преподаватель» и «Статистика». Кроме того, на основе разработанной системы можно создать целый ряд учебников различной тематической направленности.

Литература

1. Мицкевич Т.В. Проблемы создания PDF-архивов учебных изданий. // Труды БГТУ. Сер. IX. Издат. дело и полиграфия. 2006. Вып. XIV. С. 135–136.
2. Усачев Ю.Е. Проектирование интеллектуального учебника. // Дистанционное образование. 2000. № 4. С. 24–27.

ТЕХНОЛОГИЯ РАДИОЧАСТОТНОЙ ИДЕНТИФИКАЦИИ (RFID). ПЕРСПЕКТИВЫ ИСПОЛЬЗОВАНИЯ И ВОЗНИКАЮЩИЕ ПРОБЛЕМЫ

А.А. Бобцов, Д.А. Камнев, А.С. Кремлев, С.А. Топилин
Научный руководитель – д.т.н., профессор В.В. Григорьев

Рассмотрена технология радиочастотной идентификации и перспективы ее использования при решении ряда прикладных задач в различных областях деятельности человека. Указаны основные проблемы, возникающие при внедрении технологии, и предложены возможные варианты их решения.

Введение

Технология радиочастотной идентификации (RFID – Radio Frequency Identification), позволяющая дистанционно идентифицировать различные физические объекты, появившаяся в середине XX века, в последние годы совершенствуется особенно интенсивно, что связано с развитием микроэлектроники, позволившим реализовать многие идеи технологии RFID, ранее недоступные по чисто технологическим причинам, а также с появлением стандартов, применение которых обеспечило совместимость технических решений от разных производителей [1].

На протяжении продолжительного периода времени вычислительные машины, способные с относительно высокой скоростью обрабатывать и надежно хранить данные, не решали проблему автоматического сбора и ввода-вывода информации в систему управления любым процессом – производством, торговлей, транспортными перевозками и т.п. Неотъемлемым условием надежного функционирования любой автоматизированной системы является абсолютная достоверность наличной информации. Технология RFID наиболее полно соответствует всем требованиям компьютеризированной системы управления, где необходимы распознавание и регистрация объектов и их прав в реальном режиме времени [2].

Свое широкое распространение системы RFID стали получать с середины 90-х годов. По сравнению с уже существовавшими тогда методами идентификации по штрихкоду, по магнитной полосе или по контактному ключу (технология TouchMemory фирмы Dallas), системы RFID обладали рядом существенных преимуществ. Они позволили существенно ускорить процесс идентификации, не требовали специального расположения объекта относительно идентифицирующего устройства, как в системах со штрихкодом, были более надежными, долговечными и защищенными, чем системы с магнитной полосой, работали бесконтактно, в отличие от систем TouchMemory [3].

На сегодняшний день в отдельных областях применение технологии RFID давно стало привычной практикой. В частности, более десяти лет прошло с тех пор, как компания Ford выпустила первый иммобилайзер на основе системы RFID. Однако только три-четыре года назад начали активно рассматриваться возможности более масштабного использования технологии в различных областях человеческой деятельности, что, в том числе, связано и с наличием ряда определенных проблем, возникающих при разработке и внедрении систем RFID [4].

В данной статье рассматривается технология радиочастотной идентификации (RFID), ее физические основы и особенности. Дана классификация основных элементов систем радиочастотной идентификации. Рассмотрены перспективы использования технологии при решении ряда прикладных задач в различных областях практической деятельности человека. Указаны основные проблемы, возникающие при внедрении технологии, а также предложены возможные варианты их решения.

Основы технологии RFID

Технология радиочастотной идентификации основана на обмене информацией между радиоответчиком, так или иначе связанным с объектом, и устройством опроса (считывателем), излучающим через антенну в окружающее пространство электромагнитную энергию. Когда радиоответчик, называемый радиометкой или транспондером, проходит через зону чтения ридера, он сообразно хранящейся в нем информации изменяет его сигнал и возвращает назад. Этот сигнал принимается антенной считывателя, обрабатывается его электронным блоком и по интерфейсу направляется в компьютер, где и определяется идентификатор метки [5].

Принцип работы типовой системы RFID показан на рис. 1.

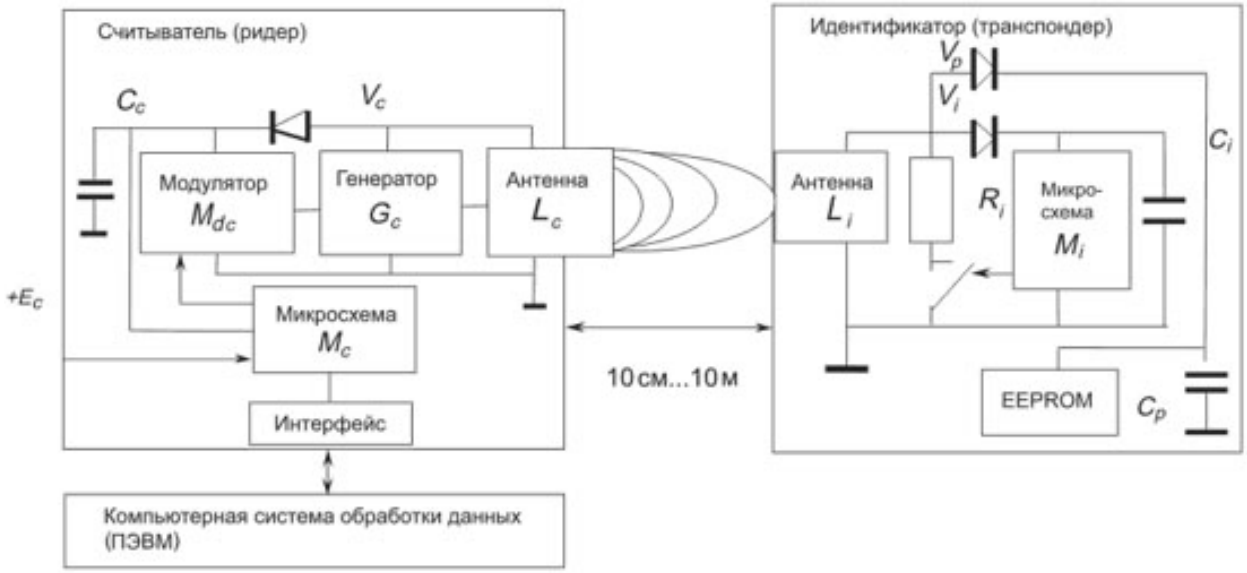


Рис. 1. Принцип работы RFID-системы

Считыватель содержит генератор высокой частоты Gc, который запитывает антенну считывателя Lc. За счет наличия электромагнитной связи между антенной считывателя и антенной идентификатора Li в последней наводится переменное напряжение, величина которого зависит от конструктивного исполнения и расстояния между картой и считывателем. Наведенное напряжение используется для питания микросхемы Mi через выпрямитель, образованный диодом Vi и фильтрующим конденсатором Ci. Микросхема Mi модулирует напряжение в антенне Li путем ее шунтирования резистором Ri сообразно необходимой для передачи кодовой последовательности. За счет связи антенн модуляция появляется в антенне считывателя Lc, детектируется диодом Vc и поступает на микросхему считывателя Mc, которая дешифрирует принятый сигнал и через интерфейс посылает его в компьютерную систему обработки данных.

При использовании идентификаторов, способных не только передавать информацию считывателю, но и получать ее для целей программирования или управления режимом работы метки, считыватель RFID-системы также содержит модулятор Mdc, модулирующий излучаемый им сигнал, тем самым осуществляя передачу определенной кодовой последовательности, а идентификатор – детектор Vp и перепрограммируемую энергонезависимую память EEPROM, в которую записывается передаваемая считывателем информация [1].

Таким образом, по видам памяти радиометки разделяются на:

- “RO” (Read Only) – данные записываются только один раз сразу при изготовлении. Такие метки пригодны только для идентификации. Никакую новую информацию в них записать нельзя, и их практически невозможно подделать.
- “WORM” (Write Once Read Many) – кроме уникального идентификатора, такие метки содержат блок однократно записываемой памяти, которую в дальнейшем можно многократно читать.
- “RW” (Read and Write) – такие метки содержат идентификатор и блок памяти для чтения-записи информации. Данные в них могут быть перезаписаны большое число раз.

Кроме того, по типу питания радиометки разделяются на:

- активные – используют для передачи данных энергию встроенного элемента питания. Они обычно программируются так, чтобы излучать свой сигнал через определенные промежутки времени (например, 1 раз в секунду). Дистанция, на которой возможно чтение таких меток может доходить до 100 метров;
- пассивные – используют энергию, излучаемую считывателем. Дистанция регистрации подобных меток существенно меньше, сильно зависит от мощности считывателя и находится в пределах 0,1–10 метров.

По исполнению, зависящему от конкретных целей и условий использования, известны следующие радиометки:

- самоклеящиеся бумажные или лавсановые метки;
- стандартные пластиковые карты;
- дисковые метки;
- различные виды брелоков;
- метки в специальном исполнении для жестких условий эксплуатации.

Типовая конструкция карты для бесконтактной RFID-идентификации представлена на рис. 2.

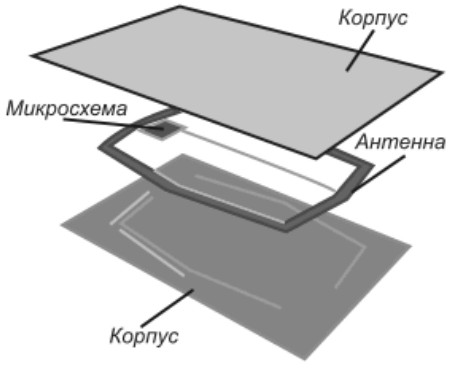


Рис. 2. Типовая конструкция карты RFID

Ридер может иметь различное исполнение – от простого переносного сканера до стационарного туннельного устройства, которое сканирует упаковки по мере их продвижения по конвейеру. Стационарные ридеры обладают большей зоной чтения и мощностью, обычно напрямую подключены к системе обработки информации. Мобильные – обладают сравнительно меньшей дальностью действия, имеют внутреннюю память для хранения полученных данных.

Антенны могут быть встроены в специальные сканеры, а также в ворота, турникеты, дверные проемы и т.п. для получения информации от предметов или людей, проходящих через зону ее действия. Конструктивно антенна может быть выносной или находиться вместе с ридером в одном корпусе [6].

На сегодняшний день для использования технологии RFID наибольшее распространение получили три частотных диапазона: 125 кГц, 13,56 МГц и 2,45 ГГц. Для ре-

шения той или иной прикладной задачи наиболее пригоден конкретный частотный диапазон, что наглядно иллюстрируется рис. 3.

В зависимости от используемой частоты RFID-системы можно разделить условно на три группы:

1. высокочастотные (850–950 МГц и 2,4–5 ГГц) – используются там, где требуются большое расстояние и высокая скорость обмена информацией;
2. промежуточной частоты (10–15 МГц) – используются там, где должны быть переданы большие массивы данных;
3. низкочастотные (100–500 КГц) – используются там, где допустимо небольшое расстояние между объектом и ридером.

Для каждого из диапазонов используются свои методы кодирования сигналов, свои скорости передачи и алгоритмы разрешения коллизий, т.е. ситуаций, когда сигналы идентификаторов, одновременно находящихся в поле считывателя, накладываются друг на друга. Механизм антиколлизии обеспечивает выборочную поочередную работу с несколькими идентификаторами. В процессе антиколлизии считыватель определяет все идентификаторы по их уникальным серийным номерам, а затем поочередно обрабатывает [2].

Рис. 3. Зависимость параметров RFID-системы от используемой частоты

Перспективы использования технологии RFID

Технология радиочастотной идентификации существует и используется с 1940-х гг. Во время второй мировой войны ВВС США использовали радиопередатчики для идентификации самолетов. Но лишь несколько лет назад технология начала активно использоваться в сфере производства, торговли, транспорта и т.д.

Прежде всего, технология RFID позиционировалась как удачная альтернатива штрих-коду. Основное преимущество радиоидентификации состоит в том, что этот метод позволяет максимально автоматизировать процесс опознавания. Считывание информации из метки осуществляется на расстоянии и не требует, в отличие от считывания штрих-кода, размещения считывающего устройства в определенном положении по отношению к метке. Информационная емкость метки существенно больше емкости штрих-кода; информацию в метках можно перезаписывать, а саму метку – спрятать; радиометки более устойчивы к воздействию окружающей среды. Радиоидентификация

обеспечивает более высокую скорость приемки и отправки маркированных товаров, так как несколько десятков меток можно считать в течение 1–2 секунд. Это ценное преимущество для логистических процессов [4].

На сегодняшний день поток новостей о разнообразных приложениях, в которых уже сейчас с успехом применяются RFID, растет ежемесячно. Новый обзор рынка RFID и перспектив его развития, сделанный компанией ID TechEX, позволил сделать вывод, что достигший в 2004 г/ уровня 1,9 млрд. долларов рынок RFID, в 2008 г. превысит 7 млрд. долларов и вырастет до 26,9 млрд. к 2015 г.

Результаты проведенного анализа позволяют сформулировать следующие перспективные области применения RFID-технологии.

1. Системы контроля доступа (СКД).
2. Учет и безопасность транспортных средств.
3. Техническое обеспечение спортивных и зрелищных мероприятий.
4. Защита компьютерных систем и телекоммуникаций от несанкционированного доступа.
5. Системы против краж для предприятий, квартир, магазинов.
6. Контроль и сопровождение объектов в технологических процессах.
7. В животноводстве, птицеводстве (вживление электронных меток под кожу).
8. Определение местоположения железнодорожных вагонов, автофургонов.
9. В метрополитене: пассажирские карты, учет рабочего времени кассиров, машинистов и т.п.
10. Лекарства – обработка заказов по кодам контейнеров.
11. Маркировка бочек в пивоваренной промышленности и винных заводах.
12. Выставочные экспонаты – «оживление» экспонатов при подходе гида.
13. Электронная подпись для лиц, работающих на опасных объектах (например, в нефтегазодобывающей и угольной промышленности).
14. Магазины – выдача и перемещение товаров и материальных ценностей.
15. Службы аварийного оповещения и спасения (например, МЧС) [7].

Применение технологии радиочастотной идентификации ведет к улучшению учета, управления и безопасности ресурсов, снижению издержек, повышению производительности, снижению потерь времени и более эффективному использованию оборудования и персонала. На сегодняшний день – это ключевая технология в таких областях, как безопасность, транспортные перевозки, производство, торговля и др.

Таким образом, перспективы применения технологии радиочастотной идентификации – самые широкие: от учета изделий на производстве и ценностей в учреждениях культуры до защиты продукции от подделок. RFID-метки начали получать широкое внедрение и, по прогнозам, обещают стать самой популярной и массовой технологией в мире, способной конкурировать по распространению с мобильными телефонами [8].

Проблемы технологии RFID

Перечислим основные проблемы, возникающие при разработке и внедрении технологии RFID.

1. Возможное экранирование сигнала при размещении на металлических поверхностях. Радиочастотные метки подвержены влиянию металла (это касается упаковок определенного вида – металлических контейнеров, иногда даже некоторых типов упаковки жидких пищевых продуктов, запечатанных фольгой). Это приводит к необходимости использования более дорогих меток, разработанных специально для установки на металлические поверхности, или к нестандартным способам закрепления меток на объекте.

2. Подверженность помехам в виде электромагнитных полей от включенного оборудования. Эта проблема практически не актуальна для систем диапазона 868–869 МГц, так как в этом диапазоне не работают другие приборы, но низкочастотные метки, работающие на частоте 125 КГц, подобному влиянию подвержены.

3. Возможность подделки сигнала ответа метки.

4. Возможность заражения меток вирусом.

5. Относительно высокая стоимость качественных меток.

Стоимость пассивной радиочастотной метки составляет от 0,15 доллара (при приобретении свыше 1 000 000 шт.) до 3 долларов (при приобретении 1 шт.). В случае с метками защищенного исполнения эта цена может достигать 5 и более долларов. Таким образом, стоимость RFID-меток на порядок превышает стоимость этикеток со штриховым кодом. Исходя из этого, использование радиочастотных меток целесообразно для защиты дорогих товаров от краж или для обеспечения сохранности изделий, переданных на гарантийное обслуживание. В сфере логистики и транспортировки грузов стоимость радиочастотной оказывается совершенно незначительной по сравнению со стоимостью содержимого контейнера, поэтому совершенно оправдано использование радиочастотных меток на упаковочных ящиках, паллетах и контейнерах.

6. Взаимные коллизии (перекрытие сигналов нескольких считывателей или одновременный ответ нескольких меток).

7. Отсутствие исследований влияния на организм человека.

Для решения проблемы подверженности систем радиочастотной идентификации помехам в виде электромагнитных полей от включенного оборудования могут быть предложены следующие варианты:

1. конструктивное решение – создание радиоответчиков, использующих специфические особенности поверхностных акустических волн;
2. математическое решение – применение алгоритмов компенсации паразитного сигнала, запаздывания, помех измерительного устройства с использованием методов, предложенных в работах [9–11].

Кроме того, использование поверхностных акустических волн ведет к увеличению числа возможных кодовых комбинаций, хранящихся в радиоответчике, без увеличения размера его микрочипа. Благодаря этому радиоответчику может быть присвоен дополнительный идентификационный номер, что может использоваться в качестве решения проблемы подделки метки. Также использование поверхностных акустических волн снижает стоимость радиоответчика.

Для своевременного предотвращения распространения вируса среди радиоответчиков необходима их систематическая программная проверка антивирусной программой.

Заключение

В статье рассмотрена активно развивающаяся в настоящее время технология радиочастотной идентификации (RFID), ее физические основы и особенности. Предложена классификация основных элементов систем RFID по их специфическим характеристикам. Рассмотрены перспективы будущего использования технологии при решении ряда прикладных задач в различных областях практической деятельности человека. Указаны основные проблемы, возникающие при внедрении систем RFID, а также предложены возможные варианты их решения.

Литература

1. Стасенко Л. Современные технологии радиочастотной идентификации // Системы безопасности, 2004. № 2(56).
2. Стасенко Л. Тихая революция // БДИ, 2004. №3 (54).
3. Барсуков В.С., Пономарев А.А. Беспроводные технологии «последнего дюйма». // Специальная техника, 2005. №1.
4. Насакин Р. Универсальные ярлыки // Компьютерра, 2007. №1–2.
5. Рувинова Э. Радиочастотная идентификация. Бесконтактная технология. // Электроника. Наука, технология, бизнес. 2004. №6.
6. Гудин М., Зайцев В. Технология RFID: реалии и перспективы. // Компоненты и технологии. 2003. №4.
7. Федоров М. Стандарты и тенденции развития RFID-технологий. // Компоненты и технологии. 2006. № 1.
8. Басина Н. RFID – перспективы и реальность. // CIO. 2006. №9.
9. Бобцов А.А., Кремлев А.С. Синтез наблюдателя в задаче компенсации конечномерного квазигармонического возмущения. // Известия РАН. Теория и системы управления, 2005. №3. С. 5–11.
10. Бобцов А.А., Кремлев А.С. Адаптивная идентификация частоты смещенного синусоидального сигнала // Известия вузов. Приборостроение. 2005. №4. С. 22–26.
11. Бобцов А.А., Кремлев А.С., Цвикевич В.И. Синтез наблюдателя для конечномерного возмущения. // Научно-технический вестник СПбГИТМО (ТУ). 2003. Выпуск 7. Информационные, вычислительные и управляющие системы. С. 81–85.

WNETMESS – ПРОГРАММА ДЛЯ ОБМЕНА МУЛЬТИМЕДИЙНЫМИ ДАННЫМИ МЕЖДУ МОБИЛЬНЫМИ УСТРОЙСТВАМИ ПО WI-FI И ENTERNET-ПОДОБНЫМ СЕТЯМ

С.Е. Антонов, Н.С. Токалов
Научный руководитель – кандидат технических наук, старший преподаватель А.П. Ищенко

WNetMess является программой передачи сообщений между пользователями КПК на базе Windows Mobile 5.0. В программе реализована возможность передачи текстовых и графических сообщений, а также файлов. Передача сообщений ведется при помощи Wi-Fi, что обеспечивает возможность использования данного программного продукта в условии отсутствия подключения к Интернет.

Введение

WNetMess является программой передачи разнообразной информации между пользователями КПК, коммуникаторов и др. устройств на базе Windows Mobile 5.0, Windows Mobile 2003. В дальнейшем планируется расширение списка поддерживаемых устройств, выделение возможности связи компьютеров с различными ОС, имеющими подключение к единой сети. Станет возможной связь внутри локальной сети, например, КПК и ноутбука, смартфона и коммуникатора, без использования сети Интернет. Появиться возможность использовать единую службу сообщений, доступную всем устройствам.

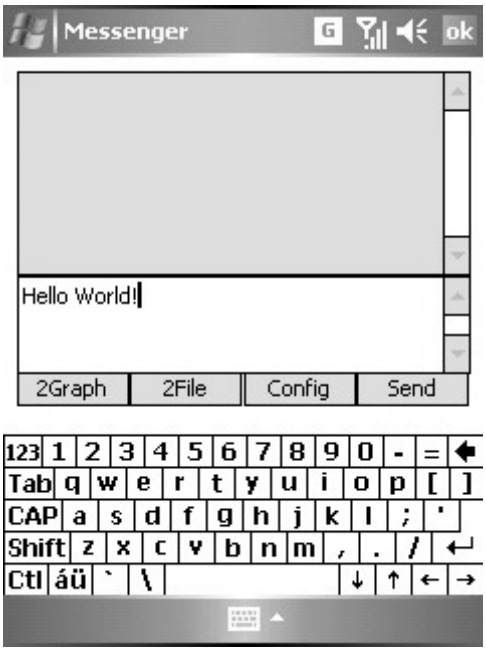


Рис. 1. Стартовое диалоговое окно программы WNetMess

На данный момент наибольшую потребность в единой службе сообщений, не требующей выделения отдельного сервера IRC или ICQ или выхода в Интернет, испытывают мобильные устройства (КПК, коммуникаторы и т.п.) на базе Windows Mobile. Разработчики реализовали в мобильных устройствах возможности использования сетевых интерфейсов и сетевых сервисов – например, через прямое подключение КПК к настольному компьютеру, имеющему подключение к локальной сети, или используя Wi-Fi. У КПК есть возможность работы с файлами на удаленных компьютерах. Однако разработчики в целях безопасности не реализовали возможность открытия файлов и папок КПК на общий доступ, не реализовали поддержку Mail-Slot'ов для Windows Mobile. Это делает невозможным прямой обмен файлами и сообщениями между КПК,

подключенными к единой локальной сети (без использования сети Интернет, т.е. e-mail и подобных сервисов). С помощью программных продуктов сторонних разработчиков можно создать на одном КПК Ftp-сервер, подключиться к нему другим КПК и обмениваться файлами. Увы, такие программы обычно не бесплатны и довольно громоздки, сложны, чтобы использовать их повсеместно. Они чаще всего не дают возможности простого обмена текстовыми сообщениями между пользователями КПК.

Программа WNetMess (рис. 1) создана, чтобы решить проблему передачи любых сообщений напрямую между пользователями КПК. WNetMess призвана сделать процесс передачи данных между КПК простым и доступным каждому. Несомненным преимуществом программы является то, что распространяется она бесплатно. Также бесплатно распространяется и ее исходный код. Это довольно актуально, учитывая тот факт, что в данный момент информации по программирование под КПК на базе Windows Mobile в Интернете очень мало, а на русском языке она вообще практически отсутствует. Примеров кода более-менее сложных приложений, работающих под Windows Mobile, нет вообще, так как в большинстве своем они являются коммерческими. Свободное распространение кода программы – это своеобразная «рука помощи» тем, кто только начинает осваивать программирование под Windows Mobile. Следовательно, создание данного программного продукта, в некоторой мере, носит и просветительский характер. Помимо этого, такой подход к распространению поможет привлечь новых разработчиков для участия в развитии проекта.

Концептуальные особенности работы объектной модели программы

Исходный код программы разделен на классы, в которых реализованы основные принципы концепции ООП, такие как: наследование, перегрузка функций, виртуальные функции, ограничение прав доступа пользователя кода к «опасным» элементам класса. Некоторые классы имеют ограничения на создание объектов класса, т.к. не могут существовать без своих наследников. Так, например, нельзя отдельно инициировать создание объекта класса сетевой единицы, возможно использование функций данного класса только в его наследниках.

Общие иерархии взаимосвязи в классах программы представлены на рисунках 3, 4 и 7. Рассмотрим принципы функционирования основных классов.

Класс для работы с рисованием: MyDraw

В программе реализована возможность совместного рисования между пользователями. Синхронизация рисунков пользователей выполняется по нажатию на кнопку «Send». На данном этапе реализована работа интерфейсов рисования всего в двух цветах: «свой», «чужой». Однако потенциальная возможность использования разноцветных рисунков есть.

Класс MyDraw реализует основные функции работы программы с диалогом коллективного рисования (рис. 2). Схема взаимодействия классов, реализующих рисование, представлена на рис. 3. При инициализации класса MyDraw ему передается указатель на окно, в котором объект должен работать. Картинка представляется в векторной форме и в программе хранится как динамический массив объектов класса MDPoint. Объект класса MDPoint содержит информацию о координатах точки и ее цвете. Используемые по умолчанию цвета доступны для изменения, в том числе и динамического. Линия представляет собой набор точек класса MDPoint. Окончание линии маркируется точкой класса MDPoint со значением идентификатора цвета, равным нулю.

В классе реализована возможность простой адаптации рисунка к различным разрешениям экрана. Для этого достаточно в основной программе на событие изменения

размера экрана передать в функцию масштабирования рисунка новые размеры экрана или области, доступной для рисования. В программе вычисляется коэффициент соотношения полученных данных о размере экрана к размерам экрана 320×240. Далее координаты точек преобразуются к матрице 320×240 и в таком виде хранятся в памяти. При отрисовке изображения, координаты точек при помощи коэффициентов преобразуются к размерам экрана. Толщина кисти также зависит от разрешения экрана. В классе присутствует функция быстрого стирания (или закрашивания определенным цветом) активной для рисования области экрана. Таким образом, рисунок получается устойчивым к изменению графических режимов компьютера, к изменению разрешения экрана при передаче данных между пользователями программы. Эта возможность является очень актуальной, так как в различных моделях КПК и смартфонов разрешение экрана может сильно отличаться.

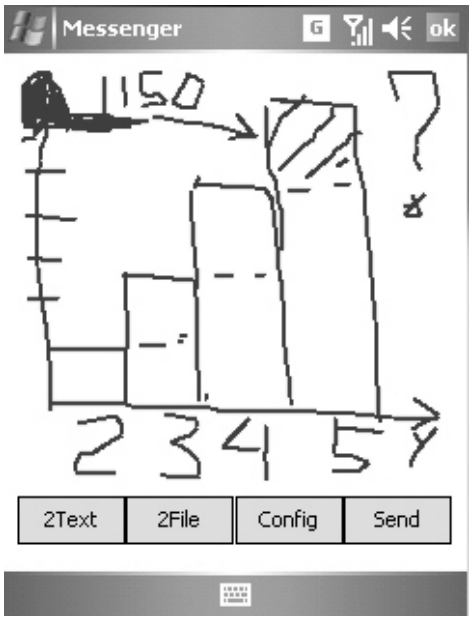


Рис. 2. Окно коллективного рисования

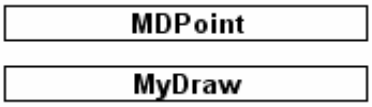


Рис. 3. Иерархия классов рисования

Сетевые взаимодействия, классы для работы с сетью

Классы NNode, NServer, NSender предназначены для обеспечения сетевых взаимодействий между пользователями программы. Структура взаимодействия классов представлена на рис. 4. Класс NNode описывает абстрактную сетевую единицу. Это очень удобно, так как по сути классы сервера и клиента отличаются только тем, что класс сервера четко «знает», на какой порт к нему будут подключаться, и не имеет возможности сам подключаться к другим сетевым единицам, а класс клиента, напротив ,имеет возможность подключаться к удаленному компьютеру на указанный порт. На своем компьютере класс клиента создает порт со случайным номером, по которому в дальнейшем и происходит «общение» сервера и клиента. После установления соединения и получения идентификатора подключения типа SOCKET, сетевые единицы ведут себя практически абсолютно одинаково – они могут отправлять и принимать сообщения, разрывать соединение, ожидать данные. Однако функции отправки и получения сообщения клиента и сервера немного отличаются, поэтому они в классе NNode объяв-

лены виртуальными и нулевыми. Они описаны (перегружены) в классах наследниках NServer, NSender. Такой подход дает возможность использовать эти функции в классе NNode, создавая более обобщенные функции отправки, например, текстового сообщения, типа, размера сообщения, сообщений типа «Ping-Pong». В связи с тем, что создание объекта типа NNode не имеет смысла, оно запрещено. Для этого конструктор класса NNode объявлен как «protected» элемент. В конструкторе класса NNode происходит инициализация основных переменных сетевой единицы. В классах-наследниках сначала вызываются конструкторы класса NNode, а затем уже собственные конструкторы, которые инициализируют специфические переменные каждого типа сетевой единицы.

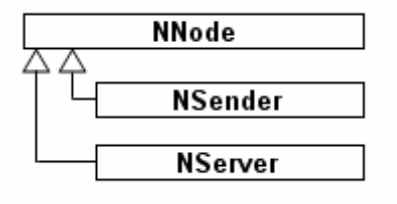


Рис. 4. Иерархия классов для работы с сетью

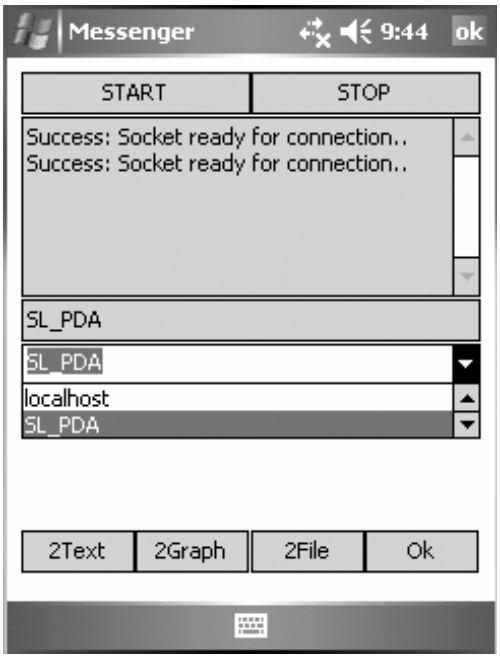


Рис. 5. Диалог управления сетевыми подключениями

Серверная компонента программы имеет возможность одновременной работы с несколькими подключениями. При запуске сервера в программе создается новый поток и выставляется флаг того, что сервер уже запущен. Невозможно запустить одновременно два потока сервера. В созданном потоке создается объект класса NServer и в бесконечном цикле запускается функция прослушивания определенного порта. При установлении соединения с сервером создается новый поток, в который передается указатель на новый сервер, который уже будет в дальнейшем непосредственно работать с подключившимся клиентом. Основной же сервер продолжает прослушивание в основном потоке сервера. Максимальное количество подключений к серверу указывается в параметрах класса сервера. Сервер, работающий с клиентским подключением, в цикле читает сообщения от клиента и, если требуется, реагирует на них.

Сообщения передаются в определенном бинарном формате. При окончании передачи и обработки сообщения клиент и сервер обмениваются пакетами «ping-pong», чтобы удостовериться в доставке данных и проверить их корректность. Если в процессе получения сообщения сервер или клиент, получающий данные, обнаруживают ошибку

– например, неизвестный тип сообщения, отрицательный или слишком большой размер получаемого сегмента данных, то получение сообщения прекращается и выдается ошибка передачи данных.

В начале каждого сообщения передается его тип, который представляет собой символ типа char. Типы сообщений описаны константами в классе NNode. Далее у текстовых и графических сообщений передается размер получаемого пакета данных, и далее сами данные. Полученные данные преобразуются сначала к типу «void*», а затем к нужному типу: для текстовых сообщений к «wchar_t*», для графических сообщений к «MDPoint*». Полученные текстовые сообщения отправляются на вывод в интерфейс программы, а графические сообщения передаются объекту класса MDPoint.

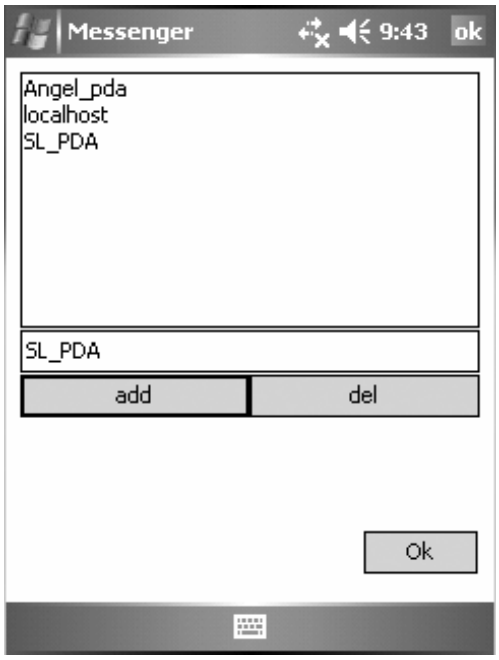


Рис. 6. Диалог управления списком контактов

Несколько иначе происходит передача файлов. При передаче файлов порядок действий немного другой, так как не владелец файла подключается к удаленному компьютеру для его передачи, а получатель файла «просит» сервер передать ему данные. Происходит все следующим образом: сначала клиент посылает серверу запрос с «просьбой» передать ему файл, затем сервер проверяет наличие у себя открытых для скачивания файлов, и, если файл есть, то начинает передачу, если нет – посылает клиенту сообщение об ошибке. Передача начинается с передачи типа сообщения (файл или ошибка). Для того чтобы не загружать файл целиком в оперативную память компьютера, файл условно разбивается на части, которые по мере надобности загружаются и обрабатываются. После типа сообщения клиенту передается количество частей, на которые будет разбит файл при доставке. Далее передается длина имени файла и само имя файла. В случае успешной доставки предыдущих сообщений начинается передача частей файла. В начале каждой части передается информация о номере пакета, размере передаваемых данных. Затем из файла загружается указанная часть и передается клиенту. Клиент, получая очередную часть файла, проверяет ее номер и размер и, если все корректно, сохраняет ее в файл. В конце передачи файла выполняется проверка синхронизации.

По окончании получения сообщения и синхронизации клиент посылает сообщение об окончании передачи данных, и связь разрывается. Далее уничтожается объект сервера, работающий с данным подключением, и уничтожается его поток.

В программе ведется «лог» работы с сервером. Лог и компоненты управления сервером вынесены в отдельный раздел (рис. 5). Также в этом разделе указывается пользователь, с которым в данный момент ведется работа. Однако, скачивать открытые файлы могут все пользователи сети, и присылать сообщения и рисунки могут также все пользователи сети. Пользователя для общения можно выбрать из списка пользователей, редактирование которого производится после нажатия на кнопку «list» (рис. 6).

Работа клиентской части программы, т.е. посылки разнообразных сообщений по сети, реализуется объектами класса NSender и ведется не в основном потоке приложения. При нажатии на кнопку отправки сообщения (отправки рисунка, скачки файла) создается новый поток, в котором и ведется передача данных. При запуске потока создается объект класса NSender, через который программа подключается к серверу и ведет с ним диалог. Новый поток создается для того, чтобы не «тормозить» работу интерфейса программы при долгой отправке сообщений (например, из-за недоступности конкретного пользователя в данный момент, из-за проблем в работе сети, или из-за скачки большого файла с сервера). По окончании сеанса клиент посылает серверу сообщение об окончании сессии и закрывает подключение, далее поток данной сессии уничтожается. В режиме отправки текста возможна одновременная (из разных потоков) отправка нескольких сообщений (но число одновременно используемых потоков ограничено). Для экономии ресурсов системы для данного клиента при скачке файлов возможно только одно подключение к серверу. Также для данного клиента возможно только одно подключение к серверу при передаче рисунков. При посылке графического сообщения передаются только новые части рисунка. Информация об уже посланных данных хранится в специальном параметре класса MyDraw. При попытке отправить пустое сообщение (как текстовое, так и графическое) выдается сообщение об ошибке, или это сообщение игнорируется программой.

Потоки и взаимодействия между ними

Как уже было сказано выше, WNetMess работает с несколькими потоками, количество которых варьируется в зависимости от этапа работы программы в данный момент.

Главный поток занимается взаимодействием с пользователем программы – обрабатывает переходы по разделам, нажатия на кнопки, создает при необходимости новые потоки. Поток создается при создании программы и уничтожается при ее завершении.

Основной поток сервера программы работает в отдельном потоке. Он запускается при нажатии кнопки «START» и уничтожается при нажатии кнопки «STOP» в диалоге управления подключениями, либо он может уничтожиться при критических ошибках в работе основного сервера.

При подключении к основному серверу клиента основной сервер создает новый поток для работы с конкретной сессией. Основной сервер может создавать ограниченное число таких потоков (максимальное их количество указано в константном параметре класса NNode). Ограничения на количество одновременных подключений к серверу наложены для экономии производительных мощностей компьютера, которые, к слову, у КПК не являются такими уж большими. Данный поток создается в момент подключения клиента к основному серверу и уничтожается при завершении сессии или при критических ошибках при работе с клиентом.

При инициировании пользователем отправки любого сообщения на сервер, как уже было сказано выше, создается клиентский поток. Этот поток создается для того, чтобы не задерживать работу программы при отправке большого сообщения, при получении с сервера большого файла, при сбоях в работе сети, или при отсутствии в данный момент доступа к нужному пользователю. Поток уничтожается после завершения сессии работы клиента или при критических ошибках при работе с сервером. За получение

файлов с сервера, отправку рисунков, отправку текстовых сообщений отвечают разные потоки. Потоков для отправки текстовых сообщений может быть несколько. Максимальное их число определяется переменной-параметром. Возможность создания сразу нескольких потоков очень удобна, например, если долго происходит установление соединения с сервером, при этом можно писать подряд несколько сообщений, и они дойдут сразу после установления соединения. Для передачи файлов можно создавать только один поток, чтобы сильно не загружать компьютер. Для передачи рисунков также возможно создание только одного потока, так как создание нескольких неоправданно – первый подключившийся поток и так передаст все доступные данные (изменения картинки) серверу.

В режиме рисования при необходимости вызывается поток перерисовки картинки. Данная операция выделена в отдельный поток и срабатывает через небольшой промежуток времени после наступления события «WM_PAINT» (и др.), так как при переотрисовке картинки, иногда, из-за «подтормаживания» стирания элементов ОС, случаются наложения слоев, которые приводят к некорректному отображению пользовательского рисунка. Поток уничтожается после завершения отрисовки картинки.

Взаимодействие между потоками осуществляется с использованием глобальных переменных «семафоров». В момент, когда поток использует данные некоторого более или менее сложного объекта, он выставляет значение «семафора» в значение «занято». Если в этот момент к этим же данным хотят обратиться другие потоки, то они ждут, пока переменная освободится. Когда поток заканчивает работу с данными, он выставляет «семафор» в значение «свободно», после чего любой другой поток может использовать эти данные. Таким образом, в программе реализованы взаимодействия различных потоков с текстовыми блоками («лог», форма вывода текстовых сообщений), с функцией отправки рисунка, с функцией скачки файла. Чтобы была возможность создавать только определенное количество потоков, как, например, для отправки текстовых сообщений, вводится глобальный счетчик, и, прежде чем создать поток, функция проверяет, не превышен ли лимит по количеству потоков. На входе в поток счетчик увеличивается на единицу, при уничтожении потока – уменьшается на единицу. Так, кроме потоков отправки текстовых сообщений клиента, контролируется, не превышен ли лимит подключений к серверу. Если лимит превышен, то новый поток, обрабатывающий данную сессию, создан не будет, а в «лог» сервера будет записано предупреждение.

Классы управления пользовательским интерфейсом

Классы управления пользовательским интерфейсом (рис. 7) служат своего рода оболочкой, позволяющей пользователю добавлять новые формы и элементы управления, не затрагивая структуры приложения. Этот подход дает возможность разработки пользовательских интерфесов (в том числе, дальнейшего развития нашего проекта) пользователями, не владеющими MFC или владеющими в недостаточной мере для грамотного внедрения своего кода в уже имеющийся проект без нарушения взаимодействий.

Класс WNM_Interface_Manager предназначен для хранения информации об имеющихся формах (слоях). Являясь, по своей сути, контейнером слоев, данный класс предоставляет методы добавления слоя и взятия информации о существующих. Предполагается, что в приложении будет только один экземпляр данного класса, что позволит избежать коллизий, возможных при отрисовки изображения.

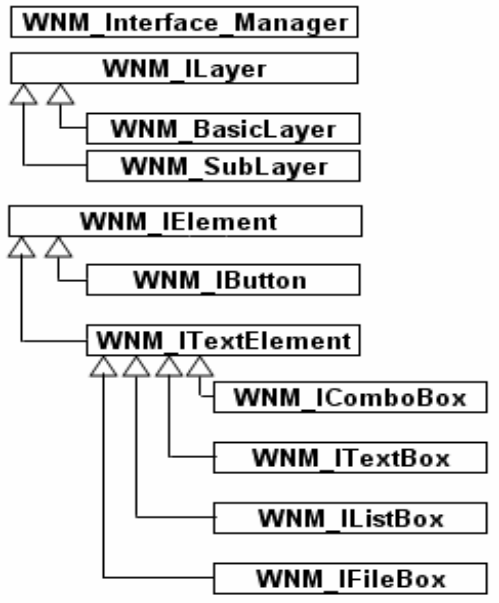


Рис. 7. Иерархия классов управления интерфейсом

Класс WNM_ILayer описывает абстрактную единицу формы, определяя лишь методы, общие для всех видов форм – преимущества такого подхода уже упоминались выше. На данный момент выделены два вида форм WNM_BasicLayer – формы, при добавлении которых в приложения в корневом меню приложения появляется кнопка перехода на данную форму, и WNM_SubLayer – формы, переход на которые осуществляется либо по выбору подменю, либо по нажатию на одну из кнопок.

Все объекты ввода/вывода (WNM_ITextElement) или управления (WNM_IButton), располагающиеся на одном из слоев, относятся к классу WNM_IElement, который также является абстрактным. В свою очередь, наследники класса WNM_ITextElement отличаются между собой незначительно (только функциями обработки содержащегося в них текста).

Заключение

Созданная программа WNetMess решает проблему передачи текстовых и графических сообщений, а также файлов напрямую между пользователями КПК. Программа имеет интуитивный интерфейс и занимает небольшое пространство на диске, что, несомненно, является положительным аспектом в рамках мобильных приложений.

Информации по программированию Windows Mobile 5.0 в Интернете очень мало, а на русском языке она вообще практически отсутствует, в то время как на английском, в основном, идет лишь описание некоторых возможностей программирования на C# или Visual Basic. Примеров кода более-менее сложных приложений, работающих под Windows Mobile, нет вообще, тогда как исчерпывающего описания создания приложения на C++ нет даже в MSDN. Бесплатно распространяемый исходный код в сочетании с достаточно прозрачной архитектурой и развернутым комментированием является мощнейшим подспорьем для разработчиков приложений для Windows Mobile. WNetMess не только делает передачу файлов такой же доступной, как по BlueTooth или IrDA, а передачу сообщений такой же удобной, как по ICQ, но и предоставляет разработчикам готовое решение для их проектов.

Литература

1. http://channel9.msdn.com/wiki/default.aspx/MobileDeveloper.VoIP

2. http://www.mobilab.ru/
3. http://www.programmersheaven.com/zone25/cat831/index.htm
4. http://www.eps-media.com/article.aspx?quickid=0609051&page=4
5. http://safari5.bvdep.com/0130255920/ch08
6. http://www.pocketpcdn.com/
7. http://www.codeproject.com/ce/
8. http://xda-developers.com/
9. http://rsdn.ru/article/dotnet/Writing_Mobile_Code.xml

3 ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ

ОЦЕНКА ПРОПУСКНЫХ СПОСОБНОСТЕЙ КАНАЛОВ СВЯЗИ В РАСПРЕДЕЛЕННЫХ СЕТЯХ ПЕРЕДАЧИ ДАННЫХ

Нгуен Дык Тай
Научный руководитель – д.т.н., профессор Т.И. Алиев

Предлагается решение задачи оценки пропускных способностей каналов связи в распределенных СПД при ограничениях на время доставки пакетов или на стоимость сети. Разработанный метод ориентирован на сети произвольной топологии и позволяет учесть алгоритм маршрутизации, вариант размещения прикладных программ и наборов данных по узлам сети, способ взаимодействия пользователей сети. Разработанные методы положены в основу программного комплекса, позволяющего проводить исследование СПД с учетом особенностей реальных сетей.

Введение

Исследование сетей передачи данных (СПД) предполагает построение математических моделей, позволяющих получить зависимости характеристик исследуемой СПД от структурно-функциональных и нагрузочных параметров. В качестве моделей СПД широко используются модели в виде сетей массового обслуживания (СеМО), в которых узлы, представляющие собой системы массового обслуживания (СМО), отображают задержки при передаче пакетов по каналам связи [1–4]. При этом точные аналитические результаты могут быть получены в том случае, если СеМО является экспоненциальной, т.е. узлы представляют собой СМО типа M/M/1 в символике Кендалла [2]. Задача оценки пропускных способностей каналов связи в СПД с использованием модели в виде разомкнутой экспоненциальной СеМО (РЭСеМО) с однородным потоком заявок была решена методом множителей Лагранжа в [2]. При этом задача решалась как математическая оптимизационная задача, и абсолютно никак не учитывались специфические особенности, присущие реальным распределеным СПД. К числу таких особенностей, в первую очередь, относятся неоднородность трафика, многообразие топологий и алгоритмов маршрутизации и т.д. Учет этих особенностей выполняется на этапе параметризации модели СПД, результаты которого оказывают существенное влияние на адекватность разрабатываемой модели.

В данной работе предлагается решение задачи оценки пропускных способностей каналов связи в распределенных СПД с неоднородным трафиком с учетом топологии сети, алгоритма маршрутизации, варианта размещения прикладных программ и наборов данных по узлам сети, способа взаимодействия пользователей сети. Разработанные методы реализованы в виде программного комплекса, обладающего наглядным графическим интерфейсом и позволяющего проводить исследование СПД с учетом особенностей реальных сетей.

Постановка задачи

Для описания исследуемой СПД используются:

1) структурные параметры, определяющие:
   - топологию СПД;
   - количество узлов n и каналов связи в СПД;
   - тип каналов связи (дуплексный, полудуплексный);
   - стоимостные коэффициенты каналов связи;

- длины каналов связи;

2) функциональные параметры, задающие:
   - способ взаимодействия пользователей сети;
   - способ маршрутизации для каждого узла СПД в виде маршрутной таблицы, содержащей основной и альтернативный маршруты;
3) нагрузочные параметры:

а) интенсивности $\Lambda_{i,j}$ потоков сообщений разных типов от пользователей, подключенных к узлам СПД; предполагается, что в сети циркулируют сообщения следующих типов:
   - запросы к прикладным программам и наборам данных, размещенных в узлах сети;
   - прикладные программы и наборы данных, передаваемые по запросу пользователям;
   - результаты выполнения прикладных программ и обработки данных в виде сообщений-ответов;
   - текстовые сообщения, которыми обмениваются пользователи;

б) длины сообщений разных типов:
   - прикладных программ L_P и наборов данных L_D;
   - запросов к прикладным программам $l_P^з$ и наборам данных $l_D^з$;
   - результатов выполнения прикладных программ l_P^o и обработки наборов данных l_D^o;
   - текстовых сообщений l_T;

в) средняя длина одного пакета $l_П$ и длина обрамления l_O.

Эти параметры используются для решения задачи параметризации модели СПД, представляемой в виде РЭСеМО, в результате которой задача оценки пропускных способностей каналов связи может быть сведена к задаче оптимизации РЭСеМО методом неопределенных множителей Лагранжа. Задача параметризации модели СПД заключается в пересчете параметров СПД, связанных с неоднородностью потоков, циркулирующих в сети. Кроме того, чтобы решить задачу оценки пропускных способностей каналов связи, необходимо рассчитать потоки пакетов в каналах. При этом расчет потоков реализуется программно на основе заданных внешних интенсивностей потоков сообщений, формируемых пользователями сети, известной топологии и маршрутизации, способов взаимодействия пользователей, а также нагрузочных параметров.

Оптимизация пропускных способностей каналов связи

В качестве модели распределенной СПД будем использовать разомкнутую СеМО, состоящую из *m* СМО, каждая из которых соответствует определенному каналу связи СПД. Принимается традиционное допущение, что поток пакетов, поступающих в каналы СПД, простейший, и время передачи пакетов в каждом из каналов, определяемое как отношение длины передаваемого пакета $l_П$ к пропускной способности канала *C*, распределено по экспоненциальному закону.

Метод расчета характеристик РЭСеМО можно найти в [2]. При этом среднее время доставки пакетов в сети, состоящей из *m* каналов связи, определяется следующим образом:

$$T_{cp} = \sum_{k=1}^{n}\sum_{l=1}^{n} \frac{\alpha_{(k,l)} l_{П}}{C_{(k,l)} - \lambda_{(k,l)} l_{П}},$$

где $C_{(k,l)}$ – пропускная способность канала связи (k,l); $\alpha_{(k,l)}$– среднее число «обращений» к каналу связи (k,l) в процессе передачи пакетов; $\lambda_{(k,l)}$ – интенсивность пакетов в канале связи (k,l). Стоимость S СПД определяется как

$$S = \sum_{k=1}^{n}\sum_{l=1}^{n} \beta_{(k,l)} C_{(k,l)} D_{(k,l)},$$

где $\beta_{(k,l)}$ – стоимостной коэффициент пропорциональности, отражающий стоимость единицы пропускной способности канала связи (k,l); $D_{(k,l)}$ – длина канала связи.

Задача оптимизации пропускных способностей каналов связи с использованием модели РЭСеМО может решаться в одной из двух постановок.

1. Минимизировать среднее время задержки пакетов в сети T_{cp} при ограничении на стоимость сети $S < S^*$. В этом случае, с учетом введенных обозначений, пропускная способность канала связи (k,l) вычисляется следующим образом ($k = \overline{1,n}$; $l = \overline{1,n}$; $k \neq l$):

$$C_{(k,l)} = \frac{S^* - \sum_{k=1}^{n}\sum_{l=1}^{n} \beta_{(k,l)} \lambda_{(k,l)} l_{П} D_{(k,l)}}{\beta_{(k,l)} D_{(k,l)}} \frac{\sqrt{\beta_{(k,l)} \lambda_{(k,l)} l_{П} D_{(k,l)}}}{\sum_{k=1}^{n}\sum_{l=1}^{n} \sqrt{\beta_{(k,l)} \lambda_{(k,l)} l_{П} D_{(k,l)}}} + \lambda_{(k,l)} l_{П}. \qquad (1)$$

2. Минимизировать стоимость СПД S при ограничении на среднее время задержки пакетов в сети $T < T_{cp}^*$. В этом случае пропускная способность канала связи (k,l) вычисляется по формуле ($k = \overline{1,n}$; $l = \overline{1,n}$; $k \neq l$):

$$C_{(k,l)} = \frac{1}{T_{cp}^*} \sum_{k=1}^{n}\sum_{l=1}^{n} \sqrt{\beta_{(k,l)} \alpha_{(k,l)} l_{П} D_{(k,l)}} \sqrt{\frac{\alpha_{(k,l)} l_{П}}{\beta_{(k,l)} D_{(k,l)}}} + \lambda_{(k,l)} l_{П}. \qquad (2)$$

Расчет потоков в каналах связи

Чтобы решить задачу определения пропускных способностей каналов связи с использованием формул (1) и (2), необходимо рассчитать интенсивности $\lambda_{(k,l)}$ потоков пакетов в каналах связи на основе заданных внешних интенсивностей потоков сообщений и известного алгоритма маршрутизации в СПД. Кроме того, расчет интенсивностей проводится с учетом выбранного варианта размещения прикладных программ и наборов данных по узлам СПД и заданного способа взаимодействия пользователей сети. Расчет потоков пакетов в каналах СПД состоит из трех этапов.

На первом этапе выполняется расчет интенсивностей $\lambda_{i,j}$ пакетов в СПД путем умножения внешних интенсивностей $\Lambda_{i,j}$ сообщений от пользователей на число пакетов в одном сообщении:

$$\lambda_{i,j} = k\Lambda_{i,j} \qquad (i, j = \overline{1,n}; \quad i \neq j),$$

где $k = 1,2,...$ – количество пакетов в сообщении.

На втором этапе расчет интенсивностей пакетов в каналах связи выполняется на основе вычисленных на первом этапе внешних интенсивностей потоков пакетов, заданных таблиц маршрутизации и вероятности передачи по основному пути. Интенсивности пакетов в каждом из каналов связи рассчитываются по формуле

$$\lambda_{(k,l)} = \sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_{i,j}X_{(k,l)}^{(i,j)} \qquad (k,l=\overline{1,n}; \quad k \neq l), \tag{3}$$

где $\lambda_{i,j}$ – интенсивность пакетов из узла i в узел j; $X_{(k,l)}^{(i,j)}$ – доля пакетов ($0 \leq X_{(k,l)}^{(i,j)} \leq 1$), проходящих по линии (k,l), причем должно выполняться условие сохранения потока в сети:

$$\sum_{k=1}^{n}X_{(k,l)}^{(i,j)} - \sum_{k=1}^{n}X_{(l,k)}^{(i,j)} = \begin{cases} -1, & l=i; \\ -0, & l \neq i; \\ 1, & l=j. \end{cases} \tag{4}$$

Формула (3) получена из условия, что интенсивность пакетов в канале связи (k,l) должна равняться сумме интенсивностей пакетов по всем путям, которые проходят через этот канал. Для того чтобы найти программно доли пакетов $X_{(k,l)}^{(i,j)}$ и, следовательно, интенсивности пакетов в каналах связи, используется обратная рекурсивная процедура, определяющая распределение интенсивностей потоков на паре (i – узел-источник, j – узел-назначение) и суммирующая их при рассмотрении всех пар узлов. Эта процедура реализуется следуюшим образом.

Сканируются все узлы, через которые пакеты могут пройти от узла-источника к узлу-назначению с учетом таблиц маршрутизации, и строится двоичное дерево, причем для каждого узла рассматриваются два пути передачи пакета: основной и альтернативный. СПД отличается от двоичного дерева тем, что ее топология произвольная. Чтобы устранить ситуацию, когда пакеты могут циркулировать в сети бесконечно, используется маскирование узлов, которые уже сканировались: Mask[узел]:=false.

На первом шаге обнуляются все доли $X_{(k,l)}^{(i,j)}$ пакетов во всех каналах связи и маскируется узел-источник i как просканированный узел: Mask[i]:= false.

При вызове рекурсивной процедуры доля пакетов в каждом из каналов связи рассчитается по правилу:

- если следующий рассматриваемый узел является основным и он не маскирован, то доля пакетов в этом канале увеличивается на величину, равную произведению вероятности передачи пакетов по основному пути на долю пакетов предыдущего канала связи;
- если следующий рассматриваемый узел является альтернативным и он не маскирован, то доля пакетов в этом канале увеличивается на величину, равную произведению вероятности передачи по альтернативному пути на долю пакетов предыдущего канала связи.

Рекурсивная процедура вызывается только тогда, когда следующий рассматриваемый узел в процессе сканирования не маскрован и не совпадает с узлом-назначения. Обратный механизм в рекурсивном алгоритме реализуется с помощью массива маскирования Mask[] - после вызова рекурсивной процедуры какой-то пары (узел-источник, узел-назначение), нужно отметить узел-источник как немаскированный: Mask[узел-источник]:=true.

Если основной путь рассматриваемого узла совпадает с узлом-назначения и альтернативный путь этого узла является маскированным узлом, то к доле пакетов основного канала связи добавляется доля пакетов альтернативного, равная произведению вероятности передачи пакета по альтернативному пути на долю пакетов предыдущего канала связи, так как пакеты не могут повторно попадать в один и тот же узел СПД. При этом обеспечивается условие (4).

На рис. 1 показан пример распределения по каналам связи долей пакетов, передаваемых в СПД с интенсивностью $\lambda_{1,4}$ от пользователей, подключенных к узлу 1, к пользователям, подключенным к узлу 4. Рекурсивная процедура обозначается как $R_h(k,l)$, где k – узел-источник, l – узел-назначение, h – порядковый номер вызова рекурсивной процедуры.

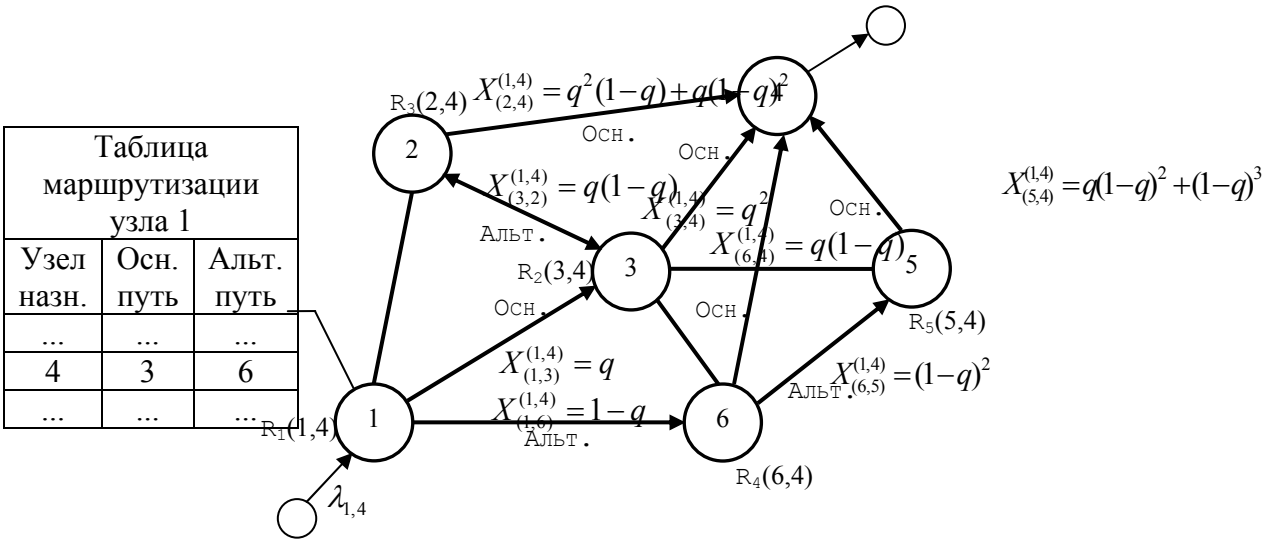


Рис. 1. Пример распределения долей пакетов $\lambda_{1,4}$

Распределение долей пакетов $\lambda_{1,4}$ в каналах связи для пары (1,4) реализовано с помощью рекурсивной процедуры $R_1(1,4)$ следующим образом. Процедура $R_1(1,4)$ определяет по таблице маршрутизации узла 1 следующие узлы основного пути (узел 3) и альтернативного пути (узел 6), по которым пакет должен пройти от узла 1 к узлу назначения 4. Затем выбирается основной путь (узел 3) и вызывается рекурсивная процедура $R_2(3,4)$. При этом доля $X_{(1,3)}^{(1,4)}$ пакетов в канале (1,3), соединяющем узел 1 и основной узел 3, будет вычисляться как произведение доли пакетов предыдущего канала (в этом случае она равна 1) и вероятности q передачи пакета по основному пути, т.е. $X_{(1,3)}^{(1,4)} = q$. В свою очередь, процедура $R_2(3,4)$ снова определяет по таблице маршрутизации узла 3 основной и альтернативный пути, по которым пакет должен пройти от этого узла к узлу назначения. Пусть основным путем является узел назначения 4, а альтернативным – узел 2. Доля $X_{(3,4)}^{(1,4)}$ пакетов в канале (3,4), соединяющем узел 3 и основной узел 4, будет вычисляться как произведение доли пакетов предыдущего канала (в этом случае она равна q) и вероятности q передачи пакета по основному пути, т.е. $X_{(3,4)}^{(1,4)} = q^2$. Поскольку узел 4 представляет собой узел назначения, то $R_2(3,4)$ выбирает альтернативный путь от узла 3 к узлу 4 путем вызова процедуры $R_3(2,4)$. При этом, доля $X_{(3,2)}^{(1,4)}$ пакетов равна произведению доли пакетов q предыдущего канала на вероятность (1–q) передачи пакетов по альтернативному пути, т.е. $X_{(3,2)}^{(1,4)} = q(1-q)$. Таким образом, рекурсивная процедура сканирует все узлы, по которым пакет пройдет от узла-источника к узлу назначения, рассчитывая доли пакетов в каналах связи. Эта процедура позволяет вычислить интенсивности пакетов в каналах связи, через которые пакеты проходят при передаче от узла-источника до узла назначения, т.е. между одной парой взаимодействующих узлов. Для того чтобы получить интенсивности пакетов разных типов в СПД между всеми узлами, нужно просуммировать интенсивности всех возможных передач для прикладных программ, наборов данных и текстовых сообщений.

На третьем этапе выполняется расчет коэффициентов передач для каждого канала связи *(k,l)* как отношение интенсивности пакетов в этом канале к суммарной (внешней) интенсивности поступления пакетов в СПД: $\alpha_{(k,l)} = \lambda_{(k,l)} / \lambda_{вн.}$, где

$\lambda_{вн.} = \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_{i,j}$ – суммарная (внешняя) интенсивность поступления пакетов в СПД.

Расчет потоков пакетов в каналах связи выполняется с учетом способов взаимодействия пользователей сети, которые существенно влияют на результаты расчетов потоков и, следовательно, пропускные способности каналов связи. В работе рассматриваются следующие способы взаимодействия пользователей:

a) RDA (Remote Data Access) – по запросу пользователя прикладная программа или набор данных пересылаются в узел пользователя и обрабатываются в нем;

b) DBS (DataBase Server) – по запросу пользователя прикладная программа и набор данных обрабатываются в узле-сервере, в котором они находятся, и пользователю пересылаются только результаты обработки;

c) AS (Application Server) – прикладные программы в процессе реализации в узле-сервере могут обращаться к наборам данных, расположенным в других узлах сети.

Программный комплекс

Рассмотренный метод оценки пропускных способностей каналов связи СПД реализован в виде программного комплекса, написанного на языке Visual C++ с использованием библиотеки базовых классов MFC с архитектурой Doc/View для удобства разработки и реализации приложений под Windows. Программа работает под ОС Windows 2000/XP с требованием памяти не более 64 Мбайт. Программа имеет наглядный графический интерфейс, позволяющий достаточно просто задавать структурно-функциональные и нагрузочные параметры исследуемой СПД.

Программа обладает следующими возможностями:

1) задание топологии исследуемой СПД одним из следующих способов (рис. 2):
   - графически путем указания мышкой на экране компьютера местоположения узлов и соответствующих каналов связи;
   - автоматически путем выбора из перечня типовых топологий, а именно: «звезда», «кольцо», «дерево», «полносвязная»;
   - аналитически путем указания декартовых координат узлов СПД или расстояний между ними (рис. 3);
2) построение и изменение произвольной топологии с использованием соответствующей матрицы связей (рис. 2);
3) выбор вариантов распределения прикладных программ и наборов данных по узлам СПД;
4) автоматическое создание всех таблиц маршрутизации в узлах по критерию «количество хостов» или «взвешенного графа» с возможностью изменения таблиц маршрутизации и вероятности передачи пакетов по основному пути;
5) расчет потоков пакетов в каналах связи и параметризация модели СПД;
6) расчет вероятностей передачи пакетов:
   - от пользователей к каналам связи;
   - между соседними каналами;
   - от каналов к пользователям сети;
7) расчет пропускных способностей каналов связи методом множителей Лагранжа;
8) исследование характеристик СПД путем варьирования структурно-функциональных и нагрузочных параметров.

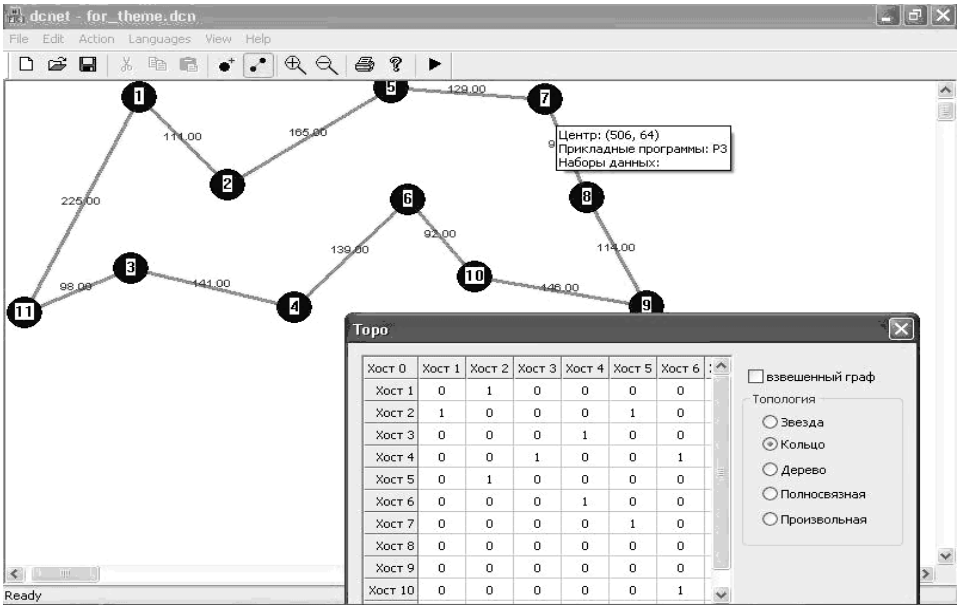


Рис. 2. Построение и изменение произвольной топологии

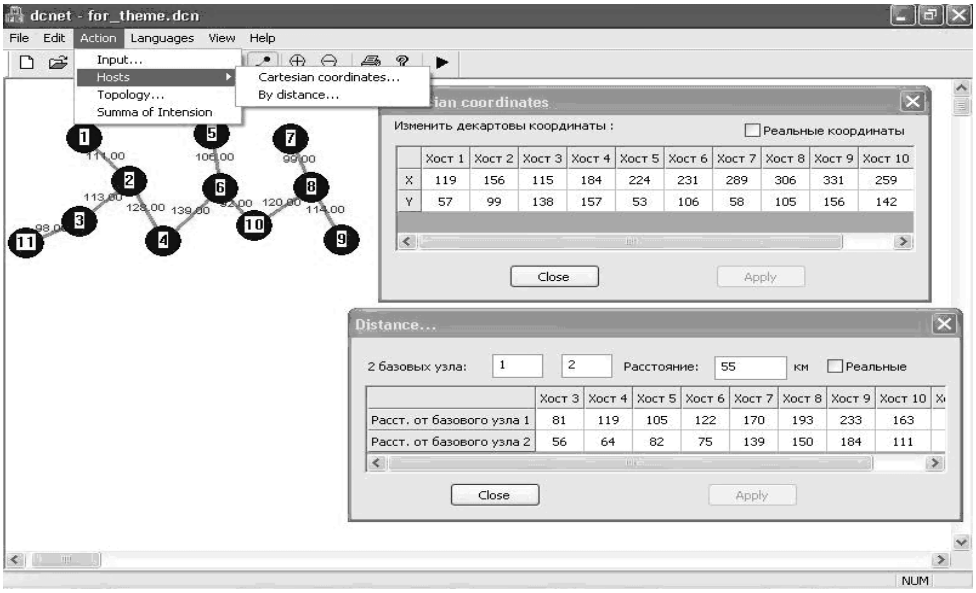


Рис. 3. Аналитическое задание топологии

На рис. 4 показан интерфейс для ввода исходных параметров СПД. В этой закладке можно выбрать одну из двух постановок задачи оптимизации: минимизировать стоимость СПД при ограничении на среднее время задержки пакетов в сети или минимизировать среднее время задержки пакетов в сети при ограничении на стоимость сети. Кроме того, интерфейс позволяет задавать такие параметры, как модель взаимодействия, тип каналов связи, длина пакета, длина запросов и ответов разных типов по прикладным программам, наборам данных и текстовым сообщениям.

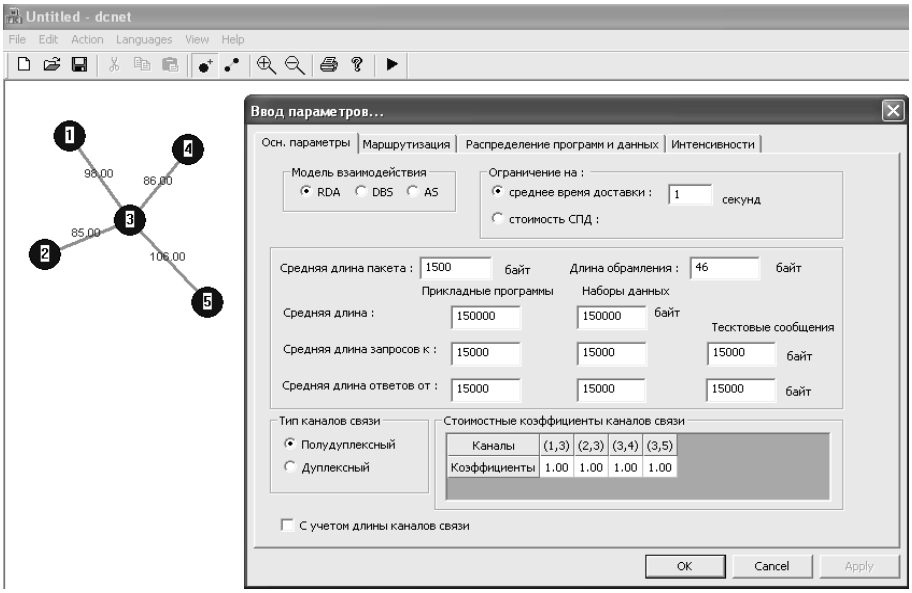


Рис. 4. Интерфейс для ввода исходных параметров СПД

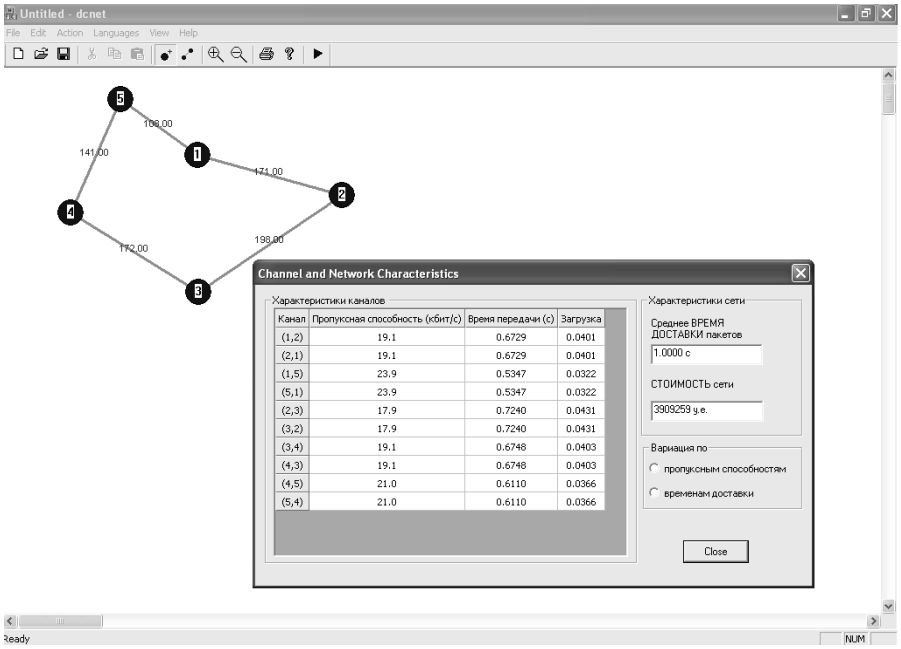
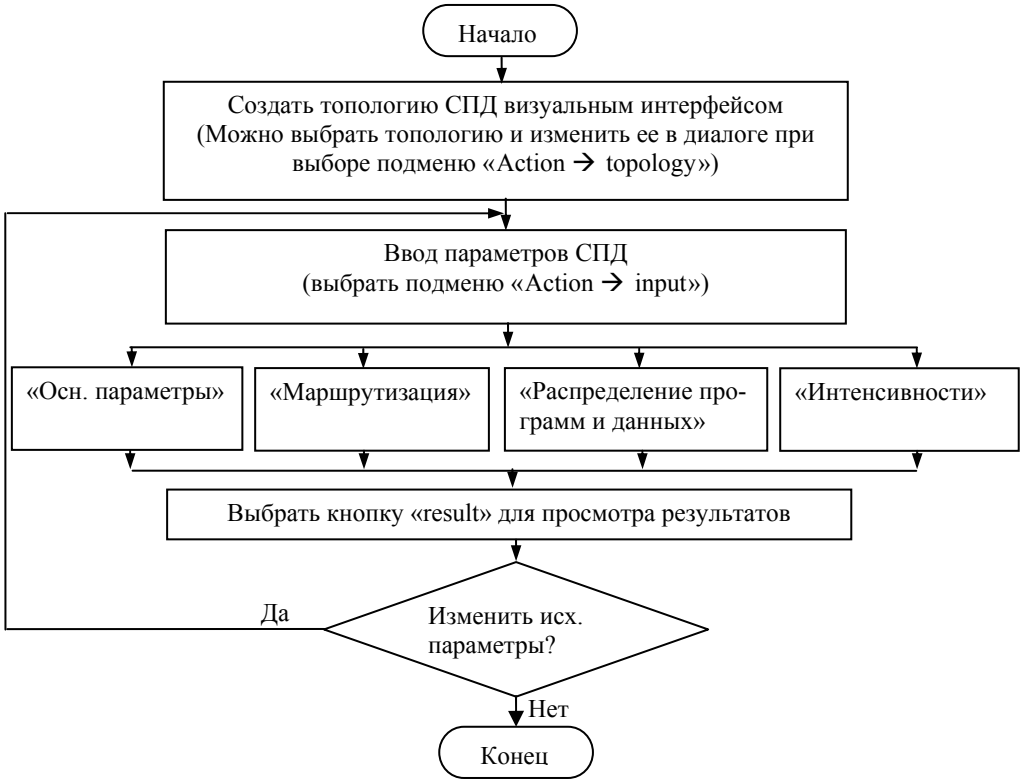


Рис. 5. Пример результатов оптимизации и расчета характеристик СПД

В закладке «Маршрутизация» отображаются созданные автоматически таблицы маршрутизации для каждого узла по критерию «количество хостов» или «взвешенного графа». Здесь же предусмотрена возможность изменения маршрутов и вероятности, в соответствии с которой пакет (или сообщение) пойдет по основному пути. В закладке «Распределение программ и данных» предоставляется возможность ввода варианта распределения прикладных программ и наборов данных по узлам СПД. В закладке «Интенсивности» задаются значения интенсивностей запросов к прикладным программам и наборам данных и текстовых сообщений.

На рис. 5 представлен пример результатов оптимизации и расчета характеристик СПД. Для заданного времени задержки пакетов в сети программа позволяет рассчитать оптимальные пропускные способности каналов связи, обеспечивающие минимальную стоимость СПД, а также время передачи данных по каждому каналу и загрузку всех каналов связи. Здесь же предусмотрено варьирование пропускных способностей и значения времени передачи пакетов в каналах связи.

Работа пользователя с программой реализуется по следующей схеме:



Заключение

Задача параметризации модели и оценки пропускных способностей каналов связи распределенной СПД, в которой расчет потоков пакетов в сети представляет собой основную часть, реализована в виде программного комплекса, позволяющего исследовать и проектировать реальные СПД с использованием модели в виде разомкнутой экспоненциальной СеМО с учетом специфических особенностей реальных СПД, таких как неоднородность потока поступающих в сеть сообщений, многообразие топологий СПД, способов распределения прикладных программ и наборов данных по узлам сети, способов взаимодействия сети, вариантов маршрутизации.

Литература

1. Клейнрок Л. Теория массового обслуживания. М.: Машиностроение, 1979. 432 с.
2. Клейнрок Л. Вычислительные системы с очередями. М.: Мир, 1979. 598 с.
3. Столлингс В. Современные компьютерные сети. 2-е изд. СПб: Питер, 2003. 783 с.
4. Вишневский В.М. Теоретические основы проектирования компьютерных сетей. М.: Техносфера, 2003. 512 с.

ПРОСТРАНСТВЕННО-ЧАСТОТНАЯ ФИЛЬТРАЦИЯ НА ОСНОВЕ МНОГОСЛОЙНОЙ ГОЛОГРАФИЧЕСКОЙ РЕШЕТКИ

В.С. Серебрякова

Научный руководитель – к.т.н., доцент А.Т. Серобабин

В статье рассмотрены вопросы пространственно-частотной фильтрации лазерного излучения на фоне мощной солнечной засветки приемных устройств оптического диапазона в спутниковых системах передачи информации. Предложен вариант пространственно-частотного фильтра на основе голограммы Денисюка. Описан математический аппарат, представлены экспериментальные данные, а также практическая реализация фильтра.

Введение

Традиционные средства связи в отношении их видов, объема, дальности, оперативности и надежности передачи информации непрерывно совершенствуются. Но уже ясно, что требования, предъявляемые к каналам дальней связи, не могут быть полностью удовлетворены наземными средствами проводной, радио- и волоконно-оптической связи. Понятно, что на сегодняшний день для организации глобальной связи, позволяющей передавать и принимать информацию с любой точки Земли, возможно только использование искусственных спутников Земли (ИСЗ) как ретрансляторов большого потока информации. Объем трафика, передаваемый по космическим каналам связи, с каждым годом возрастает. Естественно, что открытые оптические каналы связи никогда не станут альтернативой ВОЛС и в будущем смогут их заменить, однако спутниковые системы являются незаменимыми в тех трудно доступных районах земли, где остальные средства связи просто бессильны.

Практика подтвердила, что использование ИСЗ для связи, в особенности для дальней международной и межконтинентальной, для телевидения и телеуправления, при передаче большого потока информации позволяет устранить многие затруднения. Вот почему спутниковые системы связи (ССС) в короткий срок получили небывало быстрое, широкое и разностороннее применение. Результатом проведенных в последнее десятилетие исследований и разработок явилось создание разнообразной лазерной аппаратуры, которая подтвердила большие перспективы ее использования.

В связи с этим работа является актуальной и направлена на разработку перспективных систем космической связи. Но для обеспечения устойчивого приема оптического излучения в космическом пространстве необходимо добиться превышения полезного сигнала над шумом.

Основным шумовым источником излучения для оптических систем приема информации является Солнце. Распределение энергии, излучаемой Солнцем, охватывает широкий спектральный диапазон Проведенный энергетический расчет открытого лазерного канала связи между высокоорбитальными КА (с параметрами: дальность связи 80663 км, минимальная высота функционирования лазерного канала 16914 км, расходимость лазерного излучения 0,1') показал, что энергетическая плотность на входной апертуре приемной аппаратуры составит $0{,}897 \cdot 10^{3} \cdot P_{пер}$ Вт/м^2, а значение действующей энергетической освещенности при прямой засветке солнечным излучением – $0{,}658 \cdot 10^{15} \cdot P_{пер}$ Вт/м^2. Поэтому для компенсации влияния Солнца на приемные устройства оптического диапазона необходимо проводить как пространственную, так и частотную фильтрацию. В современных оптических системах пространственная фильтрация осуществляется с помощью диафрагмы, а частотная – с помощью призмы, линзы и диафрагмы.

На сегодняшний день существует два основных типа фильтров [1, 2], которые могут служить аналогом для нашего изобретения: косинусный датчик (с рассеивающим элементом) и датчик с теневым полем. В конструктивную схему косинусного фильтра

входят нейтральный фильтр, молочное стекло, темно-красный фильтр, приемник излучения и усилитель. Молочное стекло рассеивает падающее на него излучение и создает на чувствительной площадке приемника излучения освещенность, изменяющуюся пропорционально косинусу угла падения. Темно-красный и нейтральный фильтры предназначены для согласования спектрального диапазона и ослабления потока от Солнца, падающего на приемник излучения. Датчик с теневым полем включает следующие элементы: диафрагма, нейтральный фильтр, темно-красный фильтр, приемник лучистой энергии и усилитель. Принцип действия аналогичен первому варианту. Для фильтрации солнечного излучения чувствительная поверхность приемника затемняется диафрагмой.

С точки зрения угловой чувствительности (основной технико-эксплуатационной характеристики) при работе фильтров в одинаковых условиях с приемниками, имеющими одинаковые характеристики, на основании экспериментов следует, что датчик с теневым полем в несколько раз чувствительней косинусного фильтра. С экономической точки зрения фильтр с теневым полем дешевле косинусного за счет использования диафрагмы, которая в несколько раз дешевле специальный фильтрующих стекол.

Целью данной работы является исследование многослойных голографических решеток в качестве пространственно-частотного фильтра.

Теоретическое исследование возможностей пространственно-частотной фильтрации многослойной голографической решетки

Предположим, что голографический фильтр выполнен по методу Денисюка. Структуру голографического фильтра можно представить как многослойную дифракционную решетку с периодом по осям x – Λ_x; y – Λ_y и по оси z – Λ_z . Для упрощения решаемой задачи будем считать, что рассматривается только разрез по оси x, так как структуру голограммы можно считать симметричной. Исходя из условий эксперимента, входное поле представляет собой плоскую волну; следовательно, на голограмме будет сфокусировано изображение точечного источника, представляющее собой сферическую волну, исходящую из точки, расположенной на поверхности и имеющей координаты x =0, y = 0, z = 0. Его можно описать выражением

$$U(r) = \frac{a}{2}\exp(-jkr) = \frac{a}{2}\exp\left[-k(x^2 + y^2 + z^2)1/2\right], \qquad (1)$$

где r – радиус-вектор; λ – длина волны света, $k = 2\pi/\lambda$ –волновое число; a – действительная амплитуда световой волны.

Для дальнейшего анализа необходимо рассмотреть морфологические особенности голограммы. Она представляет собой слоистую структуру, расположенную по глубине. Приведенное деление условно, обычно описывают большое число слоев, до 10, однако для упрощения математического анализа остановимся на приведенном условном разделении голограммы на три слоя [3]. Таким образом, голограмма представляет собой фазовый объект, показатель преломления которого меняется по закону:

$$n(x, y, z) = n_0 + \Delta n_i(x, y, z), \qquad (2)$$

где n_0 – показатель преломления носителя; $\Delta n_i(x, y, z)$ – изменения показателя преломления, $i = N$ – число слоев. Так как уплотнения в каждом слое располагаются в голограмме по сфере, то $\Delta n_i(x, y, z)$ может быть представлено выражением

$$\Delta n(x, y, z) = \frac{M}{2}\sum \sin\left(\frac{x^2\pi}{\lambda d_i}\right), \qquad (3)$$

где M – индекс глубины модуляции решетки, определяющий максимальное изменение фазы проходящего через него света: d_i – глубина i-го слоя; N – число слоев (рис. 1).

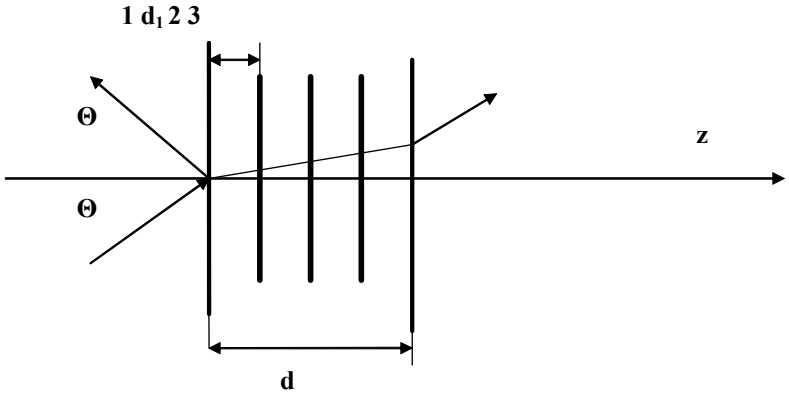


Рис. 1. К расчету дифракции света на голограмме: 1, 2, 3 – зоны раздела сред с различными коэффициентами преломления; d – толщина голограммы; d_1 – толщина первого слоя голограммы; Θ – угол падения и угол отражения света

Амплитудный коэффициент пропускания голограммы с учетом выражения (2) и (3) равен

$$t(x,y)=\exp(jn_0)\prod_{i=1}^{N}\exp\left\{j\frac{M_i}{2}\sin\left[\frac{\pi x^2}{\lambda d_i}\right]\right\}.$$

Используя известное разложение [2]:

$$\exp\{j\sin\varphi\}=J_0(z)+2j\sum_{k=1}^{\infty}J_{2k-1}(z)\sin[(2k-1)\varphi]+2\sum_{k=1}^{\infty}J_{2k}(z)\cos(2k\varphi),$$

и принимая во внимание, что при малых индексах модуляции M амплитуда поля второго порядка дифракции и выше будет мала и их влиянием можно пренебречь, выражение для амплитудного коэффициента пропускания приводится к виду:

$$t(x,y)=e^{jn_0}\prod_{i=0}^{N}\left\{J_0\left(\frac{M_i}{2}\right)+2jJ_1\left(\frac{M_i}{2}\right)\sin\left[\frac{\pi x^2}{\lambda d_i}\right]\right\},$$

где $J_i(.)$ – функция Бесселя первого рода i-го порядка.

При освещении голограммы полем (1) комплексная амплитуда поля в выходной плоскости определяется выражением

$$U_0(x,d)=U(x,z)t(x,z)=\frac{a}{d}\exp\left[-jk\left(x^2+d^2\right)\right]\cdot e^{jn_0}\prod_{i=0}^{N}\left\{J_0\left(\frac{M_i}{2}\right)+2jJ_1\left(\frac{M_i}{2}\right)\sin\left[\frac{\pi x^2}{\lambda d_i}\right]\right\}, \quad (4)$$

Принимая во внимание соотношения [4]

$$\sin\alpha\sin\beta=1/2[\cos(\alpha-\beta)-\cos(\alpha+\beta)];$$

$$\sin z=1/2j[\exp(jz)-\exp(-jz)],$$

и ограничиваясь $N=3$, выражение (4) запишем в виде

$$U_0(x,d)=\frac{a}{d}J_0\left(\frac{M_1}{2}\right)J_0\left(\frac{M_2}{2}\right)J_0\left(\frac{M_3}{2}\right)e^{jn_o}e^{jk\frac{x^2}{2d}}+\frac{a}{d}e^{jn_0}\left\{J_0\left(\frac{M_1}{2}\right)J_1\left(\frac{M_2}{2}\right)J_0\left(\frac{M_3}{2}\right)\right\}\cdot$$

$$\left(\exp\left[jk\frac{x^2}{2}\left(\frac{1}{d}-\frac{1}{d_2}\right)\right]-\exp\left[-jk\frac{x^2}{2}\left\langle\frac{1}{d}-\frac{1}{d_2}\right\rangle\right]\right)+J_0\left(\frac{M_1}{2}\right)J_0\left(\frac{M_2}{2}\right)J_1\left(\frac{M_3}{2}\right)\cdot$$

$$\left\langle\exp\left(jk\frac{x^2}{2}\left(\frac{1}{d}-\frac{1}{d_3}\right)\right)-\exp\left(jk\frac{x^2}{2}\left(\frac{1}{d}-\frac{1}{d_3}\right)\right)\right\rangle+J_1\left(\frac{M_1}{2}\right)J_0\left(\frac{M_2}{2}\right)J_0\left(\frac{M_3}{2}\right)\cdot$$

$$\left\langle\exp\left[jk\frac{x^2}{2}\left(\frac{1}{d}-\frac{1}{d_1}\right)\right]-\exp\left[-jk\frac{x^2}{2}\left(\frac{1}{d}-\frac{1}{d_1}\right)\right]\right\rangle$$

где $d'_i = d_i / \cos\alpha_i$, а α_i – угол дифракции.

Группируя световые волны одного пространственного частотного порядка между собой, интенсивность выходного поля будем описывать выражением

$$I(x,d) = \frac{a}{d} e^{jn_0} \cdot J_0^2\left(\frac{M_1}{2}\right) J_0^2\left(\frac{M_2}{2}\right) J_0^2\left(\frac{M_3}{2}\right) - 4J_0\left(\frac{M_1}{2}\right) J_1\left(\frac{M_1}{2}\right) J_0\left(\frac{M_2}{2}\right) J_1\left(\frac{M_2}{2}\right) J_0^2\left(\frac{M_3}{2}\right) \cdot$$
$$\cos\left[k\frac{x^2}{2}\left(\frac{1}{d_2} - \frac{1}{d_1}\right)\right] - 2J_1^2\left(\frac{M_1}{2}\right) J_1^2\left(\frac{M_2}{2}\right) J_0^2\left(\frac{M_3}{2}\right) \cos\left[k\frac{x^2}{2}\left(\frac{1}{d_2} - \frac{1}{d_1}\right)\right] +$$
$$2J_0^2\left(\frac{M_1}{2}\right) J_1^2\left(\frac{M_2}{2}\right) J_1^2\left(\frac{M_3}{2}\right) \cos\left[k\frac{x^2}{2}\left(\frac{1}{d_3} - \frac{1}{d_1}\right)\right] - \frac{1}{2} J_0\left(\frac{M_1}{2}\right) J_1\left(\frac{M_1}{2}\right) J_0\left(\frac{M_2}{2}\right) \cdot \quad (5)$$
$$J_1\left(\frac{M_2}{2}\right) J_1^2\left(\frac{M_3}{2}\right) \cos\left[k\frac{x^2}{2}\left(\frac{1}{d_2} - \frac{1}{d_1} - \frac{1}{d_3}\right)\right] + \ldots + C$$

где C – коэффициент, характеризующий сумму постоянных составляющих.

Анализ выражения (5) показывает, что интенсивность выходного поля представляет собой сумму постоянных составляющих (постоянной засветки), а также полей, изменяющихся по закону $\cos[kx^2/2\Delta d]$, где $1/\Delta d = 1/\Delta d_i - 1/\Delta d_k$. При смещении источника света на голограмме на величину Δx в выражении для интенсивности поля надо учесть изменение координат источника, например $\cos k/2(x-\Delta x)^2$ $[1/d'_2 - 1/d'_1]$. Определим, на какую величину необходимо произвести смещение источника света, чтобы фаза в центральной точке при $x = 0$ изменилась на величину π. Предположим, в исходном состоянии фаза равнялась 0. Тогда для изменения фазы в центральной точке на π необходимо сместить источник на величину

$$\Delta x = \lambda \frac{d_1 \cos(d_1 \pm \Delta d_1) d_2 \cos(d_2 \pm \Delta d_2)}{d_1 \cos(d_1 \pm \Delta d_2) - d_2 \cos(d_2 \pm \Delta d_2)}.$$

Рассматривая голограмму как оптический, т.е. дифракционный элемент, возникает необходимость определить, какой вид дифракции будет наблюдаться на голограмме. Если параметр $Q = 4kd\sin\Theta\operatorname{tg}\Theta$ (d – толщина голограммы; Θ – угол падения волны) удовлетворяет условию[5]:

$$\pi/12 << Q << 4\pi, \quad (6)$$

то наблюдается как дифракция Рамана-Ната, так и дифракция Брегга. Для голограммы Денисюка условие (6) практически всегда выполняется, и в данном случае присутствует дифракции Брегга, которая имеет обратное распространение электромагнитной световой волны.

Комплексные амплитуды векторов электромагнитных волн в любой точке (кроме границ раздела сред) удовлетворяют уравнениям Максвелла:

$$rotE = j\omega\mu_0 H;$$
$$rotH = j\omega\varepsilon E,$$

где

$$\varepsilon = \begin{cases} \varepsilon_0 \\ \varepsilon_1 = \varepsilon_0\left(1 + \gamma\sum_{i=1}^{N} \sin\left(\frac{\pi x^2}{\lambda d_1}\right)\right), \end{cases}$$

ε_0 – средняя проницаемость среды слоя; γ – индекс модуляции; N – число слоев.

На границе раздела сред электромагнитные составляющие удовлетворяют граничным условиям:

$E_{1x} = E_{2x}$; $E_{1y} = E_{2y}$; $H_{1x} = H_{2x}$; $H_{1y} = H_{2y}$ при $z = 0$;

$E_{2x} = E_{3x}$; $E_{2y} = E_{3y}$; $H_{2x} = H_{3x}$; $H_{2y} = H_{3y}$ при $z = d$,

где цифровой индекс соответствует номеру среды (рис. 1).

Любая падающая волна может быть представлена как сумма плоских волн, каждая из которых имеет $H_y = 0$ (H – волна), $E_z = 0$ (E – волна). Для решения задач дифракции плоской волны на дифракционной решетке необходимо построить решение волнового уравнения:

$$\Delta U = \overline{n^2}[x, y]U = 0, \qquad (7)$$

где $U=E_y$, для H – волны; $U = H_y$ /n для E – волны; $\acute{n}^2 = n^2{}_0$ в случае H – волны и $\acute{n}^2 = \acute{n}^2[x,y]$ в случае – волны.

Рассмотрим дифракцию H – волны, падающей под углом Θ на слой с поперечной модуляцией. Так как по координате слой является однородным, то прошедшая и отраженная волны, так же как и падающая, будут плоскими. Поэтому решение уравнения (7) для трех областей запишется в виде

$$U_1[x, z_1] = B\exp[-jk_1(x\sin\theta + z\cos\theta)] + B\exp[-jk_1(x\sin\theta - z\cos\theta)]; \quad z < 0$$

$$U_3[x_1, z] = D\exp[-jk(x\sin\theta + z\cos\theta)]; \quad z > d$$

$$U_2[x_1, z] = \exp[\pm jk_1 x\sin\theta]\left\{A_0\chi_1\frac{k_1 z}{2} + A_n^{'}\chi_2\frac{k_1 z}{2}\right\}; \quad 0 < z < d,$$

где $\chi_{1,2}$ – линейно независимые решения уравнения Хилла, соответствующие собственному числу,

$$a = \frac{2}{k_1}(\sigma_0 k_c^2 - k_1^2\sin^2\theta); \quad k_1^2 = \omega^2\mu_0\varepsilon_1;$$

$$\sigma_0 = 1 - (\gamma/2)\int\sin(\frac{\pi x^2}{\lambda d_i})dx; \quad k_c^2 = \omega^2\mu_0\varepsilon,$$

μ_0 – магнитная проницаемость среды.

С учетом того, что $E_y[x, y] = U[x, y]$, и граничных условий

$$E_{1y} = E_{2y}; \qquad \frac{\partial E_{1y}}{\partial z} = \frac{\partial E_{2y}}{\partial z} \qquad при \qquad z = 0;$$

$$E_{2y} = E_{3y}; \qquad \frac{\partial E_{2y}}{\partial z} = \frac{\partial E_{3y}}{\partial z} \qquad при \qquad z = d,$$

можно составить систему уравнений для нахождения коэффициентов A, B, D.

Для случая E – волн результат решения остается прежним, отличие заключается лишь в величине коэффициентов при неизвестных A, A/. На рис. 2 представлены численные расчеты дифракции H – волн на синусоидально модулированной диэлектрической решетке для γ = 0,1 и отношении d/d_i. Дифракционная эффективность отражательной решетки η может быть близка к 98 % даже при малых γ, но достаточно большой толщине слоя d/d_i.

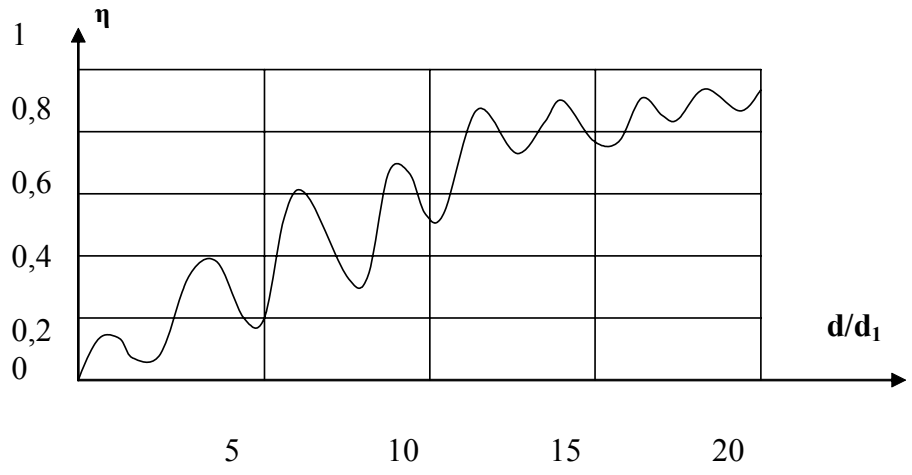


Рис. 2. Зависимость дифракционной эффективности от относительной толщины слоя

Решение волнового уравнения (7) доказывает, что в голограмме существует дифракция Брегга, имеющая обратное направление распространения.

Дифракционная решетка в данном случае выполняет не только пространственную фильтрацию, но и частотную. Зарегистрированная голограмма на одной световой волне образует многослойную дифракционную решетку. Если освещать эту решетку белым светом под тем же углом, при котором велась запись голограммы (солнечный или отраженный свет совпадает с направлением лазерного излучения – наиболее неблагоприятный вариант), то от первой прослойки отразится небольшое количество света, большая часть его проникает дальше, отразится частично от второй, третьей и т.д. прослоек. Разность хода лучей между всеми отраженными от разных прослоек пучками будет равна двойному расстоянию между прослойками; она равна λ_1, для той области, где прослойки разделены расстоянием 1/2 λ_1. Интерферируя между собой, пучки, отраженные от голограммы дадут максимум для света с длинной волны λ_1. Для всякой другой длины волны (λ) найдется такое число слоев m, которое даст разность хода, равную нечетному кратному полуволны 1/2 λ. Соответствующее m определится из условия $m\lambda_1 = 1/2(2p + 1)\ \lambda$.. Таким образом, луч с длинной волны λ , отраженный от первого слоя, будет ослаблен лучом, отраженным от (m + 1)-го слоя, луч отраженный от второго слоя, нейтрализуется лучом, отраженным от (m + 2)-го слоя, и т.д. Следовательно, в отраженном свете цвет с длиной волны λ будет более или менее исключен. Голограмма, изготовленная по методу Денисюка, выполнят роль частотного фильтра, пропуская только волну с длиной записи голограммы, остальные составляющие частотного спектра или будут отсутствовать, или их влияние будет существенно снижено.

Экспериментальная проверка пространственной и частотной фильтрации голографического фильтра

Для обеспечения устойчивого приема оптического излучения в космическом пространстве необходимо добиться превышения полезного сигнала над шумом. Выполнение данного условия возможно при пространственной и частотной фильтрации, используя свойства когерентного излучения и дифракции света на голографических пространственно-частотных фильтрах.

Опытная проверка свойств голограммы, выполненной по методу Денисюка, проводилась на установке, представленной на рис. 3.

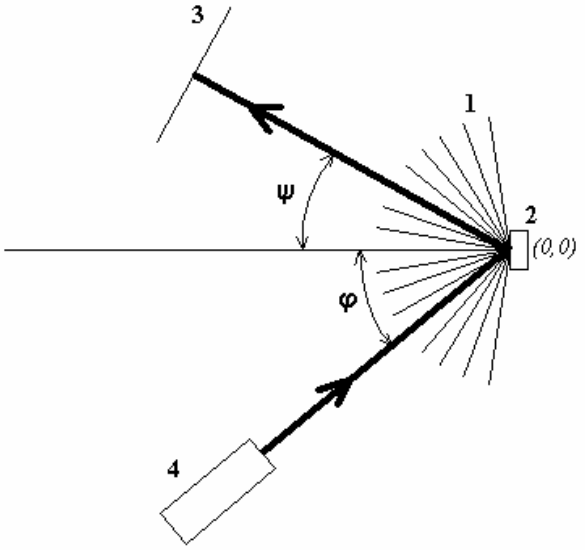


Рис. 3. Экспериментальная установка

Схема включает:

1. измерительное поле, представляющее собой горизонтальную поверхность (ватман формата А1), расчерченную на секторы по 10° относительно центральной точки с координатами (0,0);
2. голографический фильтр, изготовленный по методу Денисюка, размером 3×3 см, установленный неподвижно в точке начала координат, перпендикулярно полю (1);
3. непрозрачный белый экран, установленный также перпендикулярно полю (1) и свободно передвигающийся в плоскости постоянно перпендикулярно сектору с углом ±30° относительно точки (0,0);
4. лазер, излучающий на длине волны λ=0,63 мкм, в исходном положении находящийся перпендикулярно голограмме (2) по оси на координатной плоскости.

Лазер (4) работает в непрерывном режиме и излучает постоянную длину волны λ. Голограмма (2) является отражательной, при облучении ее когерентным светом от источника на экране (3) наблюдается дифракционная картина с ярко выраженным основным максимумом в ее центре. Лазер перемещается по секторам с одинаковым шагом в 10° на расстоянии 20 см относительно точки начала координат по кругу. Плоскость падения лазерного излучения параллельна измерительному полю (1); луч лазера направлен вдоль расчерченных секторов. Обход осуществляется в положительном φ+ (против часовой стрелки) и в отрицательном φ- (по часовой стрелке) направлениях. В зависимости от направления освещения голограммы осуществляется перемещение экрана вдоль линии, перпендикулярной сектору в 30°. Таким образом, регистрируется зависимость изменения угла отражения ψ(φ) от угла падения φ. Положение нулевого максимума (самый яркий точка, максимум дифракционной картины, помимо которого наблюдались ±1,±2,±3 порядки дифракции со значительно меньшей интенсивностью) проецировалось с экрана (3) на координатную плоскость (1) и фиксировались значения ψ+ или ψ-.

По полученным экспериментальным данным был построен ряд зависимостей ψ(φ) и проведена аппроксимация данных графиков по методу наименьших квадратов (рис. 4) в mathcad.

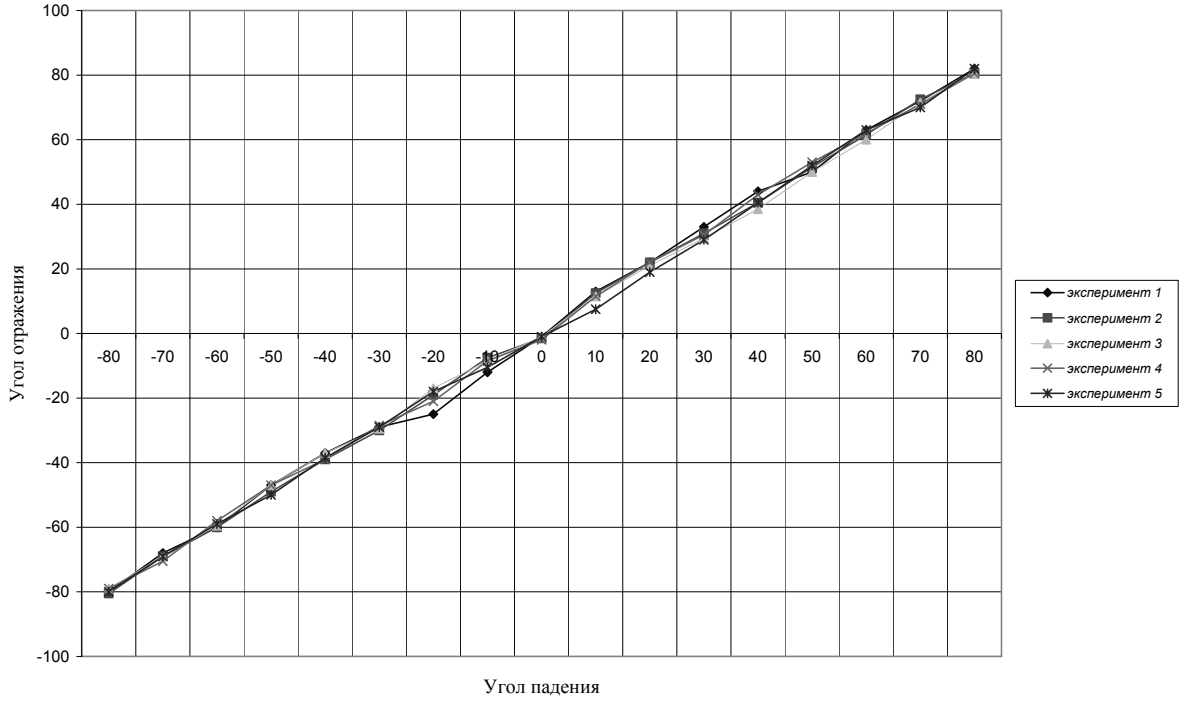


Рис. 4. Зависимость ψ(φ)

Чтобы оценить дифракционную эффективность отражательной решетки, голограмму осветили одновременно белым светом с набором длин волн λi и лазерным излучением с длиной волны λ=0,63 мкм, при этом направление белого света совпадает с направлением луча лазера. В результате опыта на экране (3) наблюдалась картина разложения в спектр белого света в виде радужного изображения и максимума излучения лазера в многократно ослабленном белом пятне света (рис. 5). Причем дифракционный максимум отраженного луча лазера был значительно разнесен с разложенным в спектр белым светом.

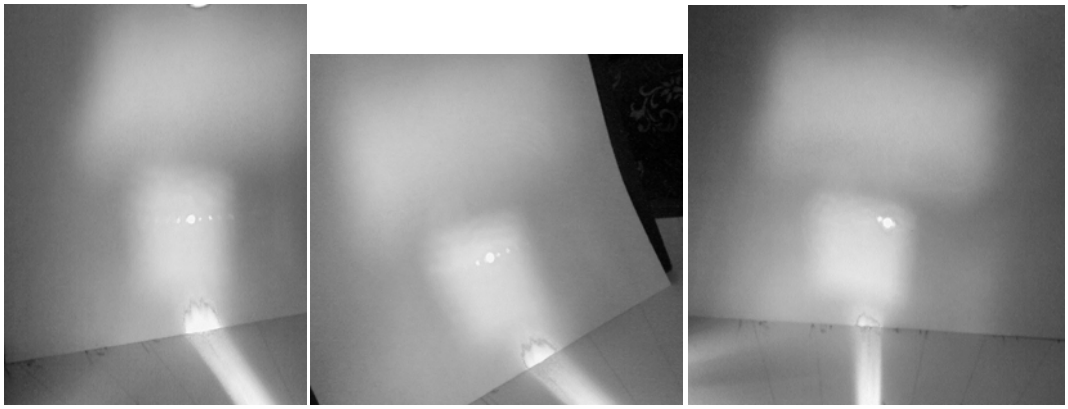


Рис. 5. Фотографии эксперимента

Исходя из проведенного эксперимента, приемник оптического излучения может быть построен по принципиальной схеме, изображенной на рис. 6.

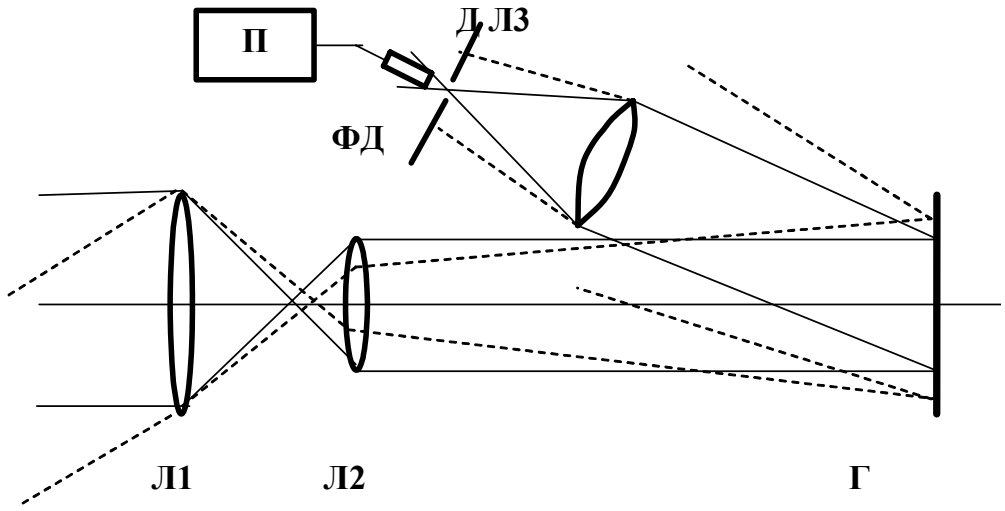


Рис. 6. Приемник оптических сигналов с голографической пространственно-частотной фильтрацией оптического излучения

На рис. 6: Л1, Л2 – входной объектив оптического приемника; Г – голограмма, пространственно-частотный фильтр; Л3 – фокусирующая линза; Д – диафрагма; ФД – фотодетектор; П – приемник электрических сигналов. Сплошными линиями показан ход лучей когерентного излучения, а пунктирными линиями – ход лучей излучения Солнца.

Таким образом, была осуществлена пространственная и частотная фильтрация оптического излучения в лабораторных условиях. Аналогичным образом может происходить фильтрация полезного информационного сигнала при засветке оптического приемника солнечным излучением на стационарных космических объектах.

Заключение

Проведенные теоретические и экспериментальные исследования показали, что многослойные голографические решетки могут выполнять функции пространственной и частотной фильтрации. При толщине оптического слоя больше десяти дифракционная эффективность голограммы превышает 90 %, тем самым обеспечивается минимальная потеря энергии когерентного излучения и максимальная потеря энергии не когерентного излучения. Использование фиксированного угла записи при формировании голограммы обеспечивает линейную характеристику угла дифракции, что позволяет успешно производить пространственную фильтрацию, приходящего излучения, разнося в пространстве когерентное излучение и излучение от Солнца, тем самым, обеспечивая максимальное отношение сигнал/шум на входе фотодетектора.

Литература

1. Ивандиков Я.М. Оптико-электронные приборы для ориентации космических аппаратов. М.: Машиностроение, 1979.
2. Солозобов М.Е. Моделирование процессов оптимальной фильтрации. // Научн. вест. МГТУ ГА. 2005. № 93.
3. Сороко Л.М. Основы голографии и когерентной оптики. М.: Наука, 1971.
4. Корн Г, Корн Т. Справочник по математике для научных работников и инженеров. – М.: Наука, 1978.
5. Кузнецов Д.С. Специальные функции. М.: Высшая школа, 1965.

ПРИМЕНЕНИЕ ГЕНЕТИЧЕСКОГО ПРОГРАММИРОВАНИЯ ДЛЯ РЕАЛИЗАЦИИ СИСТЕМ СО СЛОЖНЫМ ПОВЕДЕНИЕМ

Н.И. Поликарпова, В.Н. Точилин
Научный руководитель – д.т.н., профессор А.А. Шалыто

Данная работа рассматривает применение методов генетического программирования для построения систем со сложным поведением. Предложено представление таких систем в форме хромосомы. Адаптированы и усовершенствованы операторы генетического программирования. Предложено решение проблемы экспоненциального роста объема хромосомы и размерности пространства поиска с увеличением количества параллельных входов автомата. Рассмотрен пример.

Введение

Программные и аппаратные системы, а также их отдельные элементы часто имеют *сложное поведение*. Таким свойством обладает большинство реактивных систем, устройства управления, сетевые протоколы, диалоговые окна, персонажи компьютерных игр и многие другие объекты и системы.

В последнее время для описания сущностей со сложным поведением предлагается использовать *автоматный подход* [1, 2]. В соответствии с этим подходом сущность представляется в виде *автоматизированного объекта* – совокупности *управляющего автомата* и *объекта управления*. Объект управления характеризуется множеством *вычислительных состояний*, а также двумя наборами функций: множеством *предикатов*, отображающих вычислительное состояние в логическое значение (истина или ложь), и множеством *действий*, позволяющих изменять вычислительное состояние. Управляющий автомат определяется конечным множеством *управляющих состояний*, *функцией переходов* и *функцией действий*. При этом объект управления может быть легко реализован традиционными методами (как программно, так и аппаратно), так как его поведение уже не является сложным: предикаты и действия представляют собой простые, короткие функции, практически не содержащие ветвлений. Вся «логика» оказывается сосредоточенной в управляющем автомате.

Описание логики в форме диаграммы переходов конечного автомата хорошо структурировано, разработан ряд подходов к его декомпозиции [1, 3]. Несмотря на это, опыт показывает, что построение управляющего автомата для сложной сущности или системы обычно требует от разработчика гораздо больше усилий и порождает больше ошибок, чем реализация объекта управления. Эту проблему авторы настоящей работы предлагают решать методами *генетического программирования* [4].

Основная идея генетического программирования состоит в построении программ путем применения генетических алгоритмов к некоторой *модели вычисления*. При этом разработчику программы остается лишь задать *оценочную функцию*, определяющую для каждого возможного результата вычисления в выбранной модели численное значение, называемое его *пригодностью* (*fitness*). Таким образом, развитие генетического программирования – важный шаг в направлении повышения уровня абстракции языков программирования.

В качестве моделей вычисления в генетическом программировании чаще всего используют деревья, графы, команды процессора – «низкоуровневые» модели, имеющие ограниченный набор элементарных операций (таких как, например, запись и чтение ячеек памяти, арифметические операции, вызовы подпрограмм и т.д.). Достоинство низкоуровневых моделей состоит в их универсальности: с их помощью можно построить любую программу целиком, единообразно, вне зависимости от специфики решаемой задачи. Однако, такие модели обладают и серьезными недостатками. Во-первых, построенная программа из-за отсутствия высокоуровневой структуры редко бывает понятна человеку, что исключает возможность ее дальнейшей модификации вручную,

обобщения полученного решения и т.п. Во-вторых, из-за того, что пространство допустимых программ в этом случае очень велико, генетическая оптимизация может потребовать длительного времени.

В настоящей работе предлагается объединить автоматный и генетический подходы. При этом в качестве оптимизируемой модели вычисления выступает автоматизированный объект. Реализацию объекта управления предлагается производить вручную, а после этого применять генетический алгоритм для автоматического построения управляющего автомата. Автомат – «высокоуровневый» вычислитель: его элементарные операции – предикаты и действия, специфичные для конкретной задачи. Именно поэтому предлагаемый подход лишен вышеупомянутых недостатков, характерных для низкоуровневого генетического программирования. Здесь машине поручается именно та часть работы по написанию программ, которая хуже всего получается у человека.

1. Обзор смежных работ

Эволюционной оптимизации моделей вычислений в виде конечных автоматов были посвящены многие исследования в различных направлениях *эволюционных вычислений*. Классификация по направлениям, приведенная на Рис. 1, сделает их рассмотрение более удобным.

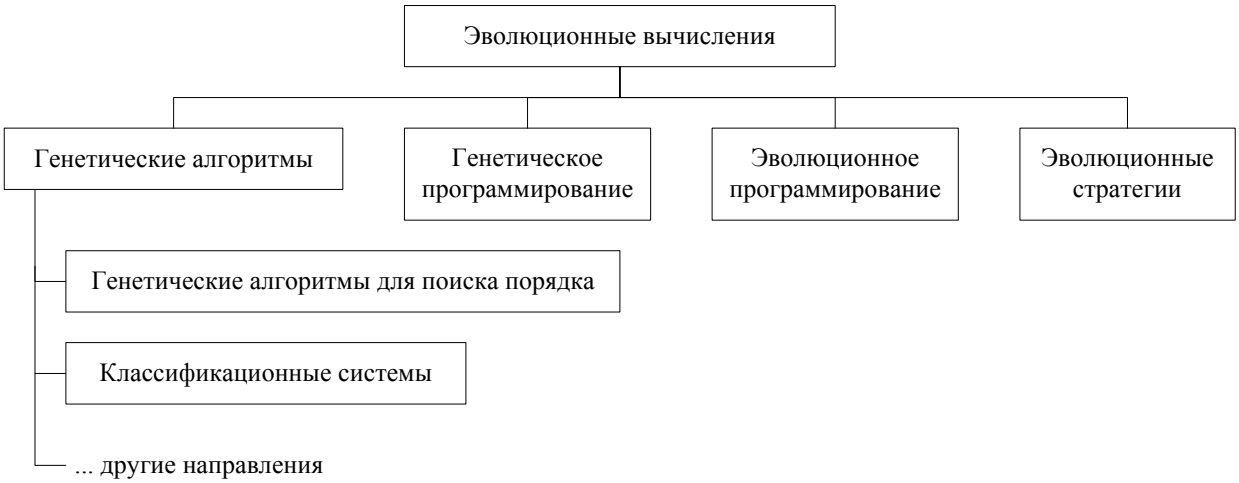


Рис. 1. Направления эволюционных вычислений

Генетические алгоритмы рассматривают методы функциональной оптимизации, основанные на модели естественной эволюции.

Классификационные системы работают преимущественно с пространством порождающих правил, а также конечных автоматов-*распознавателей*, определяющих принадлежность строки некоторому языку. Распознаватель не производит выходных воздействий, результат определяется состоянием автомата после обработки входной последовательности. В данном направлении как наиболее значимые можно выделить работы [5–12]. Более сложная форма конечного автомата – *преобразователь* (*transducer*) – отображает множество входных строк на множество выходных, возможно над другим алфавитом. Эволюционному построению преобразователей посвящена работа [13].

Генетическое программирование было впервые упомянуто в книге [4]. Первоначально предлагалось представлять программу в виде дерева. В работе [14] рассматривалось представление в виде машинного кода, а в статье [15] впервые оптимизировалась модель в форме графа, который можно интерпретировать как диаграмму переходов конечного автомата. Модель оказалась достаточно удачной, и впоследствии использовалась практически без изменений в ряде работ [16–19]. В статьях [20–22] рассматривается совместное использование различных моделей вычислений, включая гра-

фы или их модификации. В работах [23, 24] в качестве модели вычисления используются клеточные автоматы. В работе [25] применяются ациклические графы для формирования эффективно распараллеливаемых программ. В статьях [26–28] рассматривается автоматическое построение компонент логических контроллеров в виде автоматов, а в работе [29] оценивается эффективность автоматных моделей применительно к различным задачам.

В работе [30] отмечается необходимость расширения генетического программирования для работы со сложными структурами данных и описываются достижения в данном направлении. Эта тема развивается в работах [15, 31, 32], где рассмотрено использование различных заданных структур данных. Отметим, что подход, предлагаемый в настоящей работе, решает эту задачу полностью, позволяя использовать в составе автоматизированного объекта любой объект управления, включая произвольные стандартные структуры данных и их комбинации.

В разделе *эволюционное программирование* сначала изучались методы создания искусственного интеллекта в форме эволюционирующей популяции автоматов, а впоследствии основной его задачей стала оптимизация числовых параметров. Характерной особенностью данного направления можно назвать отсутствие скрещивания, при этом оптимизация осуществляется только за счет мутации. В работах этого направления [33–42] автоматы обучаются предсказывать следующий символ входной последовательности, классифицировать последовательности и копировать поведение системы, заданной частичным набором входных и соответствующих им выходных воздействий. Задачи управления рассматриваются как обучение автомата повторению поведения системы и его анализу с целью выявления входных воздействий, приводящих к требуемому результату.

В работе [43], а также в ряде работ, не относящихся напрямую к эволюционному программированию, [44–46], автоматы находят применение в играх. Авторы работы [47] рассматривают адаптивные методы эволюционного программирования. В статьях [48, 49] воссоздание неизвестных систем в виде автоматов применяется для изучения поведения агентов с целью взаимодействия с ними и эффективного тестирования последовательных электрических схем, соответственно.

Эволюционные стратегии позволяют эффективно оптимизировать параметры системы. В рамках этого направления часто рассматриваются аппаратные системы, для которых характерна экспериментальная проверка пригодности (в процессе реальной работы или ее программной эмуляции), что делает количество таких проверок решающим для производительности.

Практически во всех рассмотренных исследованиях автомат в каждый момент времени обрабатывает только одну входную переменную. Исключения составляют лишь работы [45] (четыре параллельных троичных входа) и [29] (в качестве условия перехода допускается сравнение значений двух регистров). Теоретически, любое количество параллельных входов сводится к одному, в качестве воздействий которого выступают комбинации сигналов исходных параллельных входов. Однако размер алфавита полученного таким образом входа растет экспоненциально с увеличением количества исходных параллельных входов. В обоих упомянутых работах параллельные входы не приводят к недопустимо большому алфавиту, но для реальных систем эта проблема крайне актуальна.

Также в рассмотренных работах автомат на каждом шаге может выполнить не более одного действия из заданного множества. В таком случае любая комбинация действий, которые может потребоваться выполнить одновременно, также должна считаться элементарным действием. При этом требуется априорная информация обо всех возможных комбинациях действий, либо задание вместе с элементарными действиями всех их наборов, что приводит к экспоненциальному росту количества действий. Отме-

тим, что в работе [29] допускаются действия с аргументами, а также параллельно выполняемые автоматы, отвечающие за различные действия, что значительно ослабляет проблему.

Из всех перечисленных работ наилучшие результаты в аспекте автоматического построения части программы на высоком уровне абстракции получены в статьях [27, 28] применительно к созданию управляющей программы робота. Однако и эти работы не лишены упомянутых выше недостатков, относящихся к входным и выходным воздействиям. Насколько известно авторам, исследования в области эволюционного построения систем со сложным поведением, отличных от роботов, ранее не проводились.

2. Постановка задачи

Цель настоящей работы – проверить возможность эффективного применения генетических алгоритмов для построения логики автоматизированных объектов., для которых характерно наличие параллельных входов и выходов.

Сформулируем *задачу построения управляющего автомата* более формально. Пусть задан объект управления $O = \langle V, v_0, X, Z \rangle$, где V – множество вычислительных состояний (или *значений*), v_0 – *начальное значение*, $X = \{x_i : V \to \{0,1\}\}_{i=1}^{n}$ – множество предикатов, $Z = \{z_i : V \to V\}_{i=1}^{m}$ – множество действий. Также задана оценочная функция $\varphi : V \to \mathbf{R}^+$.

Объект O может управляться автоматом вида $A = \langle S, s_0, \Delta \rangle$, где S – конечное множество управляющих состояний, s_0 – *стартовое состояние*, $\Delta : S \times \{0,1\}^n \to S \times Z^*$ – *управляющая функция*. Управляющую функцию можно разложить на две компоненты: функцию действий $\zeta : S \times \{0,1\}^n \to Z^*$ и функцию переходов $\delta : S \times \{0,1\}^n \to S$.

Пусть объекту управления соответствует значение v, а управляющий автомат находится в состоянии s. В течении одного *шага* работы автоматизированного объекта автомат переходит в новое состояние $s_{new} = \delta(s, x_1(v), \ldots, x_n(v))$, а объект управления изменяет свое значение на $v_{new} = z^l(z^{l-1}(\ldots(z^1(v))\ldots))$, где $(z^1, z^2, \ldots z^l) = \zeta(s, x_1(v), \ldots, x_n(v))$.

Задача построения управляющего автомата состоит в том, чтобы найти автомат заданного вида такой, что за k шагов работы под управлением этого автомата объект O перейдет в вычислительное состояние с максимальной пригодностью ($\varphi(v) \to \max$).

Отметим, что здесь для простоты используется частный случай модели автоматизированного объекта.

В настоящей работе для решения поставленной задачи предлагается использовать генетическое программирование. В связи с этим возникают следующие задачи: выбор представления конечного автомата в виде *особи*; адаптация генетических операторов (*мутации* и *скрещивания*) для выбранного представления; выбор параметров генетической оптимизации, подходящих для автоматов.

3. Способы представления автомата

В классической интерпретации генетического алгоритма особь представляется в виде набора *хромосом*, каждая из которых записана как битовая строка. При этом генетические операторы определяются универсальным образом в терминах битовых строк. Управляющий автомат легко представить как набор состояний, в каждом из которых его поведение определяется сужением управляющей функции $\Delta_s : \{0,1\}^n \to S \times Z^*$,

$s \in S$. Таким образом, удобно сопоставить каждому состоянию хромосому. Однако, низкоуровневая запись хромосомы в виде битовой строки не очень удобна.

В генетическом программировании распространен другой подход: для каждого класса оптимизируемых моделей вычисления выбирается свое представление, обладающее высокоуровневой структурой. Определение генетических операторов в терминах этой структуры с учетом семантики хромосомы позволяет значительно ускорить процесс эволюции.

В данном разделе авторы предлагают два варианта представления состояния в виде хромосомы.

3.1. Представление состояний: полные таблицы

Естественный способ записи хромосомы состояния – это табличное представление функции Δ_s. Таблица содержит 2^n строк (по одной для каждой возможной комбинации значений предикатов) и $m+1$ столбцов, в первом из которых записано значение функции переходов (номер целевого состояния), а совокупность остальных столбцов обозначает множество действий, которые необходимо выполнить на переходе. Пример полной таблицы одного состояния приведен на Рис. 2 (контуром обведена информативная часть таблицы, именно она и заносится в хромосому). Отметим, что в каждой строке таблицы записано множество действий, а не их последовательность, как это было определено в модели (разд. 2). Задание значения функции действий в виде множества значительно упрощает оператор скрещивания и повышает эффективность процесса эволюции. Выполнение на переходе последовательности действий эквивалентно осуществлению нескольких переходов, на которых выполняются множества действий. Таким образом, автомат исходной модели всегда может быть записан в предложенной выше табличной форме, возможно с добавлением нескольких состояний и переходов. После получения результата оптимизации лишние элементы автомата можно устранить, преобразовав множества действий в последовательности.

x_0	x_1	s	z_0	z_1	z_2
0	0	0	0	0	1
0	1	1	1	0	1
1	0	0	0	1	0
1	1	2	1	1	1

Рис. 2. Хромосома состояния: полная таблица (n = 2, m = 3, $|S|$ = 3)

Все таблицы, соответствующие состояниям одного автомата, имеют одинаковую размерность, так как число предикатов и действий объекта управления задано по условию задачи. Что касается управляющих состояний, наиболее эффективным является постепенное наращивание их числа в процессе оптимизации.

Опишем теперь генетические операторы над хромосомами состояний, записанными предложенным выше способом (в виде полных таблиц).

Алгоритм 1. Мутация полных таблиц. Алгоритм мутации состояния, представленного полной таблицей, описан на псевдокоде в листинге 1 и проиллюстрирован на Рис. 3 (здесь и далее на стрелках, обозначающих изменения значений в ячейках таблицы, написаны вероятности этих изменений). При мутации состояния с некоторой вероятностью может мутировать каждый элемент таблицы. При этом номер целевого состояния изменяется на любой из допустимых, а вместо исходного набора действий ге-

нерируется новый набор, вероятность появления единиц в котором равна доле единиц в исходном наборе.

Листинг 1. Мутация полных таблиц

```
State Mutate(State state)
{
	State mutant = state;
	for (для всех i: строк таблицы)
	{
		if (с вероятностью p1) {
			mutant[i].targetState = случайное число от 0 до nStates-1;
		}
		if (с вероятностью p2) {
			int nActsPresent = количество единиц в mutant[i].output;
			if ((nActsPresent == 0) || (nActsPresent == nActions)) {
				Index j = случайное число от 0 до nActions - 1;
				mutant[i].output[j] = !mutant[i].output[j];
			} else {
				for (для всех j: номеров действий) {
					mutant[i].output[j] = 1 с вероятностью
						nActsPresent/nActions и 0 иначе;
				}
			}
		}
	}
	return mutant;
}
```

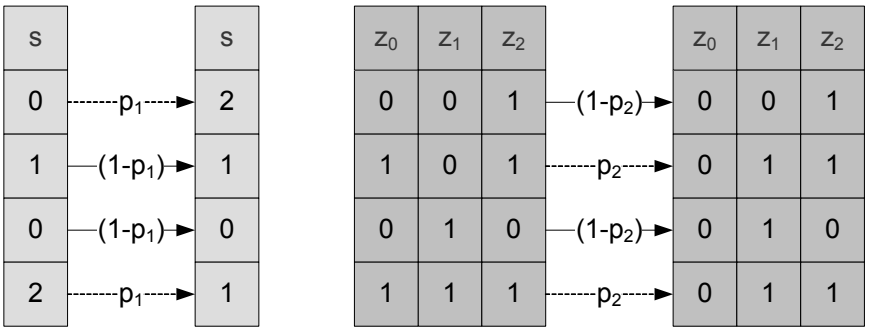


Рис. 3. Пример мутации полных таблиц

Алгоритм 2. Скрещивание полных таблиц. В настоящей работе рассматривается адаптация к предложенному представлению состояний одного способа скрещивания, известного как *одноточечное*. Руководствуясь схожими идеями, нетрудно адаптировать к табличному представлению и другие способы скрещивания. Алгоритм одноточечного скрещивания полных таблиц представлен в листинге 2 и проиллюстрирован на Рис. 4.

Листинг 2. Скрещивание полных таблиц

```
pair<State, State> Cross(State state1, State state2)
{
	State child1 = state1;
	State child2 = child1;
	int tableSize = размер таблицы;

	for (для всех j: столбцов таблицы)
	{
		int crossPoint = случайное число от 0 до tableSize;
		for (для всех i: строк таблицы от 0 до crossPoint - 1) {
			child1[i][j] = state1[i][j];
			child2[i][j] = state2[i][j];
```

```
            }
            for (для всех i: строк таблицы от crossPoint до tableSize - 1)
            {
                  child1[i][j] = state2[i][j];
                  child2[i][j] = state1[i][j];
            }
      }
      return make_pair(child1, child2);
}
```

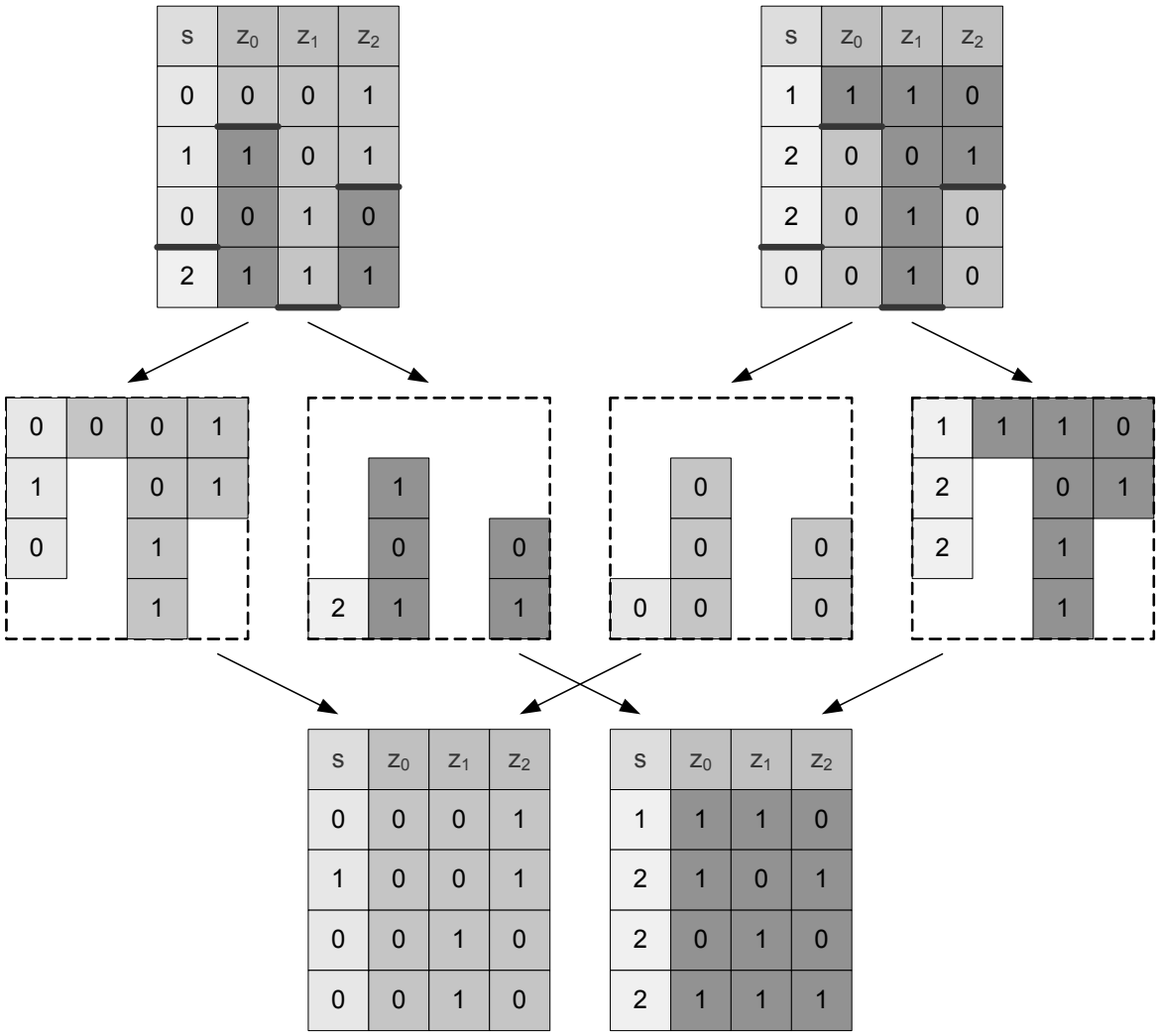


Рис. 4. Пример скрещивания полных таблиц

Основная проблема, возникающая при использовании полных таблиц рассмотренного вида – это экспоненциальный рост размерности хромосомы с увеличением числа предикатов объекта управления (напомним, что количество строк в таблице 2^n, где n – число предикатов). Опыт показывает, что в реальных задачах управляющие автоматы, построенные вручную, имеют гораздо меньше переходов, чем $|S| \cdot 2^n$. Причина, по мнению авторов, в том, что в большинстве задач предикаты имеют «локальную природу» по отношению к управляющим состояниям. В каждом состоянии *значимым* является лишь определенный, небольшой поднабор предикатов, остальные же не влияют на значение управляющей функции. Именно это свойство позволяет существенно сократить размер описания состояний. Кроме того, навязывание этого свойства в процессе оптимизации позволяет получить результат, более похожий на автомат, построенный вручную, а значит, более понятный человеку.

3.2. Представление состояний: сокращенные таблицы

Свойство локальности предикатов можно использовать для сокращения описания управляющего состояния разными способами. Авторами выбран один из подходов, при котором количество значимых в состоянии предикатов ограничивается некоторой константой r. К таблице, задающей сужение управляющей функции на данное состояние, в этом случае добавляется битовый вектор, описывающий множество значимых предикатов (Рис. 5).

x_0	x_1	x_2	x_3	x_4	x_5
0	1	0	1	0	0

x_1	x_3	s	z_0	z_1	z_2
0	0	0	0	0	1
0	1	1	1	0	1
1	0	0	0	1	0
1	1	2	1	1	1

Рис. 5. Хромосома состояния: сокращенная таблица (n=6, m=3, r=2, $|S|$=3)

Число строк таблицы в этом случае 2^r, однако, константа r обычно невелика. Ее выбор зависит от сложности задачи. Как показывает опыт, для большинства автоматизированных объектов среднее по всем состояниям значение r не больше пяти.

Опишем генетические операторы над хромосомами состояний, записанными в виде сокращенных таблиц.

Алгоритм 3. Мутация сокращенных таблиц. По сравнению с представлением в виде полной таблицы, добавилась возможность мутации множества значимых предикатов. При этом каждый из значимых предикатов с некоторой вероятностью заменяется другим, не принадлежащим множеству (Рис. 6).

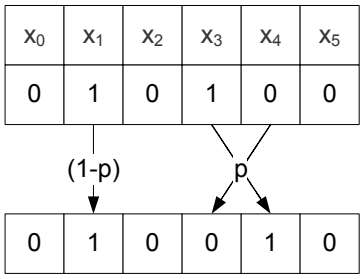


Рис. 6. Пример мутации множества значимых предикатов

Мутация самой сокращенной таблицы происходит так же, как мутация полной таблицы. Описание алгоритма приведено в листинге 3.

Листинг 3. Мутация сокращенных таблиц

```
State Mutate(State state)
{
      State mutant = state;
      if (с вероятностью p) {
            int from, to;
            случайно выбрать from и to, так что
                  (mutant.predicates[from]==1)&&
                  (mutant.predicates[to]==0);
            mutant.predicates[from] = 0;
            mutant.predicates[to] = 1;
```

```
        }
        // Остальная часть хромосомы мутирует, как в случае полных таблиц
        mutant.table = Mutate(mutant.table);
        return mutant;
}
```

Алгоритм 4. Скрещивание сокращенных таблиц. Это наиболее сложный из предлагаемых алгоритмов. Основная последовательность его шагов отражена в листинге 4.

Листинг 4. Скрещивание сокращенных таблиц

```
pair<State, State> Cross(State state1, State state2)
{
        State child1 = state1;
        State child2 = child1;

        ChoosePreds(state1.predicates, state2.predicates,
                child1.predicates, child2.predicates);

        int crossPoint = случайное число от 0 до tableSize;
        FillChildTable(state1, state2, child1, crossPoint);
        FillChildTable(state1, state2, child2, crossPoint);
        return make_pair(child1, child2);
}
```

Поскольку родительские хромосомы, представленные сокращенными таблицами, могут иметь разные множества значимых предикатов, сначала необходимо выбрать, какие из этих предикатов будут значимы для хромосом детей. Функция `ChoosePreds`, осуществляющая этот выбор, представлена в листинге 5.

Листинг 5. Выбор значимых предикатов детей при скрещивании сокращенных таблиц

```
void ChoosePreds(Predicates p1, Predicates p2, Predicates ch1, Predicates
ch2)
{
        for (для всех i: номеров предикатов) {
                if (p1[i] && p2[i]) {          // Предикат от обоих родителей
                        ch1[i] = ch2[i] = true; // достается обоим детям
                        запоминаем, что в наборах предикатов детей стало
                        на 1 меньше места;
                }
        }
        for (для всех i: номеров предикатов) {
                if (p1[i] != p2[i]) {
                        Predicates* pCh;
                        if (у обоих детей есть место) {
                                pCh = равновероятно любой ребенок;
                        } else {
                                pCh = тот ребенок, у которого еще есть место;
                        }
                        (*pCh)[i] = true;
                        запоминаем, у кого стало меньше места;
                }
        }
}
```

Работа функции `ChoosePreds` для родительских хромосом, представленных на Рис. 7, проиллюстрирована на Рис. 8.

x_0	x_1	x_2	x_3	x_4	x_5
0	1	0	1	0	0

x_0	x_1	x_2	x_3	x_4	x_5
1	1	0	0	0	0

x_1	x_3	s	z_0	z_1	z_2
0	0	0	0	0	1
0	1	1	1	0	1
1	0	0	0	1	0
1	1	2	1	1	1

x_0	x_1	s	z_0	z_1	z_2
0	0	1	1	1	0
0	1	2	0	0	1
1	0	2	0	1	0
1	1	0	0	1	0

Рис. 7. Родительские хромосомы, представленные сокращенными таблицами

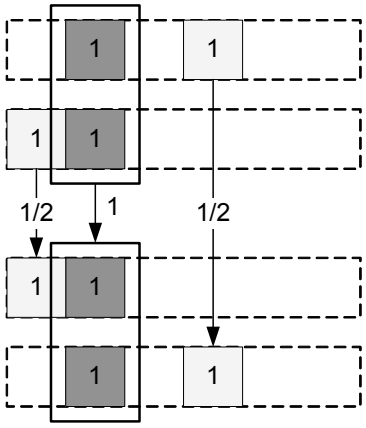


Рис. 8. Пример выбора значимых предикатов детей

После выбора значимых предикатов заполняются таблицы обоих детей. Алгоритм заполнения представлен в листинге 6.

Листинг 6. Заполнение таблиц детей при скрещивании сокращенных таблиц

```
void FillChildTable(State s1, State s2, State& child, int crossPoint)
{
      for (для всех i: строк таблицы child) {
            vector<int> lines1 = выбрать строки таблицы s1, в которых
                  предикаты, значимые для child, имеют те же значения,
                  что в строке i, причем, если предикат значим для обоих
                  родителей и i >= crossPoint, то его значение
                  не учитывается;
            vector<int> lines2 = выбрать строки таблицы s2, в которых
                  предикаты, значимые для child, имеют те же значения,
                  что в строке i, причем, если предикат значим для обоих
                  родителей и i < crossPoint, то его значение
                  не учитывается;

            vector<Probability> p1(nStates);
            vector<Probability> p2(nStates);
            for (для всех j из lines1) {
                  p1[целевое состояние у s1 в строке j] += 1.0;
            }
            for (для всех j из lines2) {
                  p2[целевое состояние у s2 в строке j] += 1.0;
            }
            Поделить значения p1 на число строк из lines1;
            Поделить значения p2 на число строк из lines2;
            vector<Probability> p = p1 + p2;
```

```
            child[i].targetState = выбрать случайно с распределением
                  вероятностей p;

            for (для всех k: номеров действий) {
                  Probability q1, q2;
                  for (для всех j из lines1) {
                        q1 += s1[j].output[k];
                  }
                  for (для всех j из lines2) {
                        q2 += s2[j].output[k];
                  }
                  Поделить q1 на число строк из lines1;
                  Поделить q2 на число строк из lines2;
                  child[i].output[k] = 1 с вероятностью (q1 + q2)/2
                        и 0 иначе;
            }
      }
}
```

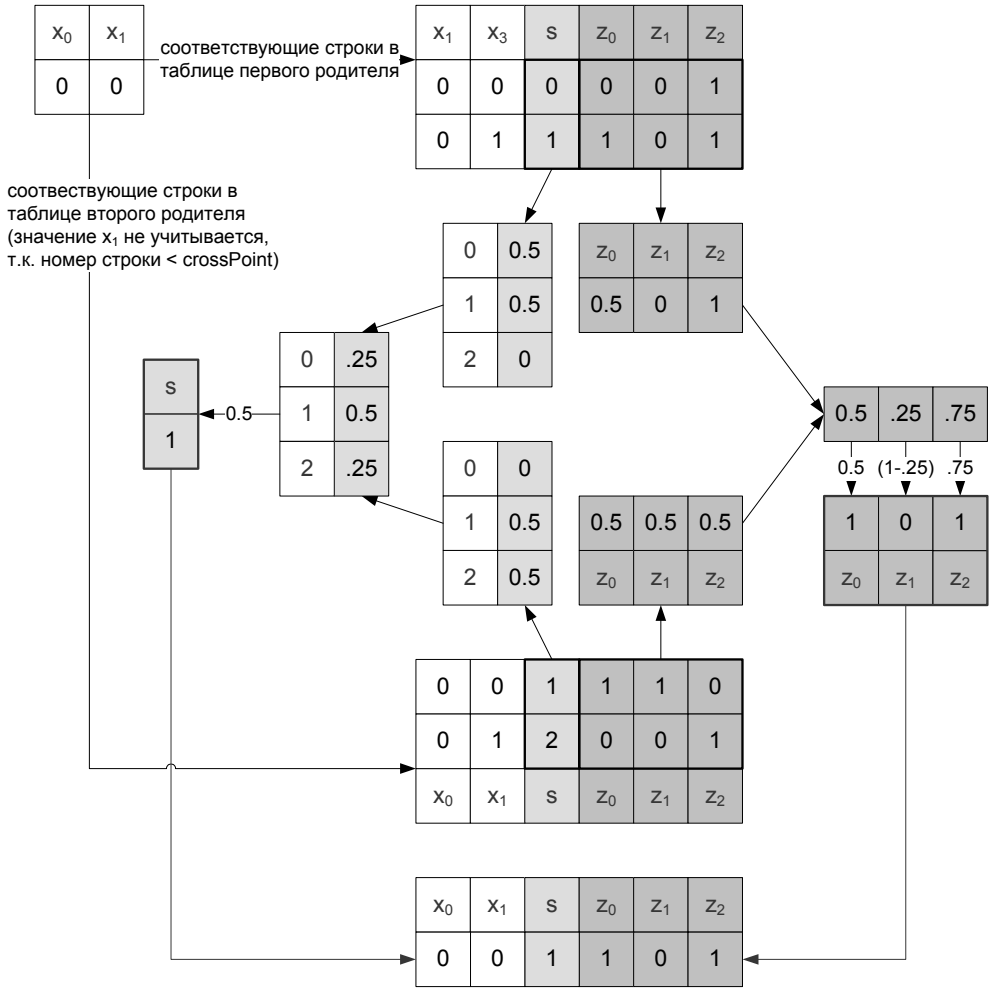


Рис. 9. Пример заполнения строки таблицы ребенка при скрещивании сокращенных таблиц

Иллюстрация примера заполнения первой строки таблицы одного из детей приведена на Рис. 9. В данной реализации оператора скрещивания на значения каждой строки таблицы ребенка влияют значения нескольких строк родительских таблиц. При этом конкретное значение, помещаемое в ячейку таблицы ребенка, определяется «голосованием» всех влияющих на нее ячеек родительских таблиц.

В описанном выше варианте алгоритма все состояния автомата имеют равное количество значимых предикатов (r – константа для всего процесса оптимизации). Однако предложенный алгоритм скрещивания легко расширяется на случай разного количества значимых предикатов у пары родителей. В этом случае необходимо добавить процедуру выбора r для каждого из детей. Такой выбор целесообразно делать случайным образом с математическим ожиданием, равным среднему арифметическому значений r пары родителей.

4. Детали применения алгоритма

4.1. Мутация, зависящая от пригодности

Что касается стратегий и параметров генетической оптимизации, основная особенность реализованного авторами подхода состоит во введении зависимости интенсивности мутаций особи от ее пригодности.

Применение интенсивной мутации к плохо приспособленным особям повышает эффективность эволюции, позволяет покидать локальные оптимумы и ослабляет проблему преждевременной сходимости популяции, тогда как для особей с высокой пригодностью оптимальны менее интенсивные мутации.

В Евклидовом пространстве оптимальный модуль вектора мутации можно с высокой точностью оценить сверху расстоянием между особью и глобальным оптимумом оценочной функции. Это расстояние в задаче не известно, однако можно ожидать тенденции к его сокращению с ростом пригодности особи. Если оценочная функция непрерывна и возрастает с приближением к оптимуму, можно проанализировать зависимость эффективности мутации от ее модуля. График на Рис. **10** отражает зависимость математического ожидания относительного сокращения расстояния до оптимума в результате попытки применения мутации.

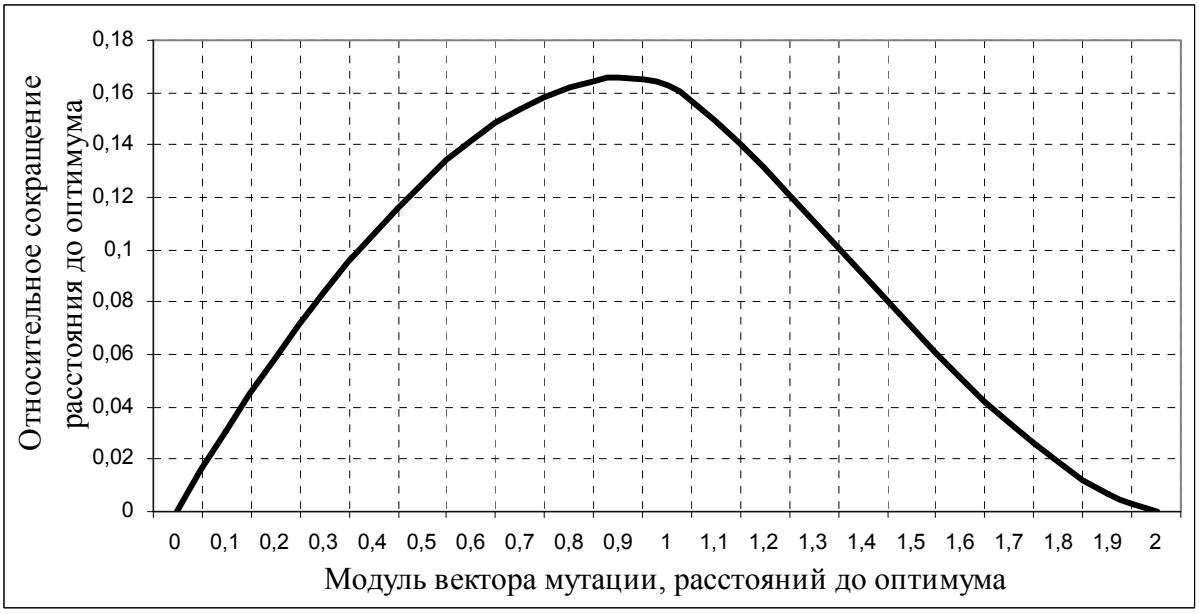


Рис. 10. Математическое ожидание сокращения расстояния до оптимума за одну операцию оценки пригодности

Можно сделать вывод, что оптимальная интенсивность мутации склонна к обратной зависимости от значения оценочной функции. В экспериментах авторы использовали интенсивность мутации, обратно пропорциональную пригодности.

4.2. Модификация оценочной функции

Задание оценочной функции на множестве вычислительных состояний объекта управления позволяет оптимизировать *поведенческие* свойства управляющего автомата. Однако, в целях улучшения понятности построенного результата оптимизации человеку, авторы считают целесообразным оптимизировать также *структурные* свойства автомата. Предлагается ввести набор стандартных структурных характеристик (например, количество достижимых управляющих состояний, суммарное число действий на переходах), которые, по желанию разработчика, будут влиять на оценку пригодности особей.

5. Экспериментальная проверка предложенных методов

В целях экспериментальной проверки предложенных методов авторами был реализован каркас *AutoGen*, позволяющий производить построение логики системы со сложным поведением, задавая в качестве входных данных только реализацию объекта управления и оценочной функции.

С помощью каркаса был реализован демонстрационный пример – построение автомата управления виртуальной «разливочной линией». Объект управления разливочной линии состоит из транспортера, на котором установлены бутылки, и дозирующей емкости (бака) с верхним (впускным) и нижним (выпускным) клапаном. Объем бака соответствует объему каждой бутылки. Перед началом работы все бутылки пусты.

Работой разливочной линии можно управлять, открывая и закрывая клапаны, а также запуская и останавливая транспортер. Управляющий автомат имеет пять двоичных входов от сигнализаторов, показывающих, правильно ли стоит бутылка под нижним клапаном, движение транспортера, опустошение и заполнение дозирующей емкости. Пятый вход автомата не несет информации и добавлен в экспериментальных целях.

Верхний клапан соединяет дозирующую емкость с трубой, подающей жидкость. Нижний клапан позволяет жидкости выливаться из бака и заполнять находящуюся под ним бутылку либо проливаться на транспортер, если бутылка не установлена. При наличии полной бутылки под нижним клапаном жидкость через клапан не проходит. Перед запуском линии бак заполнен и под нижним клапаном установлена пустая бутылка. Разливочная линия схематично изображена на Рис. 11.

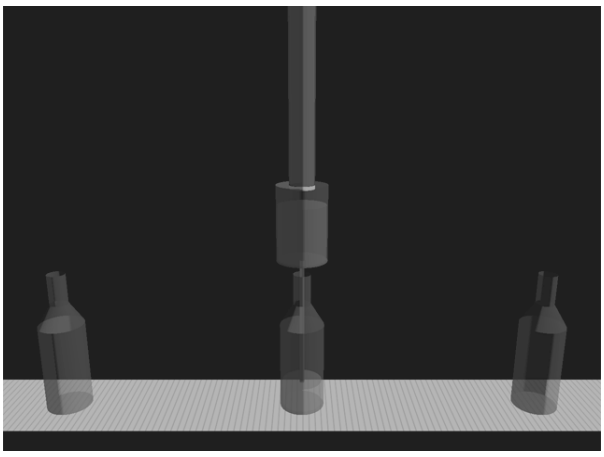


Рис. 11. Разливочная линия

Схема связей управляющего автомата разливочной линии показана на Рис. 12.

Рис. 12. Схема связей управляющего автомата разливочной линии

Задача состоит в заполнении максимального количества бутылок за определенный промежуток времени. Один из автоматов, решающих поставленную задачу, был построен авторами вручную (Рис. 13). Автомат содержит пять состояний. Здесь и далее состояние стартовым является первое состояние.

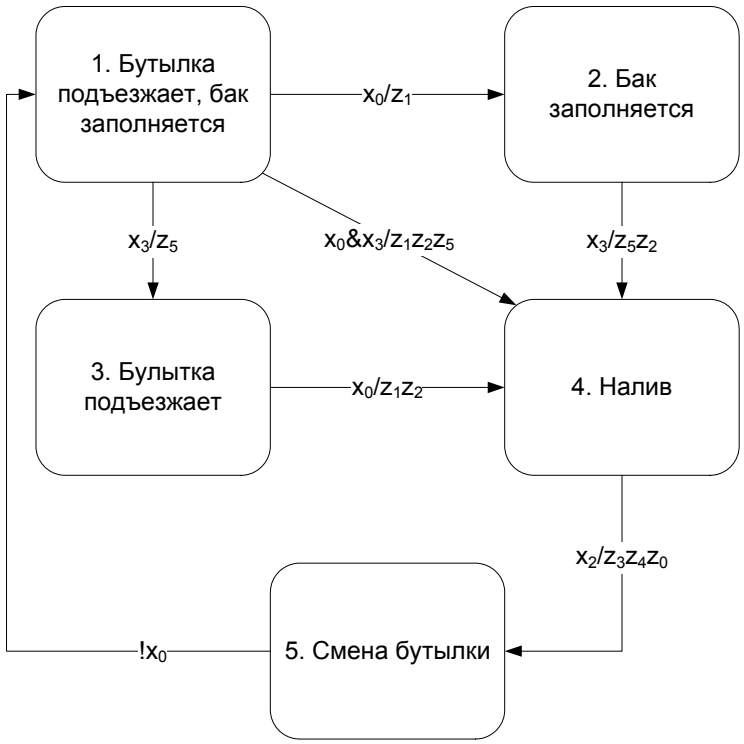


Рис. 13. Управляющий автомат разливочной линии, построенный вручную ($|S| = 5$, $\max(r) = 2$)

После этого с помощью каркаса *AutoGen* были автоматически построены автоматы с представлением состояний в виде полных и сокращенных таблиц. Автомат, представленный полными таблицами, имеет по 32 перехода из каждого состояния, что делает бессмысленным его изображение и анализ.

Автомат, представленный сокращенными таблицами, оказался значительно проще. Один из результатов оптимизации изображен на Рис. 14.

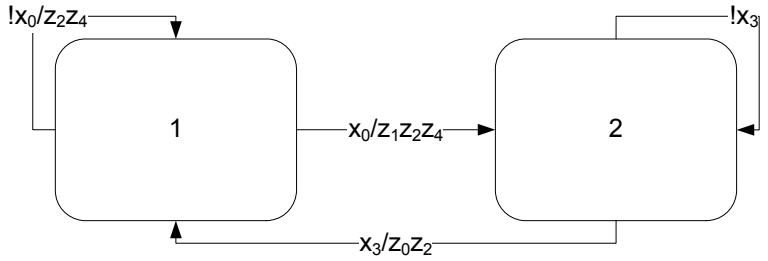


Рис. 14. Управляющий автомат разливочной линии, полученный в результате генетической оптимизации (|*S*| = 2, *r* = 1)

Анализ полученного автомата показал, что его работоспособность вызвана неявными зависимостями в эмуляторе разливочной линии. Например, этот автомат в процессе работы использует тот факт, что пропускная способность верхнего клапана вдвое больше, чем у нижнего. Таким образом, было установлено, что первоначальная реализация оценочной функции не полностью соответствует словесной формулировке задачи. Этот недостаток был впоследствии устранен путем внесения в параметры эмулятора (такие как скорость движения транспортера, пропускная способность клапанов) псевдослучайных отклонений. Результатом повторного применения генетической оптимизации стал автомат, граф переходов которого изображен на Рис. **15**.

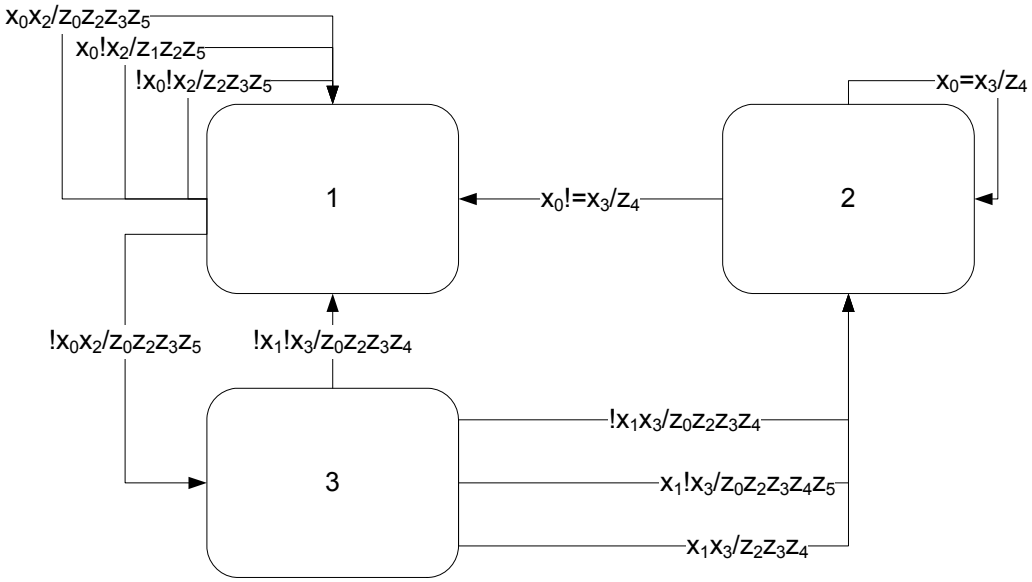


Рис. 15. Управляющий автомат "случайной" разливочной линии, полученный в результате генетической оптимизации (|*S*| = 3, *r* = 2)

В результате эксперимента авторы пришли к выводу о применимости методов генетического программирования для построения логики систем со сложным поведением. Представление состояний автоматов в форме сокращенных таблиц и разработанные для них генетические операторы показали работоспособность и эффективность. Применение предложенного подхода позволяет значительно сократить размерность пространства поиска и, соответственно, быстрее найти оптимальный управляющий автомат, пригодный для анализа и модификации.

Заключение

В работе предложены методы использования генетического программирования для построения систем со сложным поведением, теоретически обоснована целесообразность применения этих методов, а также проведена экспериментальная проверка их эффективности.

Авторы предполагают, что для дальнейшего развития генетического подхода к построению управляющих автоматов имеются широкие возможности.

Во-первых, необходимо изучить и реализовать другие решения проблемы экспоненциального роста размерности хромосомы состояния. Например, вместо ограничения числа предикатов, значимых в состоянии, можно ограничить число переходов из данного состояния (или суммарное число переходов в автомате), и задавать управляющую функцию в виде списка переходов, а не в виде таблицы. Авторам известны исследования представления управляющей функции в виде *дерева решений*.

Во-вторых, необходимо уделить внимание проблеме «долгого» вычисления оценочной функции. В генетическом программировании каждое вычисление оценочной функции подразумевает запуск модели вычисления. В случае автоматизированного объекта вычисление пригодности – это эмуляция определенного числа шагов работы заданного объекта управления совместно с оцениваемым автоматом. Такая эмуляция требует больших временных затрат по сравнению с другими этапами генетической оптимизации. Поэтому необходимо применять подходящие стратегии оценивания: турнирный метод, изменение числа шагов эмуляции в ходе оптимизации.

В-третьих, можно рассмотреть различные более общие постановки задачи построения систем со сложным поведением. В реализованном варианте вся структура объекта управления, включая предикаты и действия, задается разработчиком вручную. В то же время, для полного описания задачи достаточно задать лишь множество вычислительных состояний объекта управления и оценочную функцию на этом множестве. При таком подходе предикаты и действия могут строиться методами генетического программирования, но на основе других моделей вычисления, более подходящих для представления коротких функций, не содержащих логики. К таким моделям относятся, например, нейронные сети, язык ассемблера, упрощенные версии языков программирования высокого уровня и многие другие. Преимущества автоматного подхода к реализации систем со сложным поведением в этом случае не теряются, так как априори навязывается разделение системы на управляющий автомат и объект управления, и для каждой из этих компонент используется наиболее подходящая модель вычисления. Построенные автоматически предикаты и действия с большой вероятностью будут понятны человеку, поскольку это простые, короткие функции. Описание автомата, в свою очередь, остается высокоуровневым, как и в частном случае, подробно описанном в настоящей работе.

Можно рассмотреть также промежуточную постановку задачи, в которой структура предикатов и действий задается вручную, а некоторый набор их параметров подвергается оптимизации.

Наконец, необходимо рассмотреть подходы к представлению результатов генетической оптимизации в виде, более понятном человеку. В частности, необходимо автоматически строить граф переходов полученного автомата, а для сложных автоматов – производить их декомпозицию.

Литература

1. Шалыто А.А. SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998.
2. Шалыто А.А. Технология автоматного программирования // Мир ПК. 2003.
3. Harel D., Polity M. Modeling Reactive Systems with Statecharts. The Statemate Approach. New York: McGraw-Hill, 1998.
4. Koza J. Genetic Programming: On the programming of Computers by Means of Natural Selection. Cambridge: MIT Press, 1992.
5. Gold E.M. Language Identification in the Limit // Information and Control. 1967. № 10.
6. Belz A. Computational Learning of Finite-State Models for Natural Language Processing. PhD thesis. University of Sussex. 2000.
7. Clelland C.H., Newlands D.A. Pfsa modelling of behavioural sequences by evolutionary programming // Complex'94 – Second Australian Conference on Complex Systems. IOS Press, 1994.
8. Das S., Mozer M.C. A Unified Gradient-Descent/Clustering Architecture for Finite State Machine Induction // Advances in Neural Information Processing Systems. 1994.
9. Lankhorst M.M. A Genetic Algorithm for the Induction of Nondeterministic Pushdown Automata. Computing Science Report. University of Groningen Department of Computing Science. 1995.
10. Belz A., Eskikaya B. A genetic algorithm for finite state automata induction with an application to phonotactics // ESSLLI-98 Workshop on Automated Acquisition of Syntax and Parsing. Saarbruecken, 1998.
11. Ashlock D., Wittrock A., Wen T-J. Training finite state machines to improve PCR primer design // Congress on Evolutionary Computation (CEC'02). 2002.
12. Ashlock D.A., Emrich S.J., Bryden K.M. and others A comparison of evolved finite state classifiers and interpolated markov models for improving PCR primer design // 2004 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB'04). 2004.
13. Lucas S.M. Evolving Finite State Transducers: Some Initial Explorations // Genetic Programming: 6th European Conference (EuroGP'03). Berlin: Springer, 2003.
14. Kinnear K.E. Advances in Genetic Programming. Cambridge: MIT Press, 1994.
15. Teller A., Veloso M. PADO: A New Learning Architecture for Object Recognition // Symbolic Visual Learning. New York: Oxford University Press, 1996.
16. Banzhaf W., Nordin P., Keller R., Francone F.D. Genetic Programming – An Introduction. On the automatic Evolution of Computer Programs and its Application. San Francisco: Morgan Kaufmann Publishers, 1998.
17. Kantschik W., Dittrich P., Brameier M. Empirical Analysis of Different Levels of Meta-Evolution // Congress on Evolutionary Computation. 1999.
18. Kantschik W., Dittrich P., Brameier M., Banzhaf W. Meta-Evolution in Graph GP // Genetic Programming: Second European Workshop (EuroGP'99). 1999.
19. Teller A., Veloso M. Internal Reinforcement in a Connectionist Genetic Programming Approach // Artificial Intelligence. 2000.
20. Brameier M., Kantschik W., Dittrich P., Banzhaf W. SYSGP – A C++ library of different GP variants. Internal Report. Univ. of Dortmund. 1998.
21. Benson K. Evolving Automatic Target Detection Algorithms that Logically Combine Decision Spaces // Eleventh British Machine Vision Conference. 2000.
22. Kantschik W., Banzhaf W. Linear-Graph GP. A new GP Structure // 4th European Conference on Genetic Programming (EuroGP'02). 2002.
23. Brave S. Evolving Deterministic Finite Automata Using Cellular Encoding // Genetic Programming 1996: First Annual Conference. 1996.

24. Miller J.F., Thomson P. A Developmental method for growing Graphs and Circuits // Evolvable Systems: From Biology to Hardware. Berlin: Springer, 2003.
25. Poli R. Evolution of Graph-like Programs with Parallel Distributed Genetic Programming // Genetic Algorithms: Seventh International Conference. 1997.
26. Frey C., Leugering G. Evolving Strategies for Global Optimization. A Finite State Machine Approach // Genetic and Evolutionary Computation Conference (GECCO-2001). Morgan Kaufmann, 2001.
27. Petrovic P. Simulated evolution of distributed FSA behaviour-based arbitration // The Eighth Scandinavian Conference on Artificial Intelligence (SCAI'03). 2003.
28. Petrovic P. Evolving automatons for distributed behavior arbitration. Technical Report. Norwegian University of Science and Technology. 2005.
29. Petrovic P. Comparing Finite-State Automata Representation with GP-trees. Technical report. Norwegian University of Science and Technology. 2006.
30. Koza J.R. Future Work and Practical Applications of Genetic Programming // Handbook of Evolutionary Computation. Bristol: IOP Publishing Ltd, 1997.
31. Handley S. A new class of function sets for solving sequence problems // Genetic Programming 1996: First Annual Conference. 1996.
32. Langdon W. B. Genetic Programming and Data Structures. Boston: Kluwer, 1998.
33. Fogel L.J. Autonomous Automata // Industrial Research. 1962. № 4.
34. Fogel L.J. On the organization of intellect. PhD thesis. University of California. 1964.
35. Fogel L.J., Owens A.J., Walsh M.J. On the Evolution of Artificial Intelligence // 5th National Symposium on Human Factors in Electronics. San Diego, 1964.
36. Fogel L.J., Owens A.J., Walsh M.J. Artificial Intelligence through a Simulation of Evolution // Biophysics and Cybernetic Systems. London: Macmillan & Co, 1965.
37. Fogel L.J., Owens A.J., Walsh M.J. Artificial Intelligence through Simulated Evolution. New York: Wiley, 1966.
38. Fogel L.J. Extending Communication and Control through Simulated Evolution // Bioengineering – An Engineering View: Symp. Engineering Significance of the Biological Sciences. 1968.
39. Lutter B.E., Huntsinger R.C. Engineering applications of finite automata // Simulation. 1969. № 1.
40. Dearholt D.W. Some Experiments on Generalization Using Evolving Automata // 9th Intern. Conf. on System Sciences. Honolulu, 1976.
41. Atmar J.W. Speculation on the Evolution of intelligence and its possible realization in machine form. PhD thesis. New Mexico State University. 1976.
42. Takeuchi A. Evolutionary Automata – Comparison of Automaton Behavior and Restle's Learning Model // Information Science. 1980. № 2.
43. Burgin G.H. On playing two-person zero-sum games against nonminimax players // IEEE Trans. on Systems Science and Cybernetics. 1969. № 4.
44. Miller J.H. The Coevolution of Automata in the Repeated Prisoner's Dilemma. Working Paper. Santa Fe Institute. 1989.
45. Spears W.M., Gordon D.F. Evolving Finite-State Machine Strategies for Protecting Resources // International Syposium on Methodologies for Intelligent Systems. 2000.
46. Ashlock D. Evolutionary Computation for Modeling and Optimization. New York: Springer, 2006.
47. Fogel L.J., Angeline P.J., Fogel D.B. A Preliminary Investigation on Extending Evolutionary Programming to Include Self-adaptation on Finite State Machines // Informatica. 1994. № 4.
48. Carmel D., Markovitch S. Learning models of intelligent agents // Thirteenth National Conference on Artificial Intelligence. 1996.
49. Hsiao M.S. Sequential Circuit Test Generation Using Genetic Techniques. PhD thesis. University of Illinois at Urbana-Champaign. 1997.

ПОСТРОЕНИЕ ИЕРАРХИИ КЛАССОВ ПО ТЕКСТОВЫМ ОПИСАНИЯМ

А.А. Бреслав, А.П. Лукьянова, М.А. Коротков
Научный руководитель – к.ф.-м.н., с.н.с. Ф.А. Новиков
(Санкт-Петербургский государственный политехнический университет)

Цель настоящей работы – автоматизация объектно-ориентированного анализа предметной области, описанной на естественном (английском) языке. Описываемый подход позволяет строить первичную иерархию классов, в дальнейшем уточняя ее посредством автоматического рефакторинга и статического анализа текста. Распознаются классы, наследование, атрибуты и методы.

Введение

Анализ – один из важнейших этапов жизненного цикла программной системы, когда, изучая предметную область, разработчики выделяют абстракции, определяющие концептуальную структуру будущей системы. Существует не так уж много различных методик анализа. Гради Буч (Grady Booch) в работе [1] выделяет три классических подхода (классическая категоризация, концептуальная кластеризация, теория прототипов) поведенческий подход, основанный на ответственности, и использование уже существующих достижений (паттерны). Кроме того, Буч упоминает работу Рассела Дж. Аббота (Russel J. Abbott) [2], в которой описан неформальный метод анализа, основанный на текстовом описании. Принцип, по которому Аббот предлагал выделять классы и объекты, основывается на синтаксической роли слов в предложении: подлежащее, скорее всего, является объектом, сказуемое описывает его поведение, определения описывают атрибуты и т.д.

С момента публикации Абботом этой работы метод был уточнен и расширен. Аббот опирался в основном на части речи, считая, например, что определение – это непременно прилагательное. Позднее было отмечено, что не только прилагательные, но и существительные могут играть эту роль [4, 5]. Были предложены специальные трактовки для отдельных слов и конструкций-связок (таких как «all», «any», «within», «can be» и т.д.) [6–8].

Были созданы и автоматические средства анализа. Некоторые из них распознают лишь объекты, не выделяя классов [9, 10]. Но есть и такие, которые распознают классы и ассоциации между ними [11, 12]. Однако ни одна из известных нам систем не обнаруживает наследования.

Существуют системы, использующие доменную информацию в качестве исходных данных (в дополнение к тексту) [13, 14], но эта информация слишком конкретна: пользователю фактически предлагается самостоятельно определить объекты и методы.

Выходные данные подобных систем могут быть формализованы по-разному. Весьма распространено использование для этой цели различных диаграмм. Системы, описанные в работах [13, 14], могут строить диаграммы UML [15].

Целью настоящей работы является построение системы, позволяющей обнаруживать классы, их атрибуты и методы, а также выделять отношение наследования.

Постановка задачи

В качестве примера рассмотрим следующий текст, описывающий предметную область «Магазин».

The shop sells food. The shop sells drinks. The food's price is a double. The drinks' price is a double. There is a shop assistant in the shop. The consultant, shop assistant, consults a customer. When the customer comes the customer buys food. The customer buys drinks. There is a cash desk in the shop. The cash desk gives out a cheque.

Этот текст описывает взаимодействие следующих сущностей (классов): магазин (shop), продавец (shop assistant), консультант (consultant), еда (food), напитки (drinks), касса (cash desk), чек (cheque) и покупатель (customer). Сущности имеют атрибуты: для еды и напитков – цена (price), для магазина – продавец и касса. Также они имеют методы: магазин продает (sells), консультант консультирует (consults), покупатель приходит (comes) и покупает (buys), касса выдает чек (gives out a cheque).

Таким образом, текст фактически описывает структуру классов. Задача предлагаемого метода – распознать эту структуру и сформировать классы, связанные отношениями наследования.

Математическая модель

В качестве модели для работы с текстом был выбран граф объектных отношений. Этот граф отображает информацию о семантической структуре текста, он представляет собой неоднородную бинарную семантическую сеть [16].

Граф строится из подграфов, каждый из которых представляет одно предложение исходного текста. Принцип построения графа основывается на методе Аббота, т.е. считается, что подлежащее, как правило, представляет объект, сказуемое – метод, зависимые слова описывают главное, т.е. являются его атрибутами или параметрами, и т.д.

Каждое предложение порождает некоторый подграф (см. рис. 1). Вершины в этом подграфе представляют роли тех или иных слов или понятий, а ребра – связи между вершинами, выражаемые бинарными отношениями.

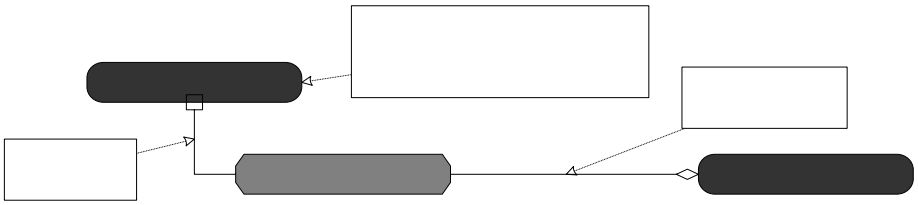


Рис. 1. Граф предложения «There is a shop assistant in the shop»
(«В магазине есть продавец»)

Одна вершина может соответствовать словосочетанию, и одно слово может порождать несколько вершин. Последняя ситуация реализуется, например, при обнаружении атрибута ранее неизвестного типа.

В предложении на рис. 1 словосочетание «shop assistant» трактуется как одно понятие, а это понятие представляется двумя вершинами в графе: объектом, играющим роль типа, и атрибутом другого объекта.

В модели используются следующие типы вершин (см. рис. 2).

- Объект – упоминание объекта, т.е. некоторой описываемой единицы.
- Атрибут – упоминание характеристики, т.е. описание аспекта состояния.
- Метод – упоминание действия некоторого объекта, т.е. описание аспекта поведения.
- Параметр – упоминание параметра (характеристики) действия.

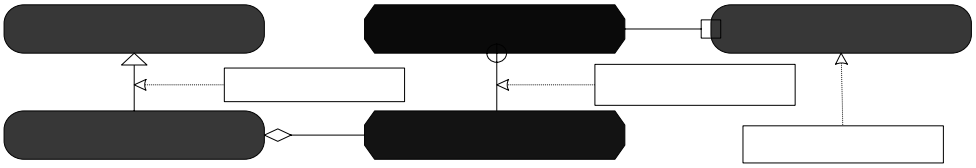


Рис. 2. Граф предложения «The consultant, shop assistant, consults a customer»
(«Консультант, будучи продавцом, консультирует покупателя»)

Используются следующие типы ребер (см. рис. 2).

- Принадлежность элемента интерфейса конкретному объекту.

- Наследование одним объектом свойств другого. Если объект А наследуется от объекта В, значит, класс, к которому принадлежит А, является частным случаем класса, к которому принадлежит В.
- Использование одним объектом свойств другого. Если такое ребро указывает на объект, это означает использование объекта в целом. Также оно может указывать на конкретный атрибут или метод. Началом ребра может быть не только объект, но и один из его методов.
- Использование методом данного параметра.
- Возвращаемое значение метода.

Все вершины и связи локализованы в одном предложении, поскольку именно предложение является наиболее крупным подразделением текста, в семантической целостности которого можно быть уверенным. Графы предложений могут быть связаны между собой, например, если установлено, что местоимение в одном предложении представляет некоторое полнозначное слово в другом[1].

Формирование объектной модели ведется по следующему общему принципу (см. рис. 3): объекты с одинаковыми именами, обнаруженные в различных предложениях объединяются в классы. Интерфейсы этих классов строятся как объединение интерфейсов всех объектов, входящих в класс, т.е. все атрибуты и методы, связанные с этими объектами отношением принадлежности, составляют интерфейс нового класса.

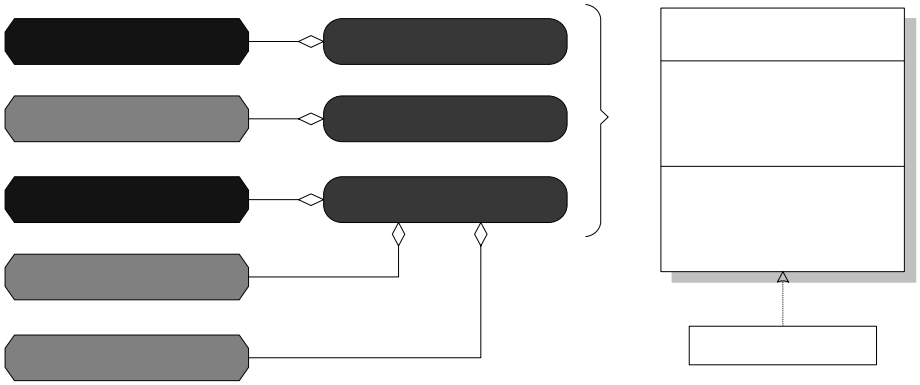


Рис. 3. Объединение объектов в класс

Терминология

Далее под словами «группа атрибутов» (объектов, методов) будем понимать все вершины соответствующего типа, имеющие одинаковые имена (эти вершины могут находиться в разных предложениях).

Вершины графа объектных отношений являются промежуточным звеном между словами в тексте и элементами конечной структуры (классами, атрибутами и методами). Группа вершин одного типа порождает класс, атрибут или метод.

Заметим, что объекты не являются элементами конечной структуры (они объединяются в классы), поэтому понятие «объект» обозначает именно вершину графа, т.е. экземпляр класса. Группа объектов формирует класс (см. рис. 4).

[1] В настоящее время задача об установлении таких связей не решена

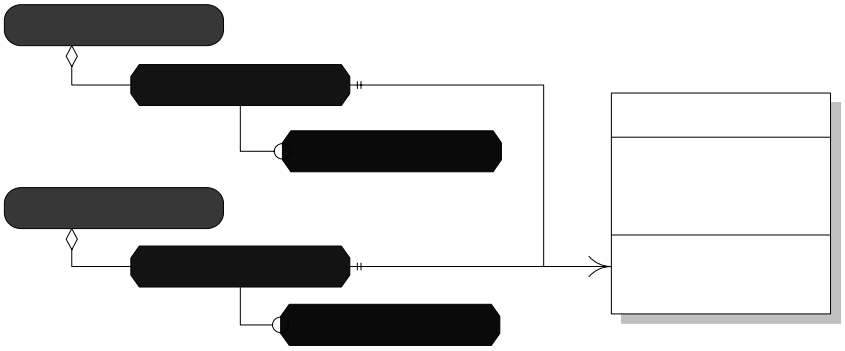


Рис. 4. Порождение метода группой вершин

Дополнительные структуры данных

Для дальнейшей работы потребуются дополнительные структуры данных. Все слова и словосочетания, представленные вершинами графа, заносятся в *словарь*, причем для каждого слова хранится список вершин каждого типа, порожденных этим словом.

Также потребуется понятие *категории*. Категория – это группа слов, объединенных общим смыслом. Структура категорий может быть очень сложной, в идеале она напоминает файловую систему, где роль файла играет слово из словаря, а роль каталога - категория. Как и в файловой системе, каждая категория имеет уникальное имя.

В тексте могут встретиться термины, специфичные для предметной области, которых нет в предопределенном словаре категорий. Здесь возникает задача самообучения программы. В прототипе эта проблема решена самым простым способом: каждое неизвестное слово автоматически помещается в категорию с таким же именем.

Сформированные элементы конченой структуры помещаются в специальное *хранилище*, из которого можно сформировать объявления классов на любом языке программирования или диаграмму UML.

Спецификаторы классов

Одним из важнейших в нашей модели является понятие *спецификатора*.

Спецификатор класса – это атрибут, достаточно выразительный для того, чтобы его наличие говорило о принадлежности объекта отдельному классу.

Расширяя пример с магазином, опишем некоторую классификацию товаров.

*The shop sets **discounted price**. The **discounted price** is lower than **usual price**.*

Мы описали классы «Цена со скидкой» и «Обычная цена» (см. рис. 6), поскольку для объектов этих классов значение вычисляется по-разному. Фактическим критерием для нас служит разница в способе вычисления, но анализ принципа работы метода (вычисления стоимости) по неформальному описанию – задача крайне трудная, и тут на помощь приходит эвристический механизм выделения спецификаторов.

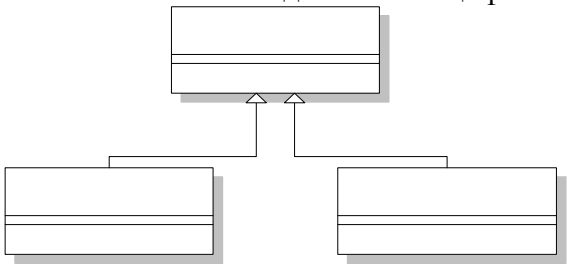


Рис. 5. Спецификаторы классов

Дело в том, что, описывая некоторый специфичный аспект поведения класса, мы вынуждены каждый раз указывать спецификатор, чтобы отличать разновидности товаров. Так, в приведенном примере использованы слова «discounted price» и «usual price». Вершины «discounted» и «usual» являются атрибутами, однако фактически они неразрывно связаны с определяемым понятием и обозначают *подклассы*.

Такое положение вещей позволяет нам использовать статистические данные о встречаемости атрибута для принятия решения о выделении спецификатора. Выделение спецификаторов очень важно, поскольку оно позволяет установить связи наследования между классами.

Для выделения спецификаторов классов рассмотрим те слова из словаря, которые являются именами объектов. Если ни один из таких объектов не имеет входящих ребер наследования, рассмотрим все атрибуты, принадлежащие этим объектам.

Не любой атрибут может служить спецификатором. Некоторые атрибуты могут изменять свои значения в процессе жизненного цикла объекта, но спецификатор является элементом индивидуальности объекта и изменяться не может, поэтому, если есть явное указание на использование значения атрибута некоторым методом или объектом (отношение использования), то выделение спецификатора по этому атрибуту невозможно. Поэтому атрибуты, связанные отношением использования, далее не рассматриваются.

Для каждого класса (группы объектов с одинаковыми именами) рассмотрим все категории, в которые входят имена выбранных атрибутов. Выберем категорию, элементы которой наиболее часто встречаются с выбранными объектами, и на каждый элемент в среднем приходится достаточно большое число упоминаний. Такая категория будет хорошим кандидатом на роль спецификатора (рис. 7).

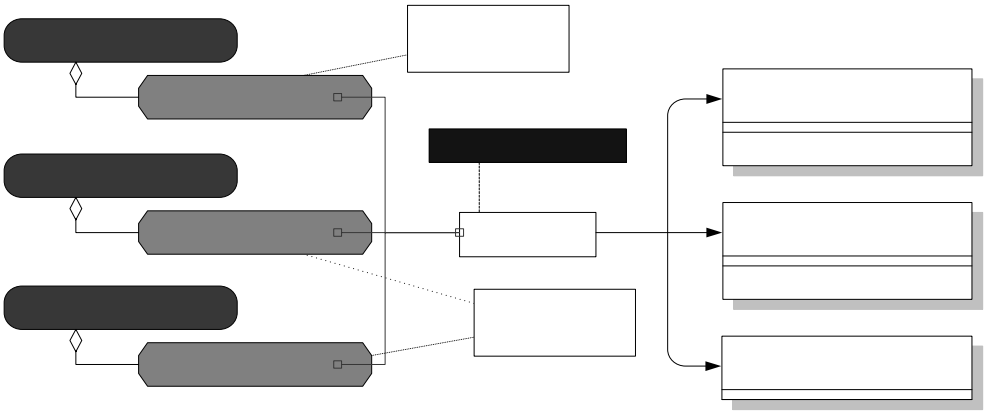


Рис. 6. Замена значений атрибутом, по спецификаторам

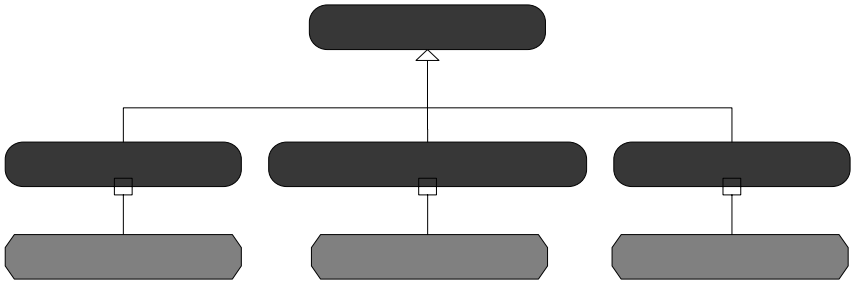


Рис. 7. Выбор суперкласса для типов атрибутов

Почему в качестве спецификатора выбирается категория? Дело в том, что фактическое имя атрибута (например, «цвет») может в тексте явно не указываться, встречаться могут только его значения (например, «желтый», «зеленый», «синий»). Эти значения должны быть объединены в единый атрибут с именем категории, но делать это до выделения спецификаторов нельзя. Таким образом, вершина типа «атрибут» может представлять собой не собственно атрибут класса, а лишь его значение, а если мы выделяем

специфицированный класс по одному из значений данного атрибута, то нужно выделить и другие классы – по остальным значениям.

Теперь добавим в хранилище новый класс, который будет родительским для выделенных по спецификатору. Каждый элемент из выбранной категории порождает новый класс, наследуемый от этого.

Типизация атрибутов

После выделения спецификаторов классов у объектов одного класса (т.е. с одним именем) могут быть атрибуты, имеющие одинаковые имена, но разные типы (в одном предложении использовался один спецификатор класса, а в другом другой). Для решения этой проблемы предлагается выбрать суперкласс для типов этих атрибутов и установить его как общий тип (рис. 8).

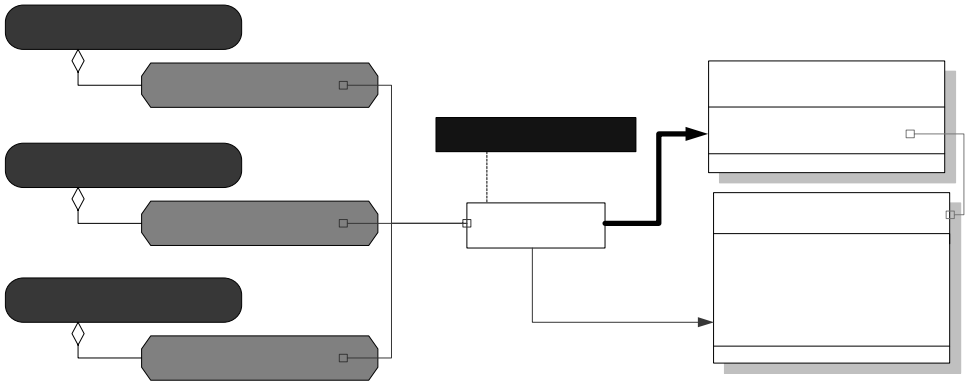


Рис. 8. Замена значений атрибутом

Для этого будем спускаться сверху вниз по иерархии наследования, будем рассматривать все атрибуты каждого класса. Для каждого атрибута возможны три принципиально разные ситуации:

- для атрибута известен тип;
- тип неизвестен, но известна категория;
- не известны ни тип, ни категория (этот случай реализуется для специфицированных классов, имена которых не принадлежат категориям).

По этому признаку атрибуты разбиваются на три списка.

Теперь необходимо обратиться к потомкам текущего класса и рассмотреть их атрибуты. Встретив атрибут, имя которого уже содержится в первом списке, добавим его в этот список. Если категория рассматриваемого атрибута встретилась во втором списке, добавить его во второй список.

После завершения обхода поддерева иерархии наследования все атрибуты родительского класса и часть атрибутов потоков будут распределены по трем спискам. Теперь нужно перенести эти атрибуты в хранилище.

Первый список – типизированные атрибуты. В этом списке могут содержаться атрибуты с одним именем, но разными типами. Для таких атрибутов необходимо найти общий суперкласс их типов и установить его как общий тип для всех. Атрибут удаляется из интерфейсов классов-наследников и сохраняется только в родительском классе, поскольку потомки унаследуют его автоматически.

Второй список – категории. Имена категорий из второго списка добавляются в хранилище как атрибуты. Областями их значений будут константы с именами конкретных слов из категории. Например, категория «цвет» порождает атрибут Цвет со значениями (красный, желтый, зеленый) (рис. 8).

Спецификаторы методов

Спецификатор метода – это параметр, достаточно выразительный для того, чтобы его наличие говорило о наличии отдельного метода.

Принцип работы алгоритма спецификации методов во многом схож с алгоритмом спецификации классов. Параметры играют роль атрибутов, а вместо наследования мы просто сохраняем неспецифицированный метод, чтобы специфицированные могли его использовать. Та же схема – с типизацией параметров, но здесь богатую почву для исследований открывает возможность перегрузки. В настоящее время перегрузка не поддерживается, но ее наличие даст возможность очень гибко работать с методами.

Дополнительно необходимо обрабатывать типы возвращаемых значений функций.

Наследование

Формирование связей наследования по тексту (отношение наследования) и по спецификаторам является довольно точной, но не очень обширной эвристикой. Рассматриваемый далее метод позволяет в значительной мере использовать данные, хранимые структурой категорий для выявления отношения наследования в предметной области.

К настоящему моменту работа с графом отношений завершена. Все выделенные элементы конечной структуры находятся в хранилище.

Наследование по категориям

Для каждой категории рассмотрим множество классов из хранилища, названных словами этой категории. Эти классы могут быть родственными, поскольку их объединяет семантика названий (рис. 9). Если в интерфейсах этих классов имеются схожие элементы, имеет смысл создать для них общий суперкласс, и вынести туда общие атрибуты и методы. При подъеме атрибутов (выбранных по общности имен) необходимо производить типизацию по методу, изложенному выше, т.е. выбирать в качестве типа общий суперкласс типов имеющихся объектов. Методы также выбираются по общности имени и приводятся к единой сигнатуре (перегрузка не поддерживается) по тому же принципу, по какому классы строятся из объектов: производится типизация параметров.

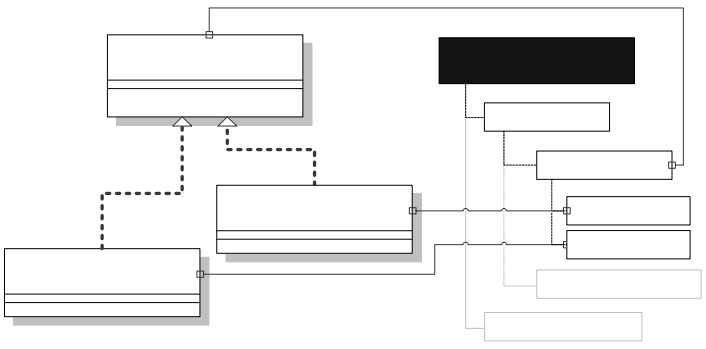


Рис. 9. Перенос ребер из категорий в хранилище

Улучшение имеющейся иерархии наследования

В процессе формирования иерархии наследования могли возникнуть ситуации, требующие применения рефакторинга. Рассмотрим следующий пример: интерфейсы некоторых (но не всех) подклассов класса **Sample** имеют непустое пересечение, т.е.

возникла необходимость в подъеме поля или подъеме метода (см. [17], рис. 11). Для улучшения имеющейся ситуации будем рассматривать классы, поднимаясь по иерархии наследования снизу вверх. Для каждого класса рассмотрим множества его непосредственных потомков и, находя общие атрибуты и методы, будем выделять новые классы, наследуя их от текущего.

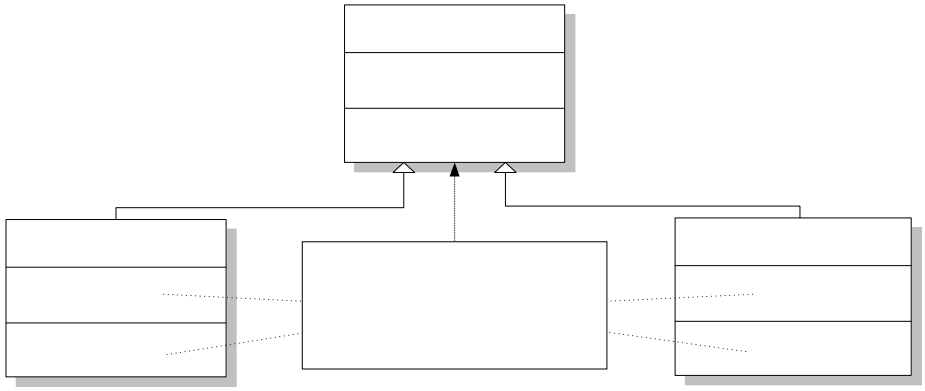


Рис. 10. Рефакторинг: подъем поля и метода

Реализация описанного метода

Разработанный прототип системы, названный ObjectSage, позволяет пользователю ввести текстовое описание и строит по нему граф объектных отношений, диаграмму классов и заголовочный файл C++ (см. рис. 11).

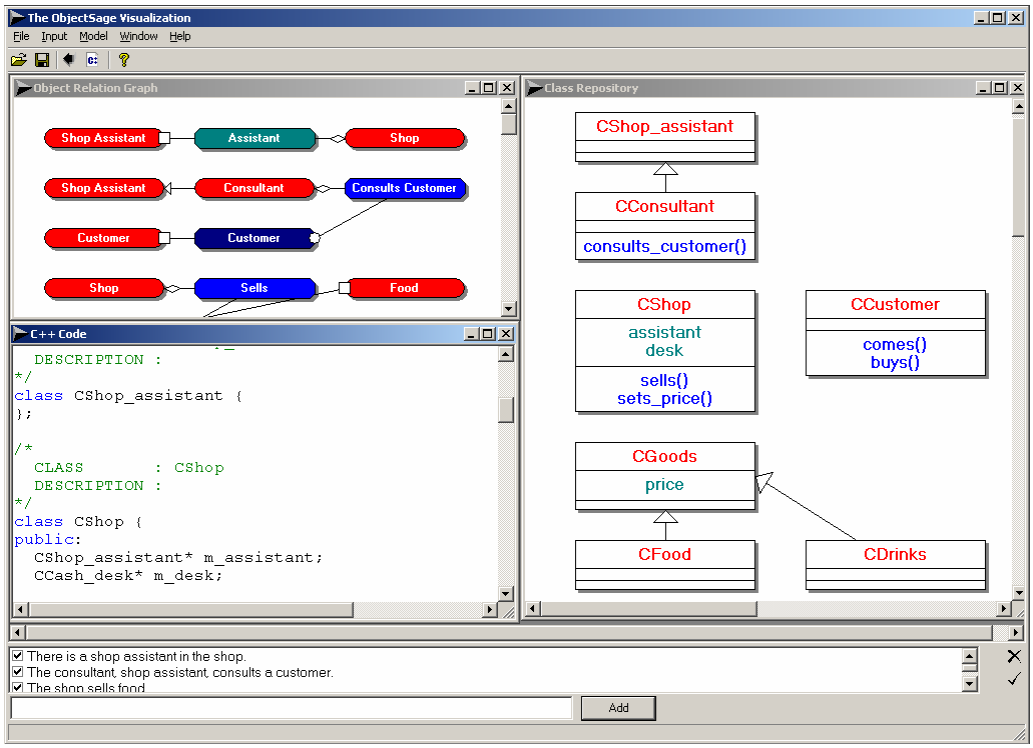


Рис. 11. Скриншот системы ObjectSage

Анализ английского языка и построение графа объектных отношений выполняется с помощью *Link Grammar Parser* [18] – разработки Carnegie Mellon University. С помощью этого пакета удалось выявить четыре из пяти типов ребер (кроме возвращаемого значения метода) и все типы вершин. Задачу о связях между предложениями по местоимениям средствами *LGP* решить не удалось. Разработанный прототип способен обрабатывать простые предложения без местоимений, косвенной речи, страдательного

залога и прочих сложных конструкций. Поддерживается работа с категориями, выделение спецификаторов и рефакторинг конечной структуры (подъем общих элементов в суперкласс). Распознаются перечислимые и примитивные типы.

Используемая структура категорий – плоская, т.е. нет вложенных категорий. Считается, что каждое слово принадлежит ровно одной категории.

Заключение

Работа с прототипом системы показывает, что предлагаемые методы пригодны для решения поставленной задачи. Применяя их, удается получить качественные модели предметной области (принимая во внимание ограничения, накладываемые средствами анализа текста). Вмешательство пользователя в процесс работы не требуется. Вся доменно-специфичная информация организуется в виде структуры категорий, причем не требуется никакого семантического описания понятий – только классификация.

Получаемые модели достаточно информативны для генерации по ним декларативной части программы, что говорит о том, что предлагаемые методы могут быть использованы при решении задач генеративного программирования.

Для достижения более полного объема функциональности необходимо использовать более сложные методы анализа текста, причем нет необходимости качественно изменять алгоритмы, предлагаемые в данной работе – необходимо лишь реализовать те концепции, которые еще не реализованы.

Литература

1. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на C++ // СПб.: Невский диалект, 1999.
2. Abbott R.J. Program Design by Informal English Descriptions // Communications of the ACM. 1983. Vol. 26, 11. P. 882–894.
3. Buchholz H., Dusterhoft A., Thalheim B. Capturing Information on Behaviour with the RADD_NLI: A Linguistic and Knowledge Base Approach // Proceedings of Second Workshop Application of Natural Language to Information System, IOS Press, Amsterdam, 1996., P. 185–196.
4. Kristen G. Object Orientation: The KISS-Method: From Information Architecture to Information Systems // Addison-Wesley, Reading, Mass., 1994.
5. Rolland C., Proix C. A Natural Language Approach for Requirements Engineering // Proceeding on Conference Advanced Information Systems Engineering. Springer-Verlag, Manchester, UK. 1992. P. 257–277.
6. Tjoa A.M., Berger L. Transformation of Requirements Specification Expressed in Natural Language into an EER Model// Proceeding: 12th International Conference on ER Approach. Springer-Verlag, Berlin. 1993. P. 206–217.
7. Juristo N., Moreno A.M. How to Use Linguistic Instruments for Object-Oriented Analysis // IEEE Software, May/June 2000. P. 80–89.
8. Mich L. From Natural Language to Object Oriented Using the Natural Language Processing System LOLITA // Natural Language Engineering, 2(2), 1996. P.161–187.
9. Delisle S., Barker K., Biskri I. Object-Oriented Analysis: Getting Help from Robust Computational Linguistic Tools. // The Fourth International Conference on Applications of Natural Language to Information Systems (OCG Schriftenreihe 129), Klagenfurt, Austria, 1999, 167–171.
10. Harmain H.M., R. Gaizauskas. CM-Builder: An Automated NL-based CASE Tool // In Proceedings of the 15th IEEE International Conference on Automated Software Engineering (ASE'2000), 2000. P. 45–53.

11. Harmain H.M., Gaizauskas R. CM-Builder: A Natural Language-based CASE Tool // Journal of Automated Software Engineering, 10. 2003. P. 157–181.
12. Börstler J. User-Centered Requirements Engineering in RECORD – An Overview // The Nordic Workshop on Programming Environment Research, Proceedings NWPER'96, Aalborg, Denmark, May 1996. P. 149–156.
13. Overmyer S.L.V., Rambow O. Conceptual Modeling through Linguistics Analysis Using LIDA // 23rd international conference on Software engineering. July 2001.
14. Perez-Gonzalez H.G., Kalita J.K. GOOAL: A Graphic Object Oriented Analysis Laboratory // OOPSLA'02, November 2002. P. 38–39.
15. Фаулер М., Скотт К. UML. Основы. // СПб: Символ-Плюс. 2003. 192 с.
16. Sowa J.F. Semantic Networks // www.jfsowa.com/pubs/semnet.htm
17. Фаулер М. Рефакторинг: улучшение существующего кода. // СПб: Символ-Плюс. 2003. 432 с.
18. Temperley D., Sleator D., Lafferty J. Link Grammar Parser // Carnegie Mellon University, http://www.link.cs.cmu.edu/link/

АВТОМАТНОЕ РАСШИРЕНИЕ ЯЗЫКА C#

М.Г. Раер
Научный руководитель – д.т.н., профессор А.А. Шалыто

Работа рассматривает паттерн проектирования State, его модификацию State Machine, описывает модификацию State Machine для языка C# на основе синтаксических событий языка.

Введение

Часто поведение объекта, его реакция на определенные события зависит от состояния, в котором он находится. Такие объекты принято описывать конечными автоматами. Существует большое количество способов реализации автоматов в программировании [1, 2]. Рассматривается весь спектр вариантов, от абсолютно не объектно-ориентированного, когда автомат реализуется вложенными операторами *if* или *switch*, до полностью объектно-ориентированного, где все элементы автомата (состояния, события, переходы, действия) реализованы как классы. Самой распространенной объектно-ориентированной реализацией является паттерн State [3]. В работе [1] рассмотрен список различных модификаций этого паттерна.

Паттерн проектирования State предлагает вынести код, отвечающий за поведение объекта в неком состоянии, в отдельный класс, и в методах этого класса, в зависимости от неких условий, производить переход в другие состояния. Таким образом, получается, что код, отвечающий за смену состояний, рассредоточен по классам состояний, что затрудняет его модификацию и поддержку и нарушает принцип централизованной логики управлений, предложенной в SWITCH-технологии [4]. Также возникает зависимость между классами состояний, так как они должны «знать» друг о друге.

Авторами работы [6] был предложен новый паттерн State Machine, устраняющий эту зависимость и централизующий логику управления в одном месте. В работе приводилась реализация на языке Java, поэтому в классы состояний пришлось ввести члены события (объекты специального класса `Event`) и наследовать состояния от базового класса StateBase. Это вносит некоторые ограничения на использование классов в качестве классов-состояний.

В работе [7] предлагается расширение языка Java для более удобной реализации паттерна State Machine.

В настоящей работе предлагается реализация паттерна State Machine для языка C# [8], снимающая указанные выше ограничения. Также предлагается дополнение языка C# (State#) для более эффективной реализации паттерна. В качестве состояний становится возможным использовать произвольный класс, автомат может реализовывать произвольное количество автоматных интерфейсов. В язык State# вводится возможность автоматического протоколирования, событие смены состояния. Условия перехода можно задавать не только событиями, но и дополнительными условиями, предусмотрена возможность описывать действия на переходе.

1. Обзор паттернов state и state machine

1.1. Паттерн State

Паттерн State позволяет объекту изменять свое поведение в зависимости от своего состояния. Он входит в группу паттернов поведения.

Вводится абстрактный класс `State`, определяющий интерфейс объекта. Подклассы класса `State` задают поведение объекта. Каждому состоянию соответствует подкласс. Определяется основной класс `Context`. В нем хранится объект текущего состояния. Класс `Context` делегирует все запросы из интерфейса `State` этому объекту.

Методы наследников класса `State` могут изменять значение этого объекта, изменяя тем самым состояние системы.

1.2. Паттерн State Machine

В работе [6] авторами предлагается паттерн State Machine, устраняющий вышеуказанные недостатки. На рис. 1 изображена его структура.

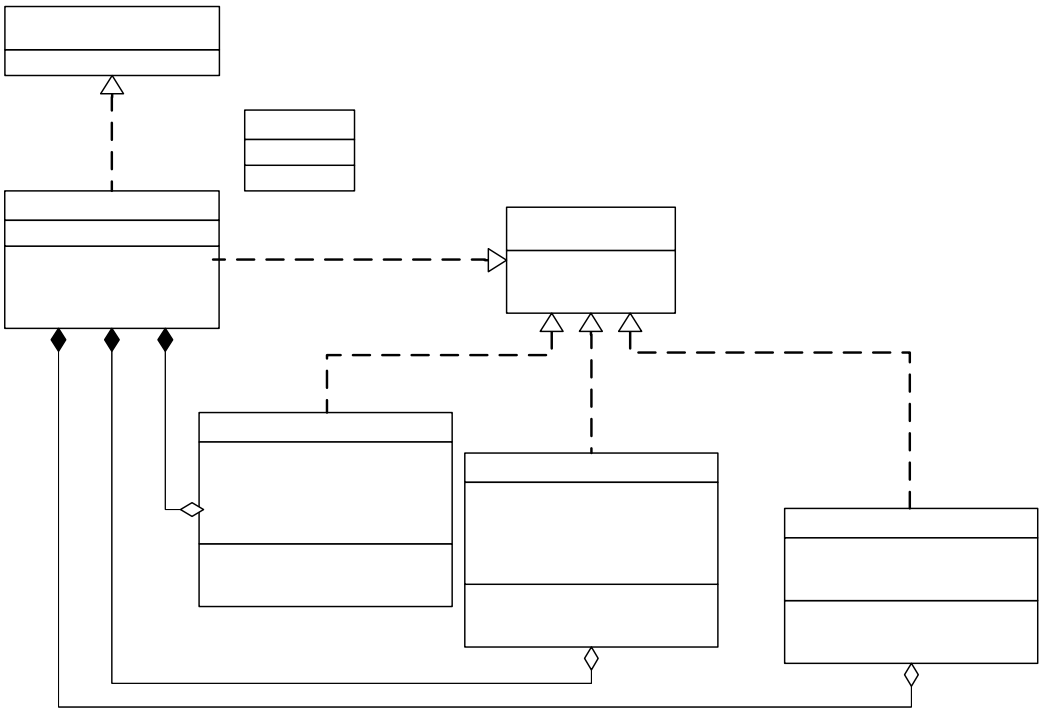


Рис. 1. Структура паттерна State Machine

Для того чтобы избавиться от зависимости классов состояний между собой, в паттерне State Machine в классы состояний были добавлены события.

Приводилась реализация на языке Java.

События – это объекты, которые передаются состояниями контексту и сообщают о необходимости смены состояния. Таким образом, вся логика переходов централизуется в классе контекста. Класс контекста на основе текущего состояния и пришедшего события определяет новое состояние.

Добавление классу состояния события реализовывалось путем добавления в этот класс члена некого класса `Event`.

```
public class ConnectedState <AI extends IConnection> extends StateBase<AI>
implements IConnection {
   public static final Event DISCONNECT = new Event("DISCONNECT");
   public static final Event ERROR = new Event("ERROR");
…
}
```

Кроме того, любое состояние через базовый класс `StateBase` должно было агрегировать интерфейс `IEventSink`, для уведомления контекста о смене состояний.

```
public abstract class StateBase<AI> {
   protected final IEventSink eventSink;
…
}
```

Это обязывает специально разрабатывать классы состояний с учетом этих требований и лишает пользователя возможности использовать в качестве классов состояний классы, изначально не предусмотренные для этого, например классы стандартных библиотек. Отсутствие в классе контекста членов событий делает невозможным использование его в качестве класса состояния, а значит и не возможным использование автомата в качестве состояния другого автомата. Это значит, что невозможно иметь один автомат вложенным в другой.

2. Реализация state# (S#)

В этой работе предлагается реализация State# – реализация паттерна State на языке C#. Она снимает ограничения на классы состояния. На рис. 2 изображена структура реализации State#.

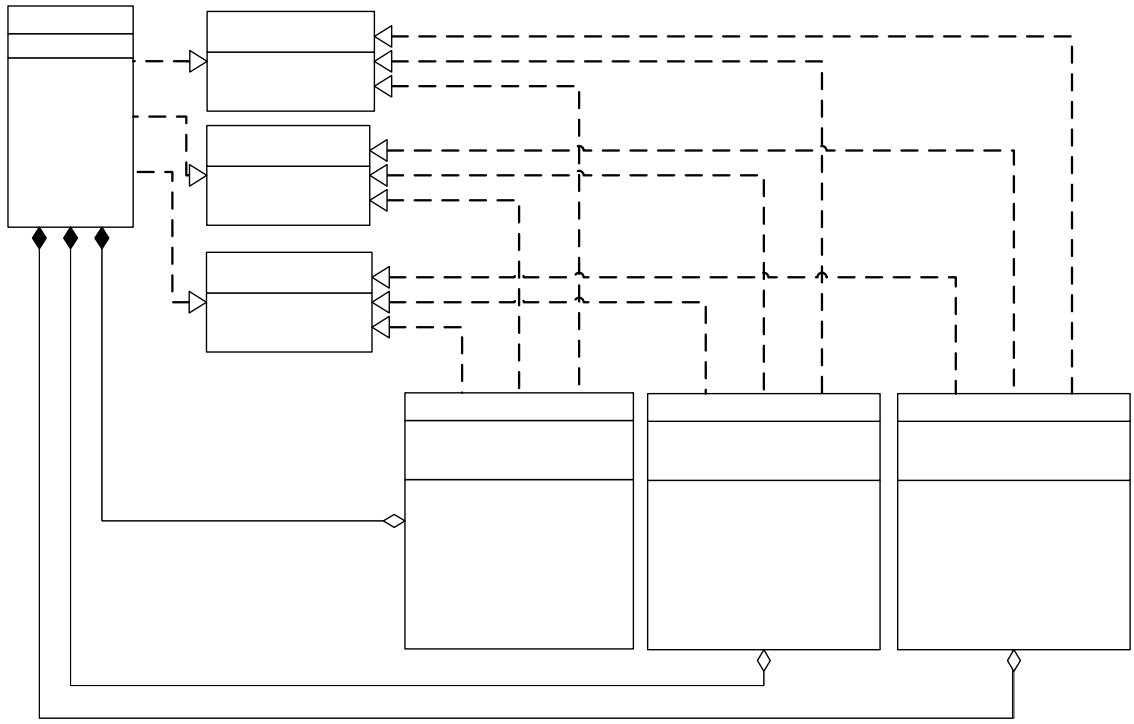


Рис. 2. Структура реализации State#

2.1. События

В качестве средства устранения недостатков, описанных в предыдущей главе, был выбран язык C# [8]. Этот язык содержит понятие событие на синтаксическом уровне. События языка C# и предлагается использовать в качестве событий, которые порождаются состояниями для извещения контекста.

```
public class State1 {
   public event EventHandler SomethingHappened;
…
}
public class Context {
   public Context() {
      state1.SomethingHappened += new EventHan-
dler(state1_SomethingHappened);
      …
   }
```

Context

-state : object

+operation1_1()
+operation1_2()

«interface»
IAutomatonInter

+operation1_1()
+operation1_2()

```
   pulic void state1_SomethingHappened(object sender, EventArgs e) {
      CurrentState = state2;
   }
}
```

Таким образом, мы избавляемся от использования объектов класса `Event`, и переходим к использованию синтаксически определенных в языке C# событий. События объявлены во многих библиотечных классах. Они являются стандартизованным механизмом оповещения объектом о какой-либо ситуации. Это позволяет использовать библиотечные классы в качестве классов состояний.

Также события могут присутствовать и в классе контекста, а значит, возможно, использование класса контекста в качестве класса состояния для другого контекста. Это позволяет реализовать вложенные автоматы, когда какое-либо состояние объемлющего автомата состоит из нескольких подсостояний вложенного.

```
public class Context1 {
   private State1 state1;
   private State2 state2;
   private State3 state3;
   …
}
public class Context2 {
   private State4 state1;
   private Context1 state2;
}
```

Здесь автомат `Context1` состоит из трех состояний: `State1`, `State2` и `State3`. А автомат `Context2` состоит из двух состояний: `State4` и `Context1`. Состояние `Context1`, так как в свою очередь является автоматом, состоит из трех подсостояний: `State1`, `State2`, `State3`.

2.2. Автоматный интерфейс

Автоматным интерфейсом назовем тот интерфейс, реализация методов которого зависит от состояния автомата. Его методы реализуются в классе автомата делегированием вызова соответствующего метода у текущего состояния (рис. 3).

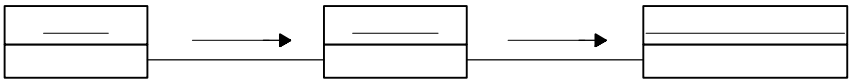


Рис. 3. Делегирование операций текущему состоянию автомата

Все классы состояний должны реализовывать все автоматные интерфейсы. Однако это требование не распространяется на неавтоматные интерфейсы. Класс состояния может и не реализовывать некий неавтоматный интерфейс, который реализуется автоматом.

В паттерне State Machine предполагалось, что автомат реализует один автоматный интерфейс. Это ограничение несколько искусственно. Предлагается использовать произвольное количество автоматных интерфейсов. Это может быть обосновано при использовании уже готовых, библиотечных интерфейсов. Единственное требование – каждый класс состояния автомата должен реализовывать все автоматные интерфейсы. При этом и класс контекста, и классы состояний могут реализовывать произвольное количество неавтоматных интерфейсов.

2.3. Действия на переходах

Switch-технология [4] предлагает задавать действия, которые выполняются при переходе автомата из одного состояния в другое. State# сохраняет эту возможность. Действия на переходе могут быть заданы в обработчике события перехода.

```
public class Context {
  public Context() {
    state1.SomethingHappened += new EventHan-
dler(state1_SomethingHappened);
    …
  }
  public void state1_SomethingHappened(object sender, EventArgs e) {
    // Действия на переходе
    CurrentState = state2;
  }
}
```

2.4. Сравнение реализации State Machine на языках Java и C#

Отсутствие встроенных в язык Java событий вынуждает к определению класса `Event` и добавлению в классы состояний экземпляров этого класса. Также класс состояния вынужден имплементировать интерфейс `IEventSink` для нотификации контекста о смене состояния.

Встроенные в язык C# события избавляют от необходимости в событиях-объектах и реализации интерфейса.

В реализации на языке Java обработка событий происходила в одном методе `castEvent`, который был методом интерфейса `IEventSink`. В этом методе следующее состояние определяется либо с использованием хэш-таблиц, либо при помощи оператора *switch*. Первое решение не позволяет задавать действия на переходах и проверять дополнительные условия, а второе делает код метода громоздким.

Реализация на языке C# использует отдельный метод для каждого перехода. Этот метод – обработчик события состояния. В него можно добавить проверку дополнительных условий для перехода и действия на переходе, код, который должен выполняться при переходе.

3. Язык State# (S#)

Для эффективной реализации подхода, изложенного в главе 2, предлагается расширение языка C#. Новый расширенный язык будет надмножеством языка C#, из чего следует, что ему будут принадлежать все программы, написанные на языке C#.

Введение в язык дополнительных синтаксических конструкций упрощает реализацию на нем шаблонных решений, паттернов.

Язык C# был разработан компанией Microsoft как язык для их новой платформы Microsoft .Net Framework. C# – это объектно-ориентированный язык высокого уровня. Как и многие современные языки C# – си-подобный язык. Однако для большей гибкости C# содержит такие дополнительные элементы, как свойства, делегаты и события.

С помощью событий языка объект сообщает всем желающим о каком-нибудь событии, произошедшим с ним. Событие вызывает код подписчиков, которые тем или иным образом реагируют на него.

Именно события позволяют нам достичь поставленной задачи. Любой класс, имеющий события и реализующий некий интерфейс, можно трактовать как состояние. Тогда другой класс (контекст), подписываясь на эти события, сможет сосредоточить логику переходов между состояниями.

3.1. Класс автомата

Для описания класса автомата в язык вводится новое ключевое слово *automaton* (англ. автомат). Оно используется вместо ключевого слова *class*, которое применяется в языке C# и обозначает классы специального вида, классы автоматов.

```
public automaton MyAutomaton
```

Автомат может содержать все члены, которые может содержать класс языка C#:

- поля;
- методы;
- свойства;
- индексаторы;
- перегрузки операторов;
- события.

3.2. Наследование и реализация интерфейсов

Также как класс в языке C#, в языке State# автомат может наследовать один класс. В State# базовый класс может быть обычным классом или автоматом. Однако класс автомата должен реализовывать как минимум один автоматный интерфейс и произвольное количество обычных интерфейсов. Перед автоматными интерфейсами ставится ключевое слово *auto*.

```
public automaton MyAutomaton : auto I1, I2, auto I3 {
…
}
```

В этом примере объявляется класс автомата `MyAutomaton`, который реализует автоматные интерфейсы `I1`, `I3` и обычный интерфейс `I2`.

Для всех состояний, включенных в автомат, классы этих состояний должны реализовывать все автоматные интерфейсы. При этом классы состояний могут реализовывать и другие интерфейсы дополнительно.

Для каждого метода из каждого автоматного интерфейса в классе автомата создается неявная реализация, которая вызывает соответствующий метод у текущего состояния. Поэтому явно реализовывать методы из автоматных интерфейсов в классе автомата не нужно.

3.3. Состояния

Состояниями в классе контекста будут считаться члены-поля, объявленные с использованием ключевого слова *state* в начале объявления. Класс членов-состояний может быть любым, в том числе классом другого контекста.

```
public automaton MyAutomaton : auto I1, I2, auto I3 {
   state MyStateClass1 state1;
   …
}
```

В работе [7] для обозначения стартового состояния оно помещалось первым в класс контекста. Однако такая семантика нетипична для таких объектно-ориентированных языков, как Java и C#. В них не важен порядок объявления членов класса. Чтобы избежать нетипичной семантики, предлагается для стартового состояния использовать дополнительное ключевое слово *initial*.

```
public automaton MyAutomaton : auto I1, I2, auto I3 {
   state MyStateClass1 state1;
   initial state MyStateClass2 state2;
   state MyStateClass3 state3;
   …
}
```

После объявления состояния в фигурных скобках помещается информация о переходах из этого состояния.

Каждый переход начинается с ключевого слова *transition*. Затем идет ключевое слово *to,* за которым указывается имя члена-состояния, в которое осуществляется переход. Затем – ключевое слово *when*, после которого – имя публичного события, которое провоцирует переход и присутствует в классе состояния.

Затем идет опциональное ключевое слово *incaseof*, после которого в круглых скобках можно указать булево выражение, являющееся условием перехода.

В конце можно указать действия, совершаемые на переходе. Для этого они указываются в фигурных скобках после ключевого слова *actions*.

```
public automaton MyAutomaton : auto I1, I2, auto I3 {
   state MyStateClass1 state1
{
   transition
   to state2
   when Happened1

   transition
   to state3
   when Happened2
};
   initial state MyStateClass2 state2
{
   transition
   to state1
   when SomethingHappened
   incaseof (someBooleanExpression)
   actions
   {
      Console.Writeln("!");
   }
};
   state MyStateClass3 state3;
   …
}
```

3.4. Автоматическое протоколирование

Одним из принципов SWITCH-технологии является протоколирование работы автомата. Протоколирование позволяет отловить большое количество ошибок. Поэтому в язык State# добавлена возможность автоматического протоколирования работы автомата.

Включение или выключение протоколирования не должно происходить в коде программы, а должно управляться извне.

Протоколирование включается в конфигурационном файле проекта. Он представляет собой xml-документ [12]. Протоколирование включается путем добавления в него секции `automatonLog` с атрибутом `Enabled` со значением `true`. Далее нужно включить протоколирование непосредственно для экземпляра автомата. Это делается добавлением в секцию `automatonLog` тэга `automatonLogInstance` со следующими параметрами:

- `Name` – имя протоколируемого автомата для ссылки на него;
- `Enabled` – `true` или `false` – включено или выключено протоколирование;
- `File` – путь к файлу, в который будет вестись протоколирование.

Чтобы связать экземпляр автомата с записью в конфигурационном файле, ему необходимо задать имя. Для этого используется конструктор с именем в качестве первого параметра. Такие конструкторы будут созданы автоматически для каждого конструктора, созданного пользователем. Если автомат будет создан без передачи имени в конструкторе, то имя будет назначено автоматически в формате:
<имя_класса_автомата>_<номер экземпляра>

```
MyAutomaton myAutomaton = new MyAutomaton(”MyAutomaton1”);
```

В результате в файле, заданном в конфигурационном файле для автомата с именем MyAutomaton1, будет сохранена вся информация о переходах автомата из одного состояния в другое.

3.5. Извещение о смене состояния

Кроме автоматического протоколирования, пользователю предоставляется возможность гибко настраивать реакцию системы на смену состояния автомата. Для этого каждый класс контекста предоставляет событие `StateChanged` типа `StateChangedEventHandler`, которое происходит каждый раз при смене состояния.

```
delegate void StateChangedEventHandler (object sender, StateChangedEven-
tArgs e)
```

В параметрах события `sender` представляет собой объект контекста. Параметр e содержит два открытых свойства `OldState` и `NewState` типа `string`, возвращающих имена членов состояний в классе контекста.

3.6. Сравнение языков State Machine и State#

Язык State Machine [7] является развитием языка Java и транслируется в Java. Язык State# является развитием языка C# и транслируется в C#. Язык State# не определяет синтаксиса для определения классов состояний, так как для реализации паттерна State Machine на языке State# в качестве классов состояний используются обычные классы.

Рассмотрим недостатки языка State Machine:

1. класс автомата должен реализовывать ровно один интерфейс, который и считается автоматным;
2. инициализация объектов состояний в конструкторе автомата описывается с помощью нового оператора `@=` вместо оператора `=;`
3. стартовым считается состояние, описанное в автомате первым. Описание состояний в автомате похоже на описание членов класса. Наделять семантикой порядок определения членов несколько нетрадиционно.

Язык State# устраняет эти недостатки, а также предоставляет дополнительные возможности:

1. возможность задавать булевские условия перехода;
2. возможность задавать действия на переходах;
3. генерация события о смене состояния;
4. настраиваемое протоколирование.

Заключение

В настоящей работе рассмотрен паттерн проектирования State Machine, являющийся развитием паттерна State.

Предлагается реализация паттерна State Machine, основанная на использовании событий языка C# как средства взаимодействия классов состояний и класса контекста. Это позволяет использовать стандартные библиотечные классы в качестве классов состояний. Использование событий языка может существенно ускорить разработку, так как позволит использовать уже готовые решения.

В рамках настоящей работы предложено расширение языка C# (State#) для более удобной реализации этого паттерна. Предложен синтаксис описания автоматов и разработан транслятор с языка State# на язык C#.

Транслятор поддерживает использование протоколирования. Протоколирование может быть полезно как средство отлаживания и тестирования. Управление протоколированием производится без перекомпиляции проекта, в конфигурационном файле. Кроме того, в автомат транслятором добавляется событие о смене состояния, которое может быть использовано извне.

В качестве расширения данной работы может быть рассмотрено написание модуля для Microsoft Visual Studio для автоматической трансляции программ на языке State#, написание транслятора с языка C# 2.0 или C# 3.0 и трансляция непосредственно в Microsoft Intermediate Language.

Литература

1. Adamczyk P. The Antology of the Finite State Machine Design Patterns. // http://jerry.cs.uiuc.edu/~plop/plop2003/Paperes/Adamczyk-State-Machine.pdf
2. Adamczyk P. Selected Patterns for Implementing Finite State Machines. // http://pinky.cs.uiic.edu/~padamczy/fsm
3. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. СПб: Питер, 2001.
4. Шалыто А.А. Switch-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998.
5. Николаенко А. Static Finite State Machime. // RSDN Magazine. 2005. № 3.
6. Шамгунов Н.Н., Корнеев Г.А., Шалыто А.А. State Machine – новый паттерн объектно-ориентированного проектирования. // Информационно-управляющие системы. 2004. № 5.
7. Шамгунов Н.Н., Корнеев Г.А., Шалыто А.А. State Machine – расширение языка Java для эффективной реализации автоматов. // Информационно-управляющие системы. 2005. № 1.
8. Троелсен Э. C# и платформа .NET. СПб.: Питер, 2002.
9. Либерти Д. Программирование на C#. СПб-Москва: Символ, 2003.
10. Microsoft Developer Network // http://msdn.microsoft.com
11. ANTLR // http://www.antlr.org
12. XML Specification // http://www.w3.org/XML/
13. XML Schema // http://www.w3.org/XML/Schema/
14. Рихтер Д. Программирование на платформе Microsoft .Net Framework. СПб: Питер, 2005.

ПРОГРАММНЫЙ МОДУЛЬ СПЕКТРАЛЬНОГО АНАЛИЗА, ОРИЕНТИРОВАННЫЙ НА РАБОТУ В РЕЖИМЕ РЕАЛЬНОГО ВРЕМЕНИ

В.Э. Базаревский, Н.В. Степанчук
(Белорусский государственный университет информатики и радиоэлектроники)
Научный руководитель – к.т.н., доцент П.Ю. Бранцевич
(Белорусский государственный университет информатики и радиоэлектроники)

Рассматриваются способы создания модуля спектрального анализа для исследования виброметрических данных в системах реального времени. Предложен способ настройки модуля на работу в реальном режиме времени. Рассмотрены способы ускорения преобразования исходного сигнала в спектр и его отображения.

Введение

Контроль за состоянием различных объектов по параметрам вибрации осуществляются с помощью программно-аппаратных комплексов, реализующих сбор, обработку и отображение вибрационных данных.

При проектировании программной части аппаратно-программных комплексов реального времени (АПКРВ) обработки сигналов, реализующих одновременный сбор и отображение результатов, возникает проблема обеспечения обработки исходных данных в режиме реального времени. Реализация таких систем требует решения ряда задач. Прежде всего, это выбор базовой операционной системы. Наиболее используемой на сегодняшний день является операционная система Windows (здесь и далее подразумеваются версии операционной системы Windows из семейства Windows NT, в частности, Windows NT, Windows 2000, Windows XP). Для ОС Windows реализовано большое количество систем разработки, соответственно, что немаловажно, существует большое количество специалистов, способных работать и/или разрабатывать различные программы для этой ОС. Значимым фактором является также ее относительно невысокая стоимость и высокая доступность и распространенность. Однако операционная система Windows не является операционной системой реального времени и не соответствует требованиям, предъявляемым к системам реального времени. Эти требования изложены в стандарте POSIX 1003.4 рабочего комитета IEEE [1–5]. Стандарт определяет ОС как систему реального времени, если она обеспечивает требуемый уровень сервиса за вполне определенное, ограниченное время.

Построение АПКРВ вибрационного контроля на базе операционной системы Windows требует ее специальной организации, неотъемлемой частью которой является требование максимально быстрой обработки данных. Таким образом, для обработки данных должен быть построен модуль, который обеспечивает обработку данных, не допускает возникновения ситуаций потери сигнальных данных и легко встраиваем в программные средства.

Теоретическая часть

Для вычисления спектра применяется алгоритм дискретного преобразования Фурье (ДПФ).

$$x(k) = \sum_{j=0}^{N-1} X(j) \cdot e^{i\frac{2\pi \cdot j \cdot k}{N}} = \sum_{j=0}^{N-1} X(j) \cdot W_N^{-jk} \ ,$$

где

$$X(j)=\frac{1}{N}\sum_{k=0}^{N-1}x(k)\cdot e^{-i\frac{2\pi\cdot j\cdot k}{N}}=\frac{1}{N}\sum_{k=0}^{N-1}x(k)\cdot W_N^{jk},\ W_N^{r}=e^{-i\frac{2\pi}{N}r}.$$

В основе алгоритмов, обеспечивающих снижение трудоемкости вычисления ДПФ, лежат следующие соотношения:

$$W_N^{k\cdot n}=W_N^{k\cdot(n+N)}=W_N^{(k+N)\cdot n};\qquad W_N^{k\cdot(N-n)}=W_N^{-k\cdot n}.$$

Алгоритмы, в которых исходная, размерности N, дискретная последовательность, представляющая сигнал, разбивается на меньшие последовательности, называются алгоритмами с прореживанием по времени. Сущность такого преобразования выглядит следующим образом:

$$X(j)=\frac{1}{N}\sum_{k=0}^{N-1}x(k)\cdot W_N^{jk}=\frac{1}{N}\sum_{l=0}^{\frac{N}{2}-1}x(2l)\cdot W_{\frac{N}{2}}^{l\cdot j}+\frac{1}{N}W_N^{j}\sum_{l=0}^{\frac{N}{2}-1}x(2l+1)\cdot W_{\frac{N}{2}}^{l\cdot j}.$$

Такую реализацию называют алгоритмом быстрого преобразования Фурье (БПФ) Кули-Тьюки с прореживанием по времени. Она состоит из последовательности ступеней, на каждой из которых выполняются однотипные действия над массивом комплексных данных, поступающим на вход ступени, а результаты, полученные в ходе преобразований, выполненных на текущей ступени, обратно перезаписываются в исходный массив, который является выходным для текущей ступени и входным для следующей ступени.

Основной вычислительной процедурой этого алгоритма является так называемая операция «бабочка», которая в комплексном виде представляется как

$$\begin{aligned}X_{m+1}(p)&=X_m(p)+W_N^r\cdot X_m(q)\\X_{m+1}(q)&=X_m(p)-W_N^r\cdot X_m(q)\end{aligned},\qquad(1)$$

где m – номер ступени преобразования исходного массива данных.

Для реализации алгоритма БПФ требуется $\log_2 N$ ступеней, а на каждой ступени выполняется $N/2$ операций «бабочка». Таким образом, трудоемкость всего алгоритма – $(N\cdot\log_2 N)/2$ операций «бабочка».

Можно заметить, что на первой, второй и третьей ступенях БПФ реализация операции «бабочка» значительно упрощается. Так, для «бабочек» с индексом, равным нулю, действительные представления выражений (1) принимают вид (первая модификация):

$$\begin{aligned}\mathrm{Re}[X_{m+1}(p)]&=\mathrm{Re}[X_m(p)]+\mathrm{Re}[X_m(q)];\\\mathrm{Im}[X_{m+1}(p)]&=\mathrm{Im}[X_m(p)]+\mathrm{Im}[X_m(q)];\\\mathrm{Re}[X_{m+1}(q)]&=\mathrm{Re}[X_m(p)]-\mathrm{Re}[X_m(q)];\\\mathrm{Im}[X_{m+1}(q)]&=\mathrm{Im}[X_m(p)]-\mathrm{Im}[X_m(q)],\end{aligned}$$

для «бабочек» с индексом $N/4$ (вторая модификация) – вид

$$\begin{aligned}\mathrm{Re}[X_{m+1}(p)]&=\mathrm{Re}[X_m(p)]-\mathrm{Im}[X_m(q)];\\\mathrm{Im}[X_{m+1}(p)]&=\mathrm{Im}[X_m(p)]+\mathrm{Re}[X_m(q)];\\\mathrm{Re}[X_{m+1}(q)]&=\mathrm{Re}[X_m(p)]+\mathrm{Im}[X_m(q)];\\\mathrm{Im}[X_{m+1}(q)]&=\mathrm{Im}[X_m(p)]-\mathrm{Re}[X_m(q)],\end{aligned}$$

для «бабочек» с индексом N/8 (третья модификация) – вид

$$\begin{aligned}\mathrm{Re}[X_{m+1}(p)]&=\mathrm{Re}[X_m(p)]+C;\\\mathrm{Im}[X_{m+1}(p)]&=\mathrm{Im}[X_m(p)]+D;\\\mathrm{Re}[X_{m+1}(q)]&=\mathrm{Re}[X_m(p)]-C;\\\mathrm{Im}[X_{m+1}(q)]&=\mathrm{Im}[X_m(p)]-D,\end{aligned}$$

где

$$C = \left(\mathrm{Re}\left[X_m(q)\right] - \mathrm{Im}\left[X_m(q)\right]\right)\cdot\sqrt{2};$$
$$D = \left(\mathrm{Im}\left[X_m(q)\right] + \mathrm{Re}\left[X_m(q)\right]\right),$$

а для «бабочек» с индексом 3*N*/8 (четвертая модификация) – вид

$$\mathrm{Re}\left[X_{m+1}(p)\right] = \mathrm{Re}\left[X_m(p)\right] - E;$$
$$\mathrm{Im}\left[X_{m+1}(p)\right] = \mathrm{Im}\left[X_m(p)\right] - F;$$
$$\mathrm{Re}\left[X_{m+1}(q)\right] = \mathrm{Re}\left[X_m(p)\right] + E;$$
$$\mathrm{Im}\left[X_{m+1}(q)\right] = \mathrm{Im}\left[X_m(p)\right] + F,$$

где

$$E = \left(\mathrm{Re}\left[X_m(q)\right] + \mathrm{Im}\left[X_m(q)\right]\right)\cdot\sqrt{2};$$
$$F = \left(\mathrm{Im}\left[X_m(q)\right] - \mathrm{Re}\left[X_m(q)\right]\right).$$

На первой ступени БПФ требуется выполнить *N*/2 «бабочек» первой модификации (2*N* операций сложения), на второй ступени по *N*/4 «бабочек» первой и второй модификаций (2*N* операций сложения), на третьей ступени по *N*/8 «бабочек» первой, второй, третьей и четвертой модификации (5*N*/2 операций сложения и *N*/2 операций умножения). На каждой следующей ступени имеется соответственно по *N*/16, *N*/32, ... , 2 «бабочек» каждой из модификаций.

Следовательно, для реализации модифицированного алгоритма БПФ требуется выполнить $N\cdot\left(2\ \log_2 N - 7\right) + 12$ операций умножения и $3N\cdot\left(2\log_2 N - 1\right) + 4$ операций сложения.

Архитектура модуля

Модуль представляет собой компонент, написанный на языке C# для платформы .Net Framework.

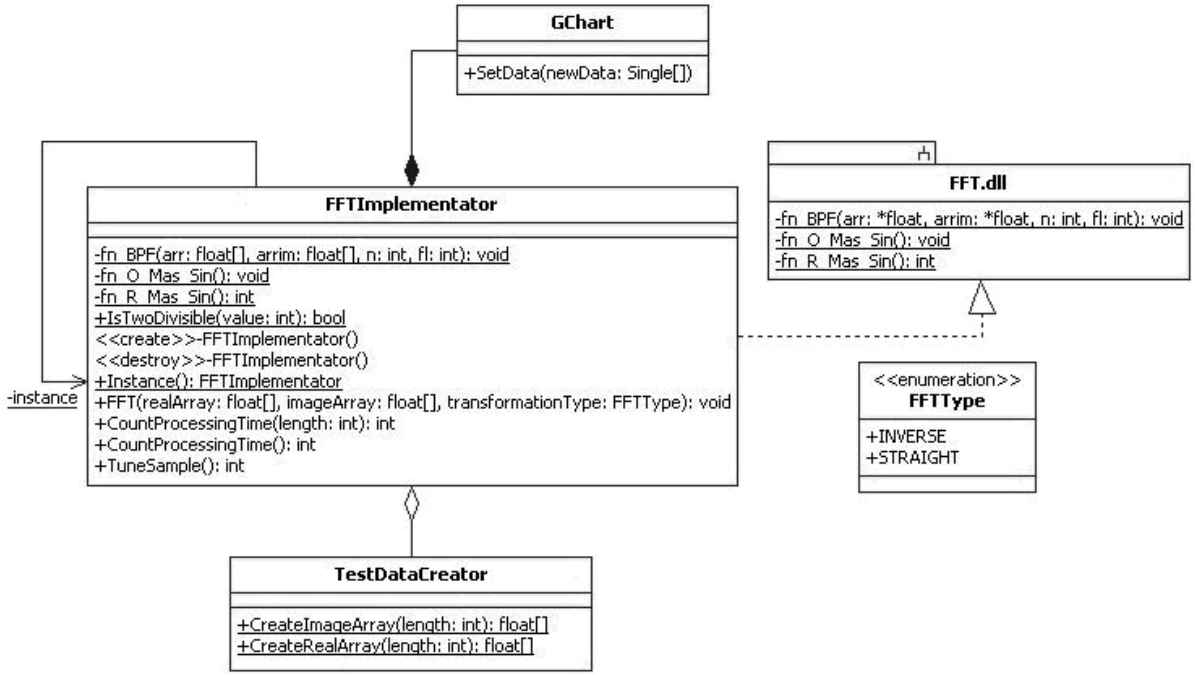


Рис. 1. Диаграмма модуля спектрального анализа

На рис. 1 изображена UML-диаграмма модуля спектрального анализа. Он состоит из следующих элементов.

FFT.dll – динамически подключаемая библиотека, реализующая создание/удаление виртуальной таблицы значений синуса, функцию БПФ. Для увеличения диапазона частот дискретизации подаваемого сигнала необходимо максимально ускорить преобразование исходного сигнала в спектр и отображение результатов. Предлагается реализация алгоритма Кули-Тьюки быстрого преобразования Фурье средствами математического сопроцессора современных ЭВМ. Выигрыш в скорости выполнения алгоритма от использования математического сопроцессора объясняется сокращением числа пересылок данных в памяти, что существенно уменьшает время работы алгоритма.

FFTType – перечисление, характеризующее тип БПФ: прямое или обратное.

GChart – компонент, отвечающий за отображение данных на экране пользователя. В нем установлены флаги двойной буферизации, что уменьшает мерцание отображаемой информации и, тем самым, улучшает практичность (usability) программного средства.

TestDataCreator – класс, реализующий функции создания тестовых данных для процедуры настройки модуля.

FFTImplementator – класс-одиночка (Singleton), оборачивающий функции DLL-библиотеки FFT.dll (так как программный модуль написан на языке C# под платформу .Net Framework, использующую промежуточный код, то непосредственное использование в программе ассемблерных вставок, которые позволяют работать с математическим сопроцессором, невозможно. Для преодоления данной сложности предлагается использование динамически подключаемых библиотек, написанных на языке C++, в котором существует возможность использования ассемблерных вставок). Данный класс расширяет возможности, предоставляемые библиотекой FFT.dll, функциями расчета среднего времени преобразования тестовых данных, проверки возможности работы модуля при заданных длине выборки и частоте дискретизации поступающих данных.

Описание процедуры настройки модуля

Время, затрачиваемее модулем на обработку и отображение выборки размером N, определяется как

$$t_м = t_{обр} + t_{отобр},$$

где $t_{обр}$ – время преобразования данных; $t_{отобр}$ – время отображения данных. N выбирается кратным степени двойки.

За время одного цикла работы модуля может быть принято n дискретных отсчетов, их число рассчитывается в соответствии с формулой:

$$n = f_{дискр} \cdot t_м,$$

где n – число отсчетов, через которые будут подаваться новые данные на обработку; $t_м$ – время, затрачиваемое модулем на обработку и отображение; $f_{дискр}$ – частота дискретизации аналогового сигнала.

Возможна ситуация, когда за время обработки поступит данных больше, чем размер обрабатываемой выборки N, что приведет к потере части дискретных отсчетов. Для корректного разрешения такой ситуации в модуле должны быть предусмотрены средства, определяющие возможность работы без потери данных в режиме реального времени.

Сценарий работы процедуры определения условий работы без потерь – следующий. Существует максимально допустимый размер выборки N, который выбирается исходя из условия решаемой задачи. Этой выборке соответствует интервал времени $t_м$, за который поступит n дискретных отсчетов, если $N<n$, то модуль может работать в реальном режиме времени с такими N и $f_{дискр}$, иначе N уменьшается вдвое, так как N кратно степени двойки, и должны быть заново рассчитаны $t_м$ и n. Эти действия повто-

ряются до тех пор, пока размер *N* не будет обеспечивать работу модуля в реальном режиме времени. Схема алгоритма настройки размера выборки изображена на рис. 2.

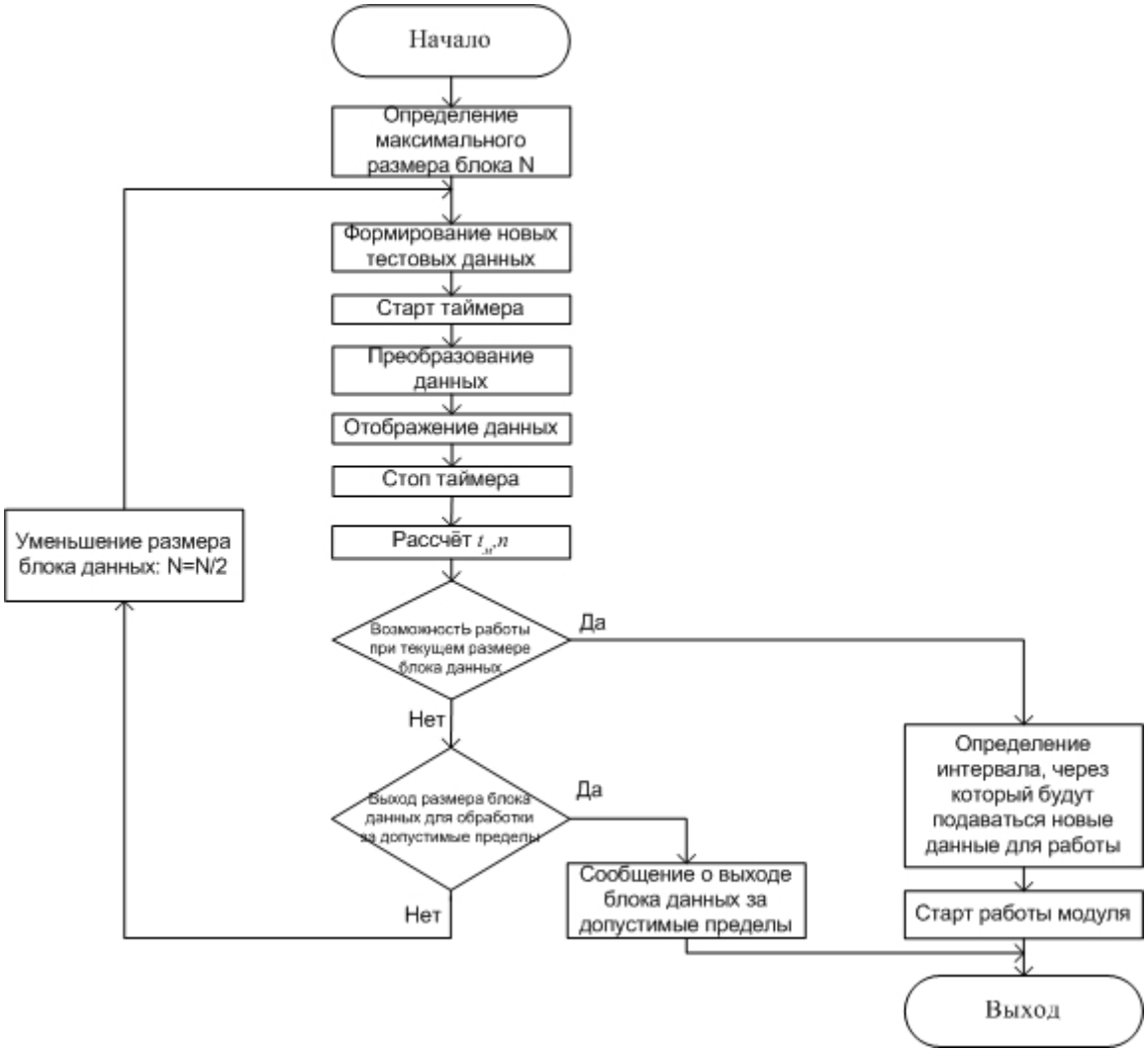


Рис. 2. Схема алгоритма процедуры настройки размера выборки для преобразования

Результаты испытаний модуля

Размер выборки	Среднее время обработки сигнала		Число дискретных отчётов, поступающих за время преобразования при заданной частоте дискретизации			
	тиков	мс	1 кГц	10 кГц	100 кГц	1 МГц
1024	860	240	0	2	27	278
2048	1840	513	0	5	60	570
4096	4160	1161	1	11	114	1154
8192	8550	2385	2	24	248	2450

Таблица. Результаты испытаний модуля

В таблице представлены характеристики работы модуля при разной длине выборки и разной частоте дискретизации подаваемого сигнала. Из предложенных данных видно, что характеристики модуля позволяют ему работать в системах вибродиагностики (при проведении опытов использовался АЦП фирмы LCARD E14-440 с максимальной частотой дискретизации 400 кГц).

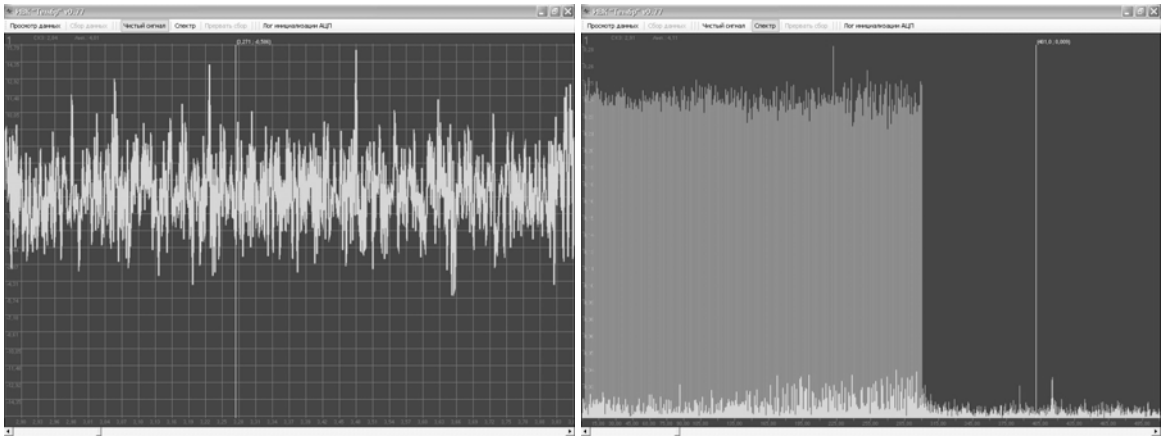


Рис. 3. Пример работы модуля в составе измерительно-вычислительного комплекса «Тембр»

Заключение

Создание дополнительных средств автоматического контроля возможности работы модуля с заданными характеристиками позволяет эффективно отслеживать возможные потери части поступающего сигнала. А синтез концепции объектно-ориентированного проектирования и использования низкоуровневых средств современных ЭВМ позволяет создать более производительное программное обеспечение без потерь во встраиваемости и сложности понимания архитектуры сторонними разработчиками.

Литература

1. Harbour M. Real-Time POSIX: An Overview / Сборник трудов международной конференции Vvconcx 93, июнь 1993. Москва.
2. IEEE Standards Project P1003.4 POSIX Part 1: System Application Program Interface (API) / Amendment 1: Rcaltimc Extension. Draft 13.-IEEE, 1992.
3. IEEE Standards Project P1003.4a Thread Extension for Portable Operating Systems. Draft 6.-IEEE, 1992.
4. EEE Standards Project P1003.4b POSIX Part 1: Rcaltimc System API Extension. Draft 6.- IEEE, 1993.
5. IEEE Standards Project P1003.13 Part 1: POSIX Rcaltimc A)-plication Support (AEP). Draft 5.- IEEE, 1992.
6. Бранцевич П.Ю., Коростель С.В., Носко Д.В. Алгоритмы и программные средства системы оценки технического состояния механизмов по вибрационным параметрам // Известия Белорусской инженерной академии. 2002. № 1(13)2. С. 36–38.
7. Бранцевич П.Ю., Костюк С.Ф., Ероховец И.Е., Носко Д.В. Проведение исследований по созданию вибродиагностической системы определения качества изготовления и сборки узлов автомобилей БелАЗ. Мн., 2005. 99 с.: Деп. в ГУ «БелИСА» 28.02.06, № Д20067.
8. Носко Д.В. Реализация программного обеспечения для обработки вибрационных сигналов на базе платформы .NET // Актуальные проблемы радиоэлектроники: научные исследования, подготовка кадров: сб. статей междунар. науч.-практ. конф. / Минский гос. высший радиотехнический колледж Минск, 2005. Ч. 2. С. 148–151.
9. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. СПб: Питер, 2001. 368 с.
10. Оппенгейм А.В., Шафер Р.В. Цифровая обработка сигналов. М.: Связь, 1979. 416 с.
11. Голд Б., Рэйдер Ч. Цифровая обработка сигналов. М.: Сов. радио, 1973. 368 с.

ПРИМЕНЕНИЕ АВТОРЕГРЕССИОННОГО ИНТЕГРИРОВАННОГО СКОЛЬЗЯЩЕГО СРЕДНЕГО В АЛГОРИТМАХ УПРАВЛЕНИЯ ПЕРЕГРУЗКАМИ ПРОТОКОЛОВ ПЕРЕДАЧИ ПОТОКОВЫХ ДАННЫХ

М.Ю. Будько, М.Б. Будько, А.В. Гирик
Научный руководитель – к.т.н., доцент Г.П. Жигулин

Настоящая статья посвящена вопросам применения краткосрочного прогнозирования в алгоритмах управления перегрузками протоколов передачи потоковых данных (таких, как RTP). Рассмотрены основные причины необходимости управления перегрузками в сетях без обеспечения качества обслуживания и подходы к реализации алгоритмов управления перегрузками в транспортных протоколах. Предложен подход к прогнозированию транспортных метрик с помощью ARIMA-процессов.

Введение

Рост объемов передаваемых по сетям данных в настоящее время таков, что проблема нехватки ресурсов встает остро даже для «быстрых» локальных сетей [1]. Если нагрузка превышает (возможно, временно) ресурсы сети, то в сети появляются перегрузки. Есть два выхода из этого положения: увеличить ресурсы и сократить нагрузку. Поскольку увеличить ресурсы чаще всего невозможно, остается только сокращение нагрузки, что в случае передачи мультимедийных данных означает ухудшение качества данных, передаваемых всем или некоторым пользователям.

Передача аудио- и видеоинформации в IP-сетях сопряжена с рядом трудностей. IP-сеть не может гарантировать надежную доставку или производительность, так как сетевой протокол IP осуществляет передачу дейтаграмм по принципу *best effort*. Пакеты в ходе доставки могут быть задержаны, переупорядочены, отброшены (потеряны) или повреждены. Протоколы более высоких уровней, работающие на конечных узлах, могут обеспечивать определенные гарантии доставки. Кроме этого, управление перегрузками и регулирование интенсивности трафика также возлагается на конечные узлы. Приложения, генерирующие мультимедийный трафик, должны учитывать текущее состояние сети и регулировать интенсивность трафика таким образом, чтобы избежать перегрузок [2].

Способы минимизации задержки потерь применяются в IP-сетях

Какие же способы гарантии минимизации задержки, вариации задержки и потерь применяются в изначально ненадежных IP-сетях? Все они сводятся к резервированию ресурсов сетевого оборудования для отдельных потоков данных. В случае использования системы интегрированного обслуживания (*Integrated Services, IntServ*) гарантии качества (*Quality of Service, QoS*) предоставляются для сетевых соединений на протяжении всего маршрута от источника данных к получателю, а в системе дифференцированного обслуживания (*Differentiated Services, DiffServ*) гарантии качества в виде приоритетного обслуживания предоставляются для классов трафика. Достоинством первой системы является прямая поддержка идеологии сквозной доставки (*end-to-end*) [3], характерной для многих потоковых приложений, недостатком – повышенная нагрузка на маршрутизаторы операторов, вынужденных поддерживать сведения о большом количестве пользовательских потоков, которые должны быть обслужены с затребованным уровнем QoS, что же касается второй системы, то ее применение существенно снижает нагрузку на магистральные маршрутизаторы, но не обеспечивает сквозной гарантии QoS для потоков данных, так как приоритетное обслуживание классов трафика (и способы задания этих классов) в различном сетевом оборудовании могут существенно отличаться. В результате ни та, ни другая система широко не применяется, и, как прави-

ло, обеспечить сквозное качество обслуживания в IP-сети не удается. Необходимо защититься хотя бы от критических перебоев в работе сервисов. Таким образом, организация потокового вещания в сетях, изначально не приспособленных для этого (а таких сетей большинство), представляет собой непростую и актуальную задачу. Эта задача может быть решена путем внедрения системы управления передачей потоковых данных как компонентов операционной системы или компонентов приложений, осуществляющих трансляцию потоковых данных. Понятно, что система управления передачей потоковых данных не является панацеей и обеспечить выполнение гарантий QoS с ее помощью нельзя, однако можно добиться своевременной реакции на критические ситуации, прогнозирования нагрузки и возможных коллапсов, адаптации сервиса к изменившимся условиям в сети [4].

Каким образом приложение может судить о текущем состоянии сети? При наблюдении за системой оно может использовать такие метрики, как средняя длина очередей или количество отброшенных пакетов. Если речь идет о транспортных или прикладных протоколах, то чаще используются такие метрики, как время прохождения пакетов по сети, вариация этого времени и число повторно переданных пакетов.

Основная задача алгоритма управления перегрузками – вовремя обнаружить, что ресурсов системы становится недостаточно, и отреагировать на это, например, снижением интенсивности передачи данных, что, в свою очередь, может быть достигнуто, например, увеличением задержки между пакетами при отправке их в сеть. Алгоритмы с явной обратной связью основаны на том, что получатель информирует отправителя о перегрузке, например, с помощью посылки специального пакета. Алгоритмы с неявной обратной связью основаны на том, что источник сам определяет факт перегрузки на основе своих локальных наблюдений за трафиком, например, по величине задержки уведомления о доставке пакета. Необходимо отметить, что в большинстве реализаций не делается попыток «заглянуть» в будущее и определить время и характер перегрузки, чаще всего предусматривается непосредственная реакция на изменение той или иной метрики.

Типичным примером протокола, в котором используется управление перегрузками, является TCP – протокол надежной доставки, работающий поверх ненадежного IP. Средства управления перегрузками появились в TCP не сразу – различные механизмы добавлялись в разное время [5]. Для оценки времени двойного оборота (и установки для отправленного пакета таймера повторной передачи) используется разновидность скользящего среднего – экспоненциальное сглаживание

$$rtt_e = \alpha \cdot rtt_e + (1-\alpha) rtt_m,$$

$$rto = \beta \cdot rtt_e,$$

где rtt_e – расчетное время двойного оборота (оценка), rtt_m – последнее измерение *rtt*, *rto* – расчетное значение периода времени для таймера повторной передачи.

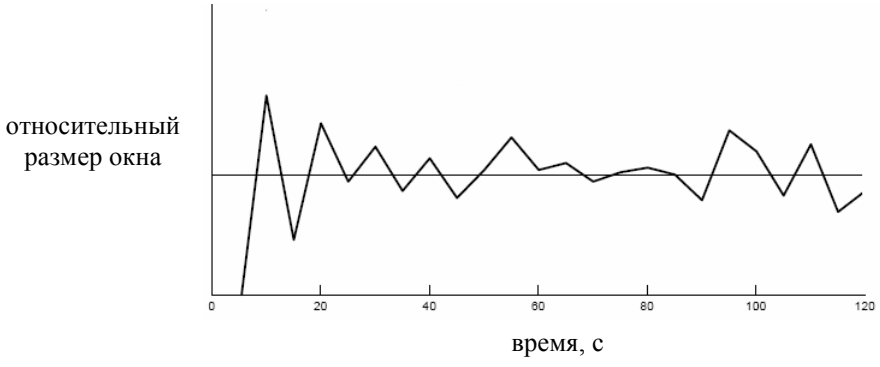


Рис. 1. Управление перегрузками AIMD (Additive Increase Multiplicative Decrease)

Так называемый «медленный старт» выполняется в начале передачи, когда скорость передачи растет экспоненциально. В тот момент, когда происходит потеря пакетов, соединение переводится в фазу предупреждения перегрузок: линейный рост скорости передачи (который обеспечивается увеличением на единицу размеров окна на каждый подтвержденный пакет) сочетается с уменьшением размера окна передачи в два раза при обнаружении потери. TCP-соединение «зондирует» доступную пропускную способность сети, каждый раз подходя к границе и отступая от нее. Результат такого поведения – характерный «пилообразный» трафик – представлен на рис. 1.

Есть несколько следствий подобного поведения. В случае, когда речь идет об эластичном трафике, удается значительно сэкономить ресурсы, сведя количество повторных передач к минимуму. Однако существуют и побочные эффекты: во-первых, при определенных условиях может наблюдаться явление, напоминающее «резонанс» множества TCP-источников, что отрицательно сказывается на функционировании сети [6]. Во-вторых, описанное поведение TCP-источников отрицательно сказывается на приложениях, передающих мультимедийный трафик и чувствительных к задержкам [4]. Протокол IP сам по себе не гарантирует своевременную доставку, наличие же в сети TCP трафика приводит к еще большей вариации задержки. При этом требование своевременной доставки для приложений реального времени (видео, аудио) является более важным, чем требование надежной доставки, так как с частичной потерей данных приложения реального времени, как правило, могут справиться.

В какой-то степени проблемы с временем доставки можно решить буферизацией данных в приемнике и использованием адаптивного кодирования (адаптивного к доступной пропускной способности соединения). Однако задержки, связанные с буферизаций, неприемлемы для многих интерактивных сервисов, например, видеотелефонии или видеоконференций, а кодеки, которые бы реализовывали принципы адаптивного кодирования и которые предусматривали бы возможность управления и изменения параметров кодирования «на лету», пока что находятся в экспериментальной стадии. С другой стороны, использование таких кодеков (по одному процессу транскодирования на каждое клиентское соединение) потребовало бы огромного количества ресурсов.

Ясно, что для передачи потоковых данных TCP и его модификации не подходят. В большинстве случаев приложения используют UDP и его разновидности [7] (в том числе собственные протоколы, работающие поверх UDP). В настоящее время все чаще используется протокол RTP (*Real Time Protocol*) [4, 8]. Трудность использования RTP – в том, что приложения должны реализовывать алгоритм управления перегрузками самостоятельно. В некоторых случаях это неудобно – реализация управления перегрузками на транспортном уровне позволила бы значительно упростить логику передачи потоковых данных во многих приложениях. Протокол DCCP (*Datagram Congestion Control Protocol*) непосредственно не предназначен для передачи мультимедийных данных, скорее это замена UDP [9]. Разработчики DCCP предполагали, что протокол будет реализован в ядре ОС и это позволит использовать его в качестве основы для протоколов прикладного уровня. В DCCP возможен выбор алгоритма управления перегрузками на этапе установления соединения. Также предусмотрена возможность задания собственного алгоритма [10].

Алгоритм TFRC

Были предложены различные алгоритмы, наиболее зрелым из которых сообществом Internet был признан TFRC (*TCP Friendly Rate Control*). Благодаря использованию этого алгоритма удается сгладить всплески скорости передачи, характерные для TCP, при сохранении той же средней пропускной способности соединения [11]. TFRC опи-

рается на то, что для устойчивого состояния TCP отправителя максимальная скорость передачи данных обратно пропорциональна корню квадратному из вероятности потерь:

$$T \sim \frac{1}{\sqrt{p}},$$

$$T = \frac{S}{rtt\sqrt{\frac{2p}{3}} + rto(3\sqrt{\frac{3p}{8}})\,p(1+32p^2)}.$$

Алгоритм управления перегрузками накладывает ограничения на максимальную скорость передачи и устанавливает, в какие моменты времени скорость может изменяться как в большую, так и в меньшую сторону.

TFRC сохраняет требование выполнять медленный старт, что неудобно для приложений реального времени. Кодеки, которые продуцируют взрывной поток данных, также плохо «уживаются» с TFRC [11]. С другой стороны, несомненна польза от управления перегрузками: интеллектуальное приложение может изменить параметры кодирования видео- или аудиопотока таким образом, что качество ухудшится, но не так, как в случае, если данные будут отброшены сетью. Способ, которым достигается изменение качества транслируемого потока, может отличаться от приложения к приложению, но можно смело утверждать, что в схемах, в которых реализуется переключение между уровнями качества, о моментах изменения желательно знать заранее.

Кратковременное прогнозирование перегрузок для отдельного соединения

В ходе разработки приложения для передачи потокового видео с использованием специального профиля протокола RTP и оригинальной схемы изменения качества транслируемых потоков перед авторами встала задача реализации механизма кратковременного прогнозирования перегрузок для отдельного соединения. Ни один из известных алгоритмов управления перегрузками не выполняет экстраполяцию данных с целью прогнозирования возможных действий для высокоуровневых прикладных механизмов. Задача решается экстраполяцией транспортных и прикладных метрик на основе модели авторегрессионных скользящих средних. Анализу подвергаются данные, которые отправитель получает по протоколу RTCP, и SNMP статистика, собранная с сетевых устройств.

Традиционно такие модели прогнозирования, как ARMA и ARIMA, и основанные на них используются при анализе временных рядов в эконометрике. Они также хорошо зарекомендовали себя в области долгосрочного и среднесрочного прогнозирования трафика в магистральных сетях, в сетях предприятий [12] и т.д.

Любое наблюдение процесса авторегресии и скользящего среднего состоит из линейной функции от предыдущего наблюдения плюс независимый случайный шум минус некоторая доля предыдущего случайного шума. Прогнозирование следующего наблюдения выполняется путем комбинирования прогнозируемого значения из оценки уравнения авторегрессии с оценкой текущего случайного шума:

$$Y_t = \delta + \varphi Y_{t-1} + \varepsilon_t - \theta\varepsilon_{t-1}.$$

Как правило, рассматриваемые процессы проявляют нестабильность [13]. Чистый интегрированный процесс (также называемый случайным блужданием) имеет тенденцию все дальше уходить от точки, в которой он находился. Нельзя рассчитывать на то, что с течением времени значение метрики будет оставаться достаточно близким к какому-либо долгосрочному среднему значению. Стационарным ARMA-процессом может быть процесс изменения или разности ряда. В этом случае сам ряд соответствует процессу авторегрессионного интегрированного скользящего среднего.

Основные трудности прогнозирования на основе ARIMA-процессов заключаются в подборе полинома и быстром расхождении доверительного интервала в области экстраполяции.

Заключение

Управление передачей потоковых данных в современных сетях без поддержки качества обслуживания является актуальной задачей, а выбор алгоритма управления перегрузками в большинстве случаев зависит от характеристик приложения. Прогнозирование перегрузок – это следующий шаг после реализации простой реакции на перегрузки. Перспективным подходом является кратковременное прогнозирование на основе ARIMA процессов.

Литература

1. Шувалов В.П., Ярославцев А.Ф. Телекоммуникационные системы и сети. Т.3, М., 2003.
2. C.S. Perkins, L. Gharai, T. Lehman and A. Mankin. Experiments with delivery of HDTV over IP networks. / Proceedings of the 12th International Packet Video Workshop, (Pittsburgh, PA, USA), April 2002.
3. J.H. Saltzer, D.P. Reed and D.D. Clark. End-to-end arguments in system design. // ACM Transactions on Computer Systems. 2, November 1984.
4. C.S. Perkins. RTP: Audio and Video for the Internet. Addison Wesley, June 2003.
5. V. Jacobson, Congestion Avoidance and Control, SIGCOMM, 1988.
6. Платов В.В., Петров В.В. Исследование самоподобной структуры трафика беспроводной сети, http://www.teletrafic.ru
7. L-A. Larzon, M. Degermark, S. Pink, The Lightweight User Datagram Protocol (UDP-Lite). // Internet Engineering Task Force, July 2004. RFC 3828.
8. H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, RTP: A transport protocol for real-time applications. // Internet Engineering Task Force, July 2003. RFC 3550.
9. E. Kohler, M. Handley and S. Floyd. Datagram congestion control protocol (DCCP). // Internet Engineering Task Force, November 2004.
10. E. Kohler, M. Handley and S. Floyd, Designing DCCP: Congestion control without reliability. / International Computer Science Institute, Berkeley, CA, USA, May 2003.
11. S. Floyd, M. Handley, J. Padhey, J. Widmer. Equation-Based Congestion Control for Unicast Applications. SIGCOMM, 2000.
12. M. Papadopouli, H. Shen, E. Raftopoulos, M. Ploumidis, F. Hernandes-Campos. Short-term traffic forecasting in a campus-wide wireless network, SIGCOMM, 2004.
13. M. Papadopouli, H. Shen, E. Raftopoulos. Evaluation of short-term traffic forecasting in wireless network, SIGCOMM, 2005.

МЕТОДИЧЕСКИЕ ОСОБЕННОСТИ ИСПОЛЬЗОВАНИЯ ГЕОИНФОРМАЦИОННЫХ СИСТЕМ ПРИ АНАЛИЗЕ ЭКОЛОГИЧЕСКОЙ ОБСТАНОВКИ В НАСЕЛЕННОМ ПУНКТЕ

Н.Г. Тихонова
Научный руководитель – д.т.н., профессор В.П. Вейко

В статье приведен разработанный автором пошаговый алгоритм работы пользователя с ГИС-системой, реализованный применительно к исследованию экологической обстановки в городе Йошкар-Ола.

Введение

В настоящий момент в практике проведения экологических изысканий и исследований широко применяются интегрированные информационные системы, в частности, геоинформационные системы (ГИС) и технологии [1]. Согласно наиболее простому определению, приведенному, например, в [2], «Геоинформационная система – это автоматизированная информационная система, предназначенная для обработки пространственно-временных данных, основой интеграции которых служит географическая информация».

Можно предположить, что работа с любым программным продуктом, в том числе и с ГИС, должна быть организована рационально. Если же это не сделано, то пользователь может потратить слишком много времени на то, чтобы получить нужный ему результат, и тогда его действия могут оцениваться как недостаточно эффективные. Поэтому задача, решение которой представлено ниже и суть которой сводится к организации эффективного взаимодействия пользователя с ГИС, может трактоваться как актуальная и отличающаяся новизной. Она была решена автором в рамках дипломного проекта, тематика которого определялась как проблема использования ГИС-технологий, в частности, программы ArcView 3.3? при оценке экологической обстановки в пределах населенного пункта. В дипломной работе автора в качестве такого населенного пункта рассматривался город Йошкар-Ола.

Алгоритм работы пользователя с ГИС-системой

Рис. 1. Алгоритм работы пользователя с ГИС-системой

Алгоритм работы пользователя (рис. 1) начинается со сбора исходных данных.

В качестве исходных данных могут выступать карты местности, представленные как на цифровых, так и на бумажных носителях, а также информация о месторасположении городских жилых и промышленных объектов, которые вносят существенный вклад в ухудшение экологической обстановки в городе. Кроме того, в качестве исходных данных используется информация от местных органов власти и центра государственного санитарно-эпидемиологического надзора.

На следующем этапе предусмотрено знакомство исполнителя с действующей на данный момент нормативной базой в области охраны окружающей природной среды. Под нормативной базой подразумеваются такие технические нормативные документы, как СНиПы, СанПиНы, технические условия (ТУ), а также юридические нормативные документы: законы РФ, постановления Правительства РФ, а также и распоряжения и указания местной администрации. На этом этапе могут быть использованы современные базы нормативной документации ГАРАНТ+ или КОДЕКС.

В ходе третьего этапа реализуется построение карт с нанесенной на них экологической информацией применительно к существующей на данный момент экологической ситуации на данной территории. Для того чтобы грамотно провести экологический анализ исследуемой территории, необходимо построить план современного использования территории для наглядного представления сложившейся ситуации, а также для создания основы проектируемой карты планировочных ограничений. Такая карта существующего использования территории называется опорным планом территории. Его основными компонентами являются жилые, садовые, общественные, административные, торговые, рекреационные зоны, зоны инженерной и транспортной инфраструктур, зоны специального назначения, промышленные и прочие городские территории.

На следующем этапе выполняется изучение и анализ источников негативного воздействия на окружающую природную среду анализируемого участка территории. Для анализа экологической обстановки применительно к территории города необходимо выделить источники негативного воздействия на окружающую природную среду. В городе такими источниками выступают: промышленные предприятия, автотранспорт, ливневая и сточная канализации, объекты инженерной инфраструктуры, очистные сооружения, свалки и полигоны промышленных и бытовых отходов, радиопередающие установки и пр.

В проведенном автором экологическом анализе были выявлены источники негативного воздействия в пределах города Йошкар-Ола, исходя из их воздействия на следующие компоненты:

- состояние воздушного бассейна;
- санитарное состояние водных ресурсов;
- санитарное состояние почв, очистка территории;
- акустическая обстановка;
- электромагнитная обстановка;
- радиационная обстановка.

На последующих этапах проводилась разработка мероприятий, проведение которых необходимо для улучшения экологического состояния территории. В качестве таких мероприятий могут выступать, например, озеленение территорий с целью улучшения качества воздуха; снижения шумового воздействия от промышленных предприятий, транспорта, строительства и организация санитарно-защитных зон; а также проведение мероприятий на конкретных промышленных установках, сбрасывающих или выбрасывающих загрязнения в окружающую среду для очистки выбрасываемых загрязнений.

На следующем этапе осуществляется перенесение информации, полученной в ходе анализа экологической ситуации, на карту планировочных ограничений. Суть этого

мероприятия состоит в следующем. По каждым из перечисленных выше основных экологических направлений были выявлены участки территорий, на которых необходимо организация и содержание их соответствующим образом в зависимости от зоны, к которой относится территория – например, статус особо охраняемой территории, санитарно-защитные зоны, зоны санитарной охраны I-III поясов источников водоснабжения и пр. Далее эти ограничения наносятся на единую карту планировочных ограничений. На карте представлены основные источники загрязнений, особо охраняемые природные территории, охранные коридоры ЛЭП, автомобильных и железных дорог, зоны санитарной охраны водозаборов I и II поясов, границы водоемов с указанием постов наблюдений с превышениями исследуемых элементов по отношению к 1 ПДК, санитарно-технические объекты, объекты инженерной инфраструктуры с охранными коридорами для каждых.

Завершает работу составление резюме в форме пояснительной записки, которая является дополнением к карте планировочных ограничений для более детального описания нанесенных на нее ограничений и мероприятий.

Заключение

В результате оценки экологической обстановки в пределах населенного пункта г. Йошкар-Ола был разработан алгоритм работы пользователя, работающего с ГИС-системой, который нацелен на минимизацию времени пользователя программного продукта при условии выполнения им всех необходимых оценочных действий и структурирования полученной информации в соответствии с требованиями, предъявляемыми заказчиком данной работы.

Анализ результатов выполненной работы показал, что изучение ГИС-технологиям студентами лучше всего организовать в ходе их работы над конкретным проектом, в рамках которого четко определены его цели и задачи. Причем действия студентов должны выполняться в строгой последовательности согласно алгоритму и методике, разработанной на его основе.

Литература

1. Трифонова Т.А. и др. Геоинформационные системы и дистанционное зондирование в экологических исследованиях. М.: Академический Проект, 2005.
2. Цветков В.Я. Геоинформационные системы и технологии. М.: Финансы и статистика, 1998.

НАШИ АВТОРЫ

Андреева Наталья Викторовна – аспирант кафедры безопасных информационных технологий
Антонов Семен Евгеньевич – студент кафедры физики
Базаревский Валентин Эдуардович – студент кафедры программного обеспечения информационных технологий Белорусского государственного университета информатики и радиоэлектроники
Баранова Дарья Павловна – аспирант кафедры лазерных технологий и экологического приборостроения
Бедняков Иван Владиславович – студент кафедры вычислительной техники
Безгодов Алексей Алексеевич – студент кафедры вычислительной техники
Береговой Михаил Викторович – аспирант кафедры безопасных информационных технологий
Бобцов Алексей Алексеевич – доктор технических наук, профессор кафедры систем управления и информатики
Бреслав Андрей Андреевич – студент кафедры компьютерных технологий
Будько Михаил Юрьевич – аспирант кафедры мониторинга и прогнозирования чрезвычайных ситуаций
Будько Марина Борисовна – аспирант кафедры мониторинга и прогнозирования чрезвычайных ситуаций
Гирик Алексей Валерьевич – аспирант кафедры мониторинга и прогнозирования чрезвычайных ситуаций
Гланштейн Марьяна Анатольевна – студент кафедры вычислительной техники
Голицына Татьяна Дмитриевна – аспирант кафедры информатики и прикладной математики
Головков Иван Викторович – аспирант кафедры безопасных информационных технологий
Гусарова Дарья Алексеевна – аспирант кафедры безопасных информационных технологий
Евтифеева Ольга Алексеевна – студент Санкт-Петербургского государственного университета
Зайцев Олег Евгеньевич – аспирант кафедры безопасных информационных технологий
Захаров Александр Владиславович – аспирант кафедры безопасных информационных технологий
Иванов Константин Васильевич – выпускник Казанского государственного технического университета им. А.Н. Туполева
Калашник Евгений Олегович – аспирант кафедры безопасных информационных технологий
Камнев Дмитрий Анатольевич – аспирант кафедры систем управления и информатики
Карлыханов Алексей Николаевич – аспирант кафедры безопасных информационных технологий
Князев Евгений Геннадьевич – аспирант кафедры компьютерных технологий
Коротков Максим Алексеевич – студент кафедры компьютерных технологий
Красс Александр Леонидович – студент кафедры компьютерных технологий
Кремлёв Артем Сергеевич – кандидат технических наук, доцент кафедры систем управления и информатики

Лакунин Максим Александрович – студент Санкт-Петербургского государственного университета
Лобанов Павел Геннадьевич – аспирант кафедры компьютерных технологий
Лужков Юрий Валерьевич – аспирант кафедры вычислительной техники
Лукичев Александр Николаевич – аспирант кафедры вычислительной техники
Лукьянова Александра Павловна – студент кафедры компьютерных технологий
Лысенко Екатерина Алексеевна – студент Санкт-Петербургского государственного университета
Марченко Антон Анатольевич – аспирант кафедры безопасных информационных технологий
Матвиенко Сергей Владимирович – кандидат технических наук, докторант кафедры безопасных информационных технологий
Мейко Анастасия Владимировна – аспирант кафедры прикладной математики и информатики Казанского государственного технического университета им. А.Н. Туполева
Минакова Наталья Александровна – аспирант кафедры безопасных информационных технологий
Мироненко Евгений Петрович – аспирант Санкт-Петербургского государственного университета аэрокосмического приборостроения
Мицкевич Татьяна Владимировна – аспирант кафедры редакционно-издательских технологий Белорусского государственного технологического университета
Нгуен Дык Тай – аспирант кафедры вычислительной техники
Нестерук Филипп Геннадьевич – кандидат технических наук, доцент кафедры безопасных информационных технологий
Новосельский Вениамин Борисович – аспирант кафедры информатики и прикладной математики
Пастухов Андрей Сергеевич – аспирант кафедры безопасных информационных технологий
Поликарпова Надежда Игоревна – студент кафедры компьютерных технологий
Раер Михаил Геннадьевич – аспирант кафедры компьютерных технологий
Ржеутская Анна Андреевна – аспирант кафедры информационных систем и технологий Белорусского государственного технологического университета
Рубинов Константин Викторович – аспирант кафедры компьютерных технологий
Серебрякова Владлена Сергеевна – аспирант кафедры физики и техники оптической связи
Симоненко Екатерина Юлиевна – студент кафедры информатики университета AIX-MARSEILLE II (Франция)
Спивак Антон Игоревич – аспирант кафедры безопасных информационных технологий
Степанчук Надежда Владимировна – студент кафедры программного обеспечения информационных технологий Белорусского государственного университета информатики и радиоэлектроники
Счастливцев Роман Романович – студент Санкт-Петербургского государственного университета
Сытник Сергей Анатольевич – аспирант кафедры компьютерных технологий
Темнов Олег Дмитриевич – аспирант кафедры безопасных информационных технологий
Тихонова Наталья Геннадьевна – аспирант кафедры лазерных технологий и экологического приборостроения
Токалов Никита Сергеевич – студент кафедры информационных систем

Топилин Сергей Александрович – аспирант кафедры систем управления и информатики
Торшенко Юлия Александровна – аспирант кафедры безопасных информационных технологий
Точилин Владимир Николаевич – студент кафедры компьютерных технологий
Тутубалин Павел Иннокентьевич – аспирант кафедры прикладной математики и информатики
Шопырин Данил Геннадьевич – кандидат технических наук, доцент кафедры компьютерных технологий

СОДЕРЖАНИЕ

Научно-технический вестник СПбГУ ИТМО. Выпуск 39. **ИССЛЕДОВАНИЯ В ОБЛАСТИ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ. Труды молодых ученых** / Главный редактор д.т.н., проф. В.Н. Васильев. – СПб: СПбГУ ИТМО, 2007. 332 с.

НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК СПбГУ ИТМО
Выпуск 39

ИССЛЕДОВАНИЯ В ОБЛАСТИ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ.
Труды молодых ученых

Главный редактор
доктор технических наук, профессор
В.Н. Васильев
Дизайн обложки В.А. Петров, А.А. Колокольников
Редакционно-издательский отдел СПбГУ ИТМО
Зав. РИО Н.Ф. Гусарова
Лицензия ИД № 00408 от 05.11.99.
Подписано в печать 10.10.07.
Заказ 1060. Тираж 100 экз.