МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК

Выпуск 25

ИССЛЕДОВАНИЯ В ОБЛАСТИ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

САНКТ-ПЕТЕРБУРГ
2006

Выпуск содержит материалы **III межвузовской конференции молодых ученых**.
Конференция была организована 10–13 апреля 2006 года Санкт-Петербургским государственным университетом информационных технологий, механики и оптики в сотрудничестве с

– Санкт-Петербургским государственным университетом
– Санкт-Петербургским государственным инженерно-экономическим университетом
– Казанским государственным университетом
– Санкт-Петербургским государственным технологическим институтом (техническим университетом)
– Санкт-Петербургским государственным политехническим университетом
– Санкт-Петербургским государственным университетом кино и телевидения
– Санкт-Петербургским государственным морским техническим университетом
– Самарским государственным архитектурно-строительным университетом
– Российским государственным гуманитарным университетом
– Мурманским государственным техническим университетом
– Ленинградским государственным университетом им. Пушкина
– ФГУП «НИТИОМ ВНЦ «ГОИ им. С.И. Вавилова»
– ОАО «Российским институтом радионавигации и времени»
– Российской академией художеств.

В выпуске представлены работы, поддержанные финансированием в рамках:

– Федеральной целевой научно-технической программы «Исследования и разработки по приоритетным направлениям развития науки и техники на 2002–2006 гг.» (Федеральное агентство по науке и инновациям);
– 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

ОРГАНИЗАЦИОННЫЙ КОМИТЕТ КОНФЕРЕНЦИИ

ISSN 1819-222X

1 ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ

ИНСТРУМЕНТАЛЬНОЕ СРЕДСТВО ДЛЯ ТРАНСЛЯЦИИ ИЕРАРХИЧЕСКИХ ДОКУМЕНТОВ В РЕЛЯЦИОННЫЕ БАЗЫ ДАННЫХ

О.Е. Вашенков

Научный руководитель – кандидат технических наук, доцент А.В. Лямин

В работе рассматривается метод трансляции иерархических документов XML в реляционные базы данных. Суть метода – в разработке специальных правил транслирования для установки соответствия между иерархическим документом и реляционной структурой.

Введение

Данные, представленные в иерархическом формате с XML-разметкой (eXtensible Markup Language – расширяемый язык разметки), удобны для прочтения человеком и для машинной обработки. Существует множество средств для обработки и преобразования таких данных, например, язык преобразований XSL (eXtensible Stylesheet Language – расширяемый язык таблиц стилей) [1, 2]. Однако в основном такие средства позволяют преобразовать документы в удобный для прочтения человеком вид.

Хранение данных в базе данных (БД) предполагает иную структуру – реляционную. Стандартные средства для обработки XML-документов не способны самостоятельно транслировать произвольные документы в реляционные структуры. В данной работе рассматривается вариант, когда данные в XML-документе распределяются по таблицам базы данных, а не хранятся в виде документов XML в отдельных ячейках таблиц. Оба подхода описаны в главе 27 книги [3]. Развитие языка SQL (Structured Query Language – структурированный язык запросов) для поддержки XML рассмотрено в работе [4]. Языки запросов, основанные на XML, представлены в работе [5]. Недостатки совместного использования реляционных и иерархических структур описаны в главе 25 книги [3] и в работе [6]. Третий подход с использованием XML-ориентированных баз данных в данной статье не рассматривается. Решение, когда документ хранится как отдельный атрибут в кортеже, предложено компанией Oracle в системе управления базами данных (СУБД) Oracle 9. Соответствующий атрибут имеет тип данных XMLTYPE. В языке SQL этой СУБД представлены средства для выборки данных из XML-документов по выражениям XPath (XML Path Language – язык для поиска фрагментов в XML-документах) [7, 8]. В книге [9] приводится описание работы с полями типа XMLTYPE в СУБД Oracle 9. Детальное описание подхода компании IBM к решению данной проблемы при работе с DB2 описано в [10]. Наш подход не требует усложнения языка запросов к базе данных, накладываются только некоторые ограничения на XML-документы и структуру БД для корректного функционирования правил транслирования. На уровне базы данных можно работать со стандартным языком запросов SQL.

Для реализации такого подхода были поставлены следующие задачи:

1. разработка набора правил для отображения иерархических XML-документов в реляционную СУБД;
2. разработка требований для XML-документов;
3. разработка требований к структуре базы данных;

4. разработка программного обеспечения для прямой и обратной трансляции иерархических документов на основе разработанных DTD-определений (Document Type Definition – определение типа документа) и структуры БД, отвечающей разработанным правилам.

Основное требование к программному обеспечению (ПО) – документная независимость и независимость от структуры базы данных, т.е. логика работы программы основана на небольшом наборе универсальных правил. Для доступа к XML-документам используются стандартные средства обработки XML, а для доступа к базе данных – стандартный драйвер баз данных с единым интерфейсом, независимо от производителя СУБД, способный выполнять SQL-запросы и предоставлять метаинформацию о структуре БД.

Трансляция данных

Рассмотрим основу преобразования – правила трансляции. Метод, основанный на таких правилах, позволяет транслировать произвольные документы XML в базу данных, а также выполнять обратное преобразование. Апробация правил трансляции происходила с использованием документов описания электронных методических комплексов системы дистанционного обучения, которые необходимо было транслировать и хранить в СУБД системы.

Элементы электронных учебно-методических комплексов (УМК) системы дистанционного обучения (ДО) СПбГУ ИТМО, представленные XML-документами, описывают множество различных учебных документов: учебные планы, дисциплины, семестровые курсы и сами учебные элементы. К последним относятся электронные конспекты, тесты, виртуальные лабораторные работы, практикумы и информационные ресурсы. Документы, используемые для описания этих элементов, имеют сложную иерархическую структуру, допускающую изменения в ходе всего цикла разработки системы ДО. Поэтому от системы трансляции требовалась гибкость и универсальность.

Для отображения иерархических документов в реляционную структуру был разработан набор правил. DTD-определения элементов УМК и структура БД удовлетворяют этим правилам. Основным преимуществом такого подхода является уменьшение связанности ПО системы, документов и структуры БД. При изменениях в структуре базы данных или структуре XML-документа не требуется обновления ПО трансляции.

Импорт данных

В процессе импорта требуется определить таблицы текущей схемы базы данных и их поля. Этой задачей занимается модуль интроспекции базы данных. Определение таблиц для внесения данных определяется по названиям элементов XML-документа. Элемент, которому сопоставлена таблица в базе данных, подлежит внесению. Атрибуты элементов представляют поля таблиц, названия которых совпадают с текущим или верхним по иерархии элементом. Таким образом, реализовано правило «всплытия» атрибутов, когда атрибуты, не сопоставленные таблице текущего элемента, вносятся в таблицу верхнего по иерархии элемента. Существует отдельное правило внесения элемента Data. Этот элемент представляет текстовое поле размерностью до 4000 символов и интерпретируется как поле таблицы текущего или верхнего по иерархии элемента.

В качестве примера рассмотрим операцию сопоставления документа XML, представленного ниже, и структуры базы данных.

```
<Program ID=”1”>
 <Head>
 <Comment ProgramName=”Название программы”>
 <Data><!----></Data>
 </Comment>
 </Head>
</Program>
```

Реляционная структура представлена на рис. 1.

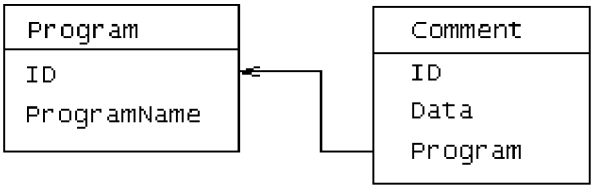


Рис. 1. Структура базы данных

В этом случае таблицам будут сопоставлены элементы Program и Comment. Специальный элемент Data представляет текстовые данные в таблице Comment. Атрибут ProgramName из элемента Comment «всплывает» и заносится в поле таблицы Program. Поле Program таблицы Comment является внешним ключом для связи с таблице Program. Для корректного заполнения этого поля в DTD-определении исходного документа следует использовать атрибут с именем Program и спецификатором #FIXED “ID”.

```
<!ELEMENT Program (Head)>
<!ATTLIST Program
 ID CDATA #REQUIRED
>
<!ELEMENT Head (Comment)>
<!ELEMENT Comment (Data?)>
<!ATTLIST Comment
 ProgramName CDATA #REQUIRED
 Program CDATA #FIXED “ID”
>
<!ELEMENT Data (#PCDATA)>
```

При внесении записей в таблицы-словари важно исключить их дублирование. Уникальность гарантируется добавлением атрибута isKey к элементу, имя которого сопоставлено таблице-словарю. Пример DTD-определения с требованием уникальности записи представлен ниже.

```
<!ELEMENT FrameIndex (…)>
<!ATTLIST FrameIndex
 FrameID CDATA #REQUIRED
 Weight CDATA #REQUIRED
 IsKey CDATA #FIXED “Yes”
>
```

Дополнительный атрибут IsKey не допускает загрузки одинаковых записей в таблицу FrameIndex. Проверка происходит на основе полей-атрибутов FrameID и Weight.

Еще одной особенностью является внесение внешних файлов, ссылки на которые указаны в XML-документах. Для организации ссылки в элементе определен атрибут Src, в базе данных ему сопоставляется текстовое поле Src для имени файла и поле типа BLOB или CLOB с названием Cnt для хранения содержания.

При внесении текстовых данных в поля Data или Cnt в них могут содержаться ссылки на внешние файлы, например в документах HTML (Hypertext Markup Language – язык разметки гипертекста). Для таких полей среда трансляции запоминает путь к данным (таблица и имя поля) для преобразования ссылок. Процесс преобразования ссылок запускается после внесения всех данных, а логика преобразования реализована на стороне базы данных в виде пакета хранимых процедур. Среда трансляции для каждой записи в подготовленном списке таблиц и полей вызывает хранимую процедуру для преобразования ссылок. Пути к реальным файлам преобразуются в запросы протокола HTTP (HyperText Transfer Protocol – протокол передачи гипертекста) на получение вложений. Процесс преобразования схематично представлен на рис. 2.

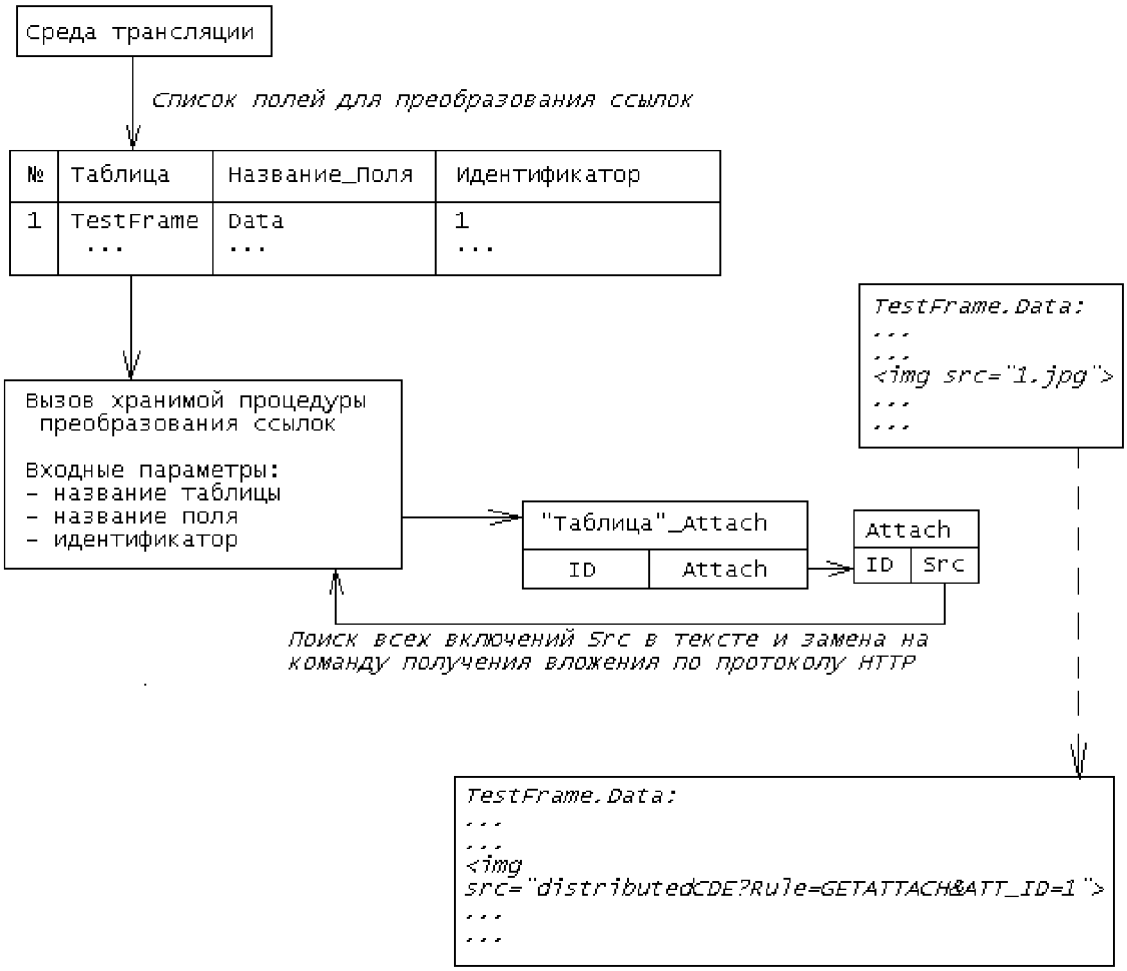


Рис. 2. Преобразование ссылок

Такой подход является специфичным для разработанного средства трансляции, однако подход, когда преобразованием данных занимается программная часть в БД, позволяет сосредоточить логику обработки данных в самой СУБД. Это упрощает реализацию модуля трансляции и модификацию логики работы системы, которая использует разработанный механизм.

Организация связей

Рассмотрим процесс преобразования иерархических данных к реляционным данным с соблюдением реляционных связей.

Существует два способа организации связи «многие к одному». Первый реализуется с помощью атрибутов-ссылок, т.е. атрибутов, названия которых совпадают с названиями таблиц. В основном такой подход используется для ссылки на таблицы-словари. Поиск в таблице-словаре происходит по значению атрибута, которое, после поиска, заменяется на идентификатор записи в таблице-словаре. Для каждой таблицы определены поля, по которым происходит поиск, и поля-идентификаторы. Эти определения хранятся в самой базе данных в отдельной таблице и дополняют метаданные, предоставляемые драйвером СУБД.

Рассмотрим пример организации связи «многие к одному» для таблиц-словарей. Предположим, элементу поставлена в соответствие таблица в базе данных.

<TestFrame Language=”English”>…</TestFrame>

Таблица TestFrame содержит поле Language, являющееся внешним ключом на таблицу Language. Таблица Language – словарь со списком доступных языков.

Среда трансляции проверит наличие записи в таблице Language по значению, представленному в атрибуте, и в поле TestFrame.Language будет занесен идентификатор записи. Процесс уточнения идентификатора записи представлен на рис. 3.

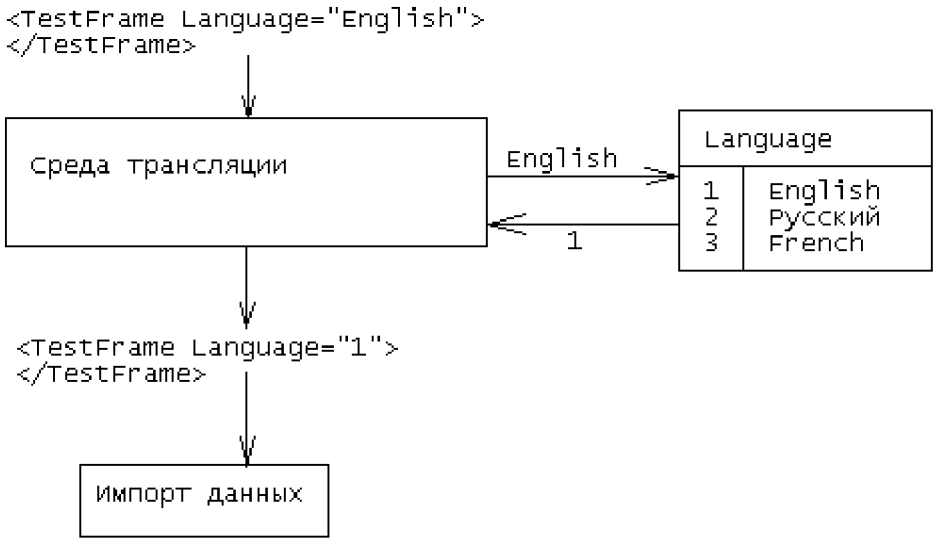


Рис. 3. Организация связи «многие к одному»

Второй способ используется при организации связи «многие к одному» по составному ключу. В этом случае используются вложенные элементы в иерархическом документе. Составным ключом служит набор атрибутов вложенного элемента, таблицей-словарем – соответствующая по названию таблица базы данных. Для предотвращения импорта данных в таблицы-словари используются служебные атрибуты isKey в DTD-определениях. Альтернатива этого способа для организации связи между элементами на различных уровнях иерархии описана ниже.

Связь «многие ко многим» организуется также за счет использования иерархии: если один элемент вложен в другой в XML-документе и существует таблица связи с

названием «Элемент1_Элемент2», то после внесения записей в две таблицы идентификаторы заносятся в таблицу связи.

Пример реализации связи «многие ко многим» показан ниже.

```
<!ELEMENT TestFrame(…, Data, Attach*,…)>
<!ATTLIST TestFrame
…
>
<!ELEMENT Attach>
<!ATTLIST Attach
 Src CDATA #REQUIRED
>
<TestFrame>
 <Data><!-- Текст вопроса --></Data>
 <Attach Src=”1.jpg”/>
 <Attach Src=”2.jpg”/>
</TestFrame>
```

Элемент TestFrame является контейнером для вопроса, вариантов ответа и вложений в системе тестирования знаний. Вложения представлены элементами Attach.

В терминах реляционной СУБД кратность отношений TestFrame и Attach – «многие ко многим». Данные документа соответствуют реляционной структуре, представленной на рис. 4.

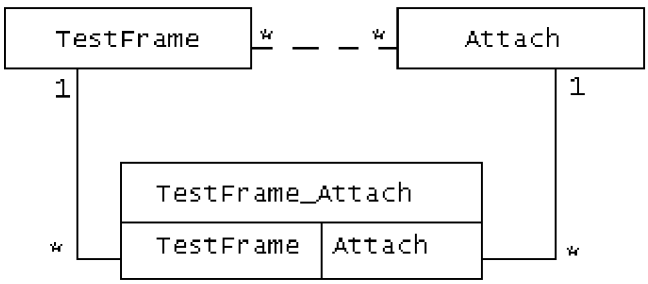


Рис. 4. Организация связи «многие ко многим»

Для организации связи между двумя элементами, каждый из которых представляет запись в таблице, т.е. не является ссылкой на данные в таблице-словаре, используются служебные атрибуты. Когда один элемент вложен в другой, в DTD-определении элемента указывается служебный атрибут, название которого совпадает с названием связанного элемента. Далее рассмотрим пример реализации связи «многие к одному». Исходному XML-документу соответствует DTD-определение:

```
<!ELEMENT TextBookUnit(PageIndex*)>
<!ELEMENT PageIndex(TextBookPage)>
<!ATTLIST PageIndex ... >
<!ELEMENT TextBookPage>
<!ATTLIST TextBookPage
 …
 TextBookUnit CDATA #FIXED "ID"
 PageIndex CDATA #FIXED "ID"
>
```

Исходный документ:

```
<TextBookUnit>
 <PageIndex ...>
 <TextBookPage .../>
 </PageIndex>
</TextBookUnit>
```

Элементы TextBookUnit соответствуют пунктам оглавления электронных учебников. PageIndex – элемент, которому соответствует таблица индексирования страниц электронных учебников. TextBookPage – контейнер для данных конкретной страницы электронного учебника. Таким образом организована связь «многие к одному» для элементов, расположенных на разных уровнях иерархии.

В структуре реляционной СУБД отношение TextBookPage связано с отношениями PageIndex и TextBookUnit связью «многие к одному». Данные документа будут преобразованы в данные табличного пространства, представленного на рис. 5.

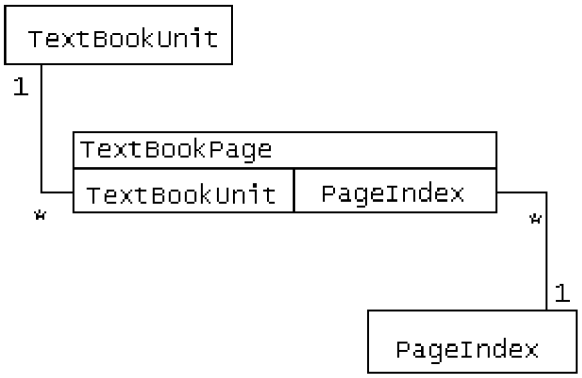


Рис. 5. Связь элементов на разных уровнях иерархии

Экспорт данных

Экспорт происходит на основе данных о структуре базы и требует указания начальной таблицы, записи из которой будут представлены XML-элементами, подчиненными корневому элементу. Так как связи между данными могут быть достаточно сложными, дополнительно требуется указать таблицы, которые участвуют в экспорте. Экспортируемые таблицы представляют собой элементы XML-документа. Их поля – атрибуты. Если поле – ссылка на таблицу-словарь, значение меняется на значение из таблицы-словаря. Таблицы классифицируются как таблицы-словари по служебной таблице с информацией о полях-идентификаторах и полях-значениях. Поле может быть ссылкой на другую таблицу, в таком случае XML-элемент является верхним по иерархии относительно элемента, соответствующего текущей таблице. Таким образом обеспечивается корректность связей «один ко многим» по составному ключу. Текстовые данные, представленные в полях Data, записываются в виде текстовых данных XML-элемента.

Рассмотрим табличное пространство, представленное на рис. 6.

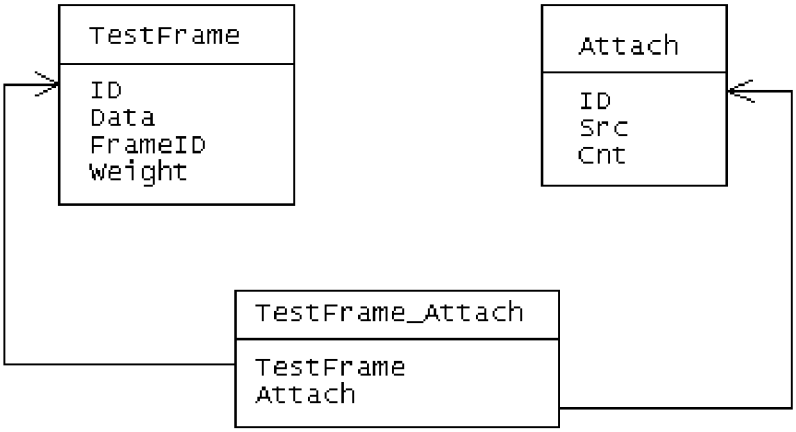


Рис. 6. Табличное пространство со связью типа «многие ко многим»

Требуется выполнить экспорт из таблиц TestFrame и Attach, итоговый документ XML может выглядеть следующим образом:

```
<?xml version=”1.0” encoding=”Windows-1251”?>
<TestFrames>
<TestFrame FrameID=”1” Weight=”1”>
 <Data><!-- вопрос тестового задания --></Data>
 <Attach Src=”1.jpg”/>
 <Attach Src=”2.jpg”/>
</TestFrame>
<TestFrame FrameID=”2” Weight=”1”>
 <Data><!-- вопрос тестового задания № 2--></Data>
 <Attach Src=”3.jpg”/>
</TestFrame>
</TestFrames>
```

При этом название корневого элемента TestFrames и список таблиц для экспорта указывает пользователь.

Практическая реализация

Новые технологии в области разработки программного обеспечения, такие как XML и Java, позволили разработать и реализовать механизм трансляции. Язык Java позволил реализовать кросс-платформенную оболочку для разбора и проверки XML-документов, сбора метаданных БД, преобразования документов в структуру БД и обратно. Уменьшение связанности со структурой документов обеспечивается при использовании XML-формата. При работе с СУБД оболочка динамически собирает метаинформацию о структуре. Статически описаны только правила трансляции иерархических документов XML и операторы языка SQL, информация о таблицах и полях собирается на основе названий элементов XML-документа, служебных таблиц-словарей и метаданных, предоставленных JDBC-драйвером базы данных (JDBC, Java Data Base Connectivity – общий модуль соединения с базами данных для Java). Корректность составления XML-документов гарантируется DTD-определениями.

Слабая связь с конкретной базой данных обеспечивается за счет интерфейса JDBC и разработанного драйвера базы данных, задача которого – выполнение базовых операций языка DML (Data Manipulation Language – язык манипулирования данными).

От JDBC-драйвера конкретной СУБД требуется обеспечение доступа к метаинформации, выполнение SQL-инструкций и обеспечение работы с LOB-объектами (Large Object – поле в базе данных без явного ограничения размера для размещения данных большого объема).

Динамическое решение является альтернативой системам со статическим форматом хранения ресурсов, структурой БД и кодом импорта/экспорта. Динамический метод позволяет избежать следующих проблем:

1. изменение структуры БД или документов приводит к необходимости изменения кода транслятора;
2. написание модуля обратной трансляции – это отдельная задача, требующая знания формата документов и структуры БД;
3. описание связей между данными при статической организации осложняется ограничениями в реализации ПО.

При использовании динамического метода наиболее сложным моментом реализации становится модуль обработки исключений, возникающих во время операций трансляции. На этапе проверки XML-документа подробные сообщения об ошибках доступны через средства Java для работы с XML. При загрузке же для вывода доступны только системные сообщения СУБД. «Незнание» системы трансляции структуры документа осложняет процесс. Однако разработанный модуль обработки ошибок позволяет пользователю определить путь к элементу, при трансляции которого произошло исключение, и системное сообщение, что упрощает поиск ошибки. А СУБД, например Oracle, позволяет установить системные сообщения для конкретных ошибок, связанных с нарушением целостности данных.

В настоящий момент программа, реализующая алгоритмы трансляции, используется в системе дистанционного обучения СПбГУ ИТМО. Как показывает практика, разработанные методы и правила позволяют менять структуру документов и базы данных без изменения кода программы, а программное средство позволяет полностью автоматизировать заполнение базы данных.

При использовании программы создатель УМК может сосредоточиться на содержательной части документа. За корректностью вводимой информации следит система проверки XML-документов на основе DTD-определений и СУБД на основе средств проверки целостности данных. Операция экспорта представляет ценность при необходимости проведения крупномасштабных изменений в базе данных, например, при изменениях в структуре семестрового курса или в банке тестовых заданий.

Использование разработанного средства позволяет минимизировать усилия по импорту и экспорту больших объемов данных между иерархическими документами и реляционными СУБД. Упрощается внедрение новых возможностей, уменьшаются затраты на поддержку программного обеспечения.

Основная область применения разработанной системы – трансляция данных между представлением в формате иерархического документа и в структуре реляционной СУБД. Возможной областью применения может служить обеспечение трансляции при построении распределенных систем, обмен данными в которых происходит в формате XML (системы на основе Web-сервисов), а хранилищами служат реляционные базы данных. Разработанная система позволит унифицировать процесс подготовки и обработки данных для обмена.

С клиентской стороны представление документов в формате XML дает гибкость при создании пользовательского интерфейса. Так как изначально документы XML пригодны для чтения человеком и компьютером, то визуальная среда для отображения данных может просто демонстрировать XML-документ или реализовывать сложный графический интерфейс. С серверной стороны используется стандартная реляционная база данных и язык SQL, без необходимости использования средств работы с XML.

Заключение

В результате проделанной работы были разработаны правила трансляции иерархических документов в реляционные СУБД, требования к структуре документов XML и структуре базы данных, а также программное средство, реализующее эти правила. Разработанное программное обеспечение, обладая слабой связанностью с документами и структурой БД, практически не нуждалось в модификации и сопровождении при разработке системы дистанционного обучения. В настоящий момент средство активно используется в процессе импорта электронных учебных материалов в формате XML в систему ДО и при модификации загруженных ранее элементов.

Литература

1. Валиков А. Технология XSLT. СПб: БХВ-Петербург, 2002. 544 с.
2. http://www.w3.org/TR/1999/REC-xslt-19991116 – нормативный документ спецификации XSLT.
3. Дейт К.Дж. Введение в системы баз данных. / 8-е издание.: пер. с англ. М.: Издательский дом «Вильямс», 2005. 1328 с.
4. International Organization for Standardization (ISO). «XML-Related Specifications (SQL/XML) Working Draft», Document ISO/IEC JTC1/SC32/WG3:DRS-020. – August 2002.
5. Bonifati A., Ceri. Corporative Analysis of Five XML Query Languages // ACM SIGMOD Record. March 2000. 29, №1.
6. Bosak J., Bray T. XML and the Second-Generation Web. // http://www.sciam.com May 1999.
7. Lehmann M. From XML to Storage and Back. // Oracle Magazine March 2003. http://www.oracle.com/technology/oramag/oracle/03-mar/index.html
8. Dillon S. XML to Relational: Bridging the Gap. // Oracle Magazine September 2005. http://www.oracle.com/technology/oramag/oracle/05-sep/index.html
9. Scardina M., Chang B., Wang J. Oracle 10g XML & SQL: Design, Build & Manage XML Applications in Java, C, C++ & PL/SQL. // Osborne, 2004. 600 p.
10. Steegmans B., Bourret R., Guyennet O., Kulkarni S., Priestly S., Cline O., Sylenko V., Wahli U. XML for DB2 Information Integration. 2004. http://www.redbooks.ibm.com/redbooks.nsf/redbooks

ИССЛЕДОВАНИЕ ТЕХНОЛОГИИ СОЗДАНИЯ МОДУЛЬНО-НАРАЩИВАЕМЫХ МНОГОПРОЦЕССОРНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ С ПРОГРАММИРУЕМОЙ АРХИТЕКТУРОЙ НА ОСНОВЕ РЕКОНФИГУРИРУЕМОЙ ЭЛЕМЕНТНОЙ БАЗЫ

П.В. Кужаков

Научный руководитель – кандидат технических наук, доцент А.Л. Андреев

В статье исследуются технологии создания высокопроизводительных модульно-наращиваемых многопроцессорных вычислительных систем (ВММВС) с программируемой архитектурой и структурно-процедурной организацией параллельных вычислений для эффективного решения широкого класса вычислительно-емких задач.

Введение

Целью работы является исследование и разработка технологии создания высокопроизводительных модульно-наращиваемых многопроцессорных вычислительных систем (ВММВС) с программируемой архитектурой и структурно-процедурной организацией параллельных вычислений для эффективного решения широкого класса вычислительно-емких задач.

Предлагаемая технология разрабатывалась с учетом следующих требований:

- обеспечение реальной производительности системы не ниже 50% от пиковой для вычислительно-трудоемких задач различных классов;
- обеспечение динамической перестройки архитектуры системы в процессе решения задач;
- использование современных ПЛИС, производительность базового модуля – не ниже 50Гфлопс;
- эффективность в решении задач цифровой обработки сигналов, математической физики, символьной обработки, управления объектами энергетики, моделирования сложных процессов, потоковой обработки информации в реальном времени, вычислительно-емкой обработки изображений, криптографии и криптоанализа, стеганоанализа, а также многих других.

1. Разработка общей структуры ВММВС

Система интегрируется из унифицированных базовых модулей, разработанных в рамках наиболее эффективного на сегодняшний день стандарта Advanced TCA для решений верхнего уровня. Концепция Advanced TCA позволяет снизить затраты на разработку и ускорить создание готовой продукции, а также расширяет возможности проектирования основанных на стандартах высокоинтегрированных вычислительных систем высокой степени готовности.

Новый класс оборудования будет основываться на стандартах аппаратных средствах, операционных системах реального времени и интерфейсах программирования (API) для приложений высокой готовности.

1.1. Технология Advanced TCA

Архитектура Advanced TCA (Advanced Telecom Computing Architecture) – это первый в индустрии открытый стандарт для создания оборудования на базе современных технологий высокоскоростной коммутации [1, 2]. Помимо основных положений, в разделах спецификации PICMG 3.0 содержится подробная информация о питании и охлаждении, комплексном управлении системой и обеспечении электронной идентифика-

ции и совместимости установленных на одной «полке» функциональных модулей. Семейство спецификаций Advanced TCA – это основа для целого ряда долговременных платформ, предусматривающих создание огромного выбора перспективных взаимозаменяемых модулей различного назначения. Кроме того, Advanced TCA – это стандарт форм-факторов плат и корпусов, оптимизированный для вычислительно-емких приложений и призванный удовлетворить требования к масштабированию систем. В частности, спецификация предусматривает пропускную способность внутренней соединительной шины корпуса до 4,5 ТБ/с, поддержку оперативной памяти свыше 8 ГБ на плату, повышенную готовность системы и запас ресурсов для дальнейшего наращивания производительности.

Спецификация Advanced TCA формируется с учетом растущих требований к вычислительным системам, определяет новую архитектуру их модульного построения и использует пакетный принцип коммутации, обеспечивая высокую степень масштабируемости коммутирующих структур.

Подсистемы для резервирования улучшают надежность и пригодность всей системы.

Не менее важным аспектом спецификаций Advanced TCA являются централизованные средства системного управления. В частности, определяются резервированные механизмы централизованного управления платформой, которые реализуются на базе интерфейса Intelligent Platform Management Interface (IPMI) 1.5 и дублированных соединений IPMI/IPMB для каждого узла. Архитектура Advanced TCA исключает так называемые «единственные точки отказа» за счет использования резервированных контроллеров управления корпусом, резервированных шин электропитания для каждой платы и избыточности, обусловленной самой топологией коммутационных матриц.

Соответствующие стандартам Advanced TCA модульные конструктивные компоненты, включая монтажные стойки, платы, модули управления и блоки для замены в условиях эксплуатации (FRU), призваны обеспечивать на протяжении всего жизненного цикла оборудования такие важные преимущества, как упрощенное техническое обслуживание, облегченная модернизация, поддержка нескольких протоколов, расширенные возможности взаимодействия, а также возможность резервирования подсистем для повышения надежности и готовности систем.

Спецификация Advanced TCA в самой своей основе разработана как недорогая альтернатива современным закрытым архитектурам высокопроизводительного оборудования. Использование модульных конструктивных компонентов, совместимых со спецификациями Advanced TCA, уже сегодня позволяет производителям вычислительных систем создавать высокопроизводительные решения с высокой пропускной способностью, обладающие запасом ресурсов для гибкого решения широкого круга задач. Дополнительное преимущество модульных конструктивных компонентов стандарта Advanced TCA состоит в снижении стоимости разработки и сокращении времени выхода продукции на рынок.

1.2. Архитектура оборудования

В спецификацию Advanced TCA входят несколько типов модулей. Базовые (несущие) платы предназначены для обеспечения межблочного обмена, подключения функциональных (AMC) модулей, поддержания общей работоспособности системы, ее модульной архитектуры. Также несущие платы могут обеспечивать специальные функции, например, взаимодействие с внешними системами. Таким образом, несущие платы в общем случае выполняют базовые функции обеспечения взаимодействия модулей. AMC (Advanced

Mezzanine) является дополнительной спецификацией, полностью согласованной с семейством спецификаций PICMG 3.X.

К несущей плате подключается до восьми модулей расширения (AMC), выполняющих специализированные интерфейсные функции в зависимости от требований конкретной задачи. Модули расширения выполняются в форм-факторе 73*185 мм, 146*185 мм; с полной и половинной высотой установки компонентов.

Модули расширения устанавливаются в несущую плату по принципу мезонинов. Модули расширения также стандартизированы по конструктивным показателям и интерфейсам обмена с несущими платами, что снижает стоимость разработки и сокращает время выхода продукции на рынок.

Для организации обмена несущей платы с модулями расширения предусмотрена значительная степень свободы, что позволяет оптимизировать интерфейс взаимодействия в зависимости от конкретной задачи, но в то же время обеспечивает совместимость модулей расширения как минимум на аппаратном уровне.

В интерфейсе взаимодействия функциональных модулей предусмотрена значительная гибкость. Возможно применение широкого спектра модулей расширения.

1.3. Элементная база системы

С учетом требований задачи наиболее эффективной элементной базой для ее решения представляются ПЛИС последних поколений [3]. За счет большой емкости они позволяют реализовывать множество задач на логических элементах, обладают высокоскоростными последовательными интерфейсами, как правило, имеют встроенные процессорные ядра, оптимизированные для специфических задач. Внутренняя архитектура современной мощной ПЛИС, по существу, способна заменить сложную печатную плату с набором специализированных процессоров, микросхем памяти и интерфейсных контроллеров ввода/вывода.

Современную ПЛИС большой емкости можно рассматривать как универсальный вычислительный модуль с конфигурируемой архитектурой и развитыми интерфейсами обмена [4, 5]. Отладочные инструменты позволяют эффективно создавать широкий спектр приложений на ПЛИС, используя, по сути, одну элементную базу [6–8].

По ряду возможностей и характеристик для решения задачи проекта выбраны ПЛИС Xilinx семейства Virtex-4. Семейство Virtex-4 делится на три основные серии: LX, SX и FX. Все они содержат примерно по 200 тысяч логических элементов, работают на тактовой частоте 500 МГц (на этой же частоте работает интегрированный контроллер SmartRAM, реализующий принцип FIFO), содержат гигабитные порты ввода/вывода с синхронизацией разных каналов (ChipSync). В серию SX добавлено 512 Xtreme DSP суммарной производительностью 256 GigaMAC/c и потреблением 57 мкВт/МГц, а в серию PX встраивается до двух 32-разрядных RISC-процессоров PowerPC (заявленная производительность – 1300 Dhrystone MIPS), до четырех 10/100/1000 Ethernet MAC-контроллеров и до 24 последовательных портов RocketIO(пропускная способность – от 6,25 Мбит/с до 11,1 Мбит/с). Связь между FPGA-логикой и встроенными процессорными ядрами осуществляется по APU (Auxiliary Processor Unit).

Основные характеристики ПЛИС:

1) три платформы LX/SX/FX: a) Virtex-4 LX: решения для высокопроизводительной логики; b) Virtex-4 SX: высокопроизводительные решения для цифровой обработки сигналов; c) Virtex-4 FX: высокопроизводительные, полнофункциональные решения для встроенных платформ;

2) технология синхронизации Xesium: до 20 модулей цифрового управления синхронизацией (DCM): точная подстройка фазы тактовых сигналов; прецизионный сдвиг фазы тактовых сигналов; умножение и деление частоты; увеличена частота входной/выходной частоты; уменьшены выходные фазовые дрожания тактовых сигналов

(jitter); низкое энергопотребление; усовершенствованный фазовый детектор; широкий диапазон сдвига фаз; два режима работы; дополнительный фазированный делитель тактового сигнала (PMCD); дифференциальная структура синхронизации для минимизации фазовых дрожаний тактовых сигналов и обеспечение скважности 50%; 32 цепи глобальных тактовых сигналов; региональный ввод/вывод тактовых сигналов;

3) модуль XtremeDSP: умножитель 18×18, умножитель-накопитель, умножитель-сумматор; возможность организации конвейерной обработки для увеличения производительности; встроенный 48-битный аккумулятор для выполнения операции умножения с накоплением (MACC); интегрированный сумматор для выполнения операций умножения с суммированием; возможность каскадирования умножителей или MACC; увеличение скорости вычисления до 2 раз по сравнению с предыдущими семействами Virtex;

4) иерархия памяти: до 1 392 Кбит распределенного ОЗУ; до 10 Мбит интегрированной блочной ОЗУ; двухпортовая архитектура; возможность организации конвейерной обработки; возможность запрограммировать ПРО, поддерживающей флаги «пустой», «полный», программируемые флаги «почти полный» и «почти пустой»; программируемость работы в синхронном и асинхронном режиме; независимый выбор ширины порта чтения и записи (в конфигурации ОЗУ); блоки по 18 Кбит; конфигурация от 16К×1 до 512×36 (от 4К×4 до 512×36 в режиме FIFO); возможность каскадирования для формирования модулей памяти 32К×1 без использования трассировочных ресурсов кристалла; возможность побайтовой записи (например, при совместном использовании с процессором PowerPC 405);

5) технология SelectIO: до 960 пользовательских входов/выходов; широкий выбор сигнальных стандартов ввода-вывода от 1.5 до 3.3 В; высокая производительность: до 600 Мбит/с HSTL и SSTL (при передаче по одному выводу входа/выхода); до 1 Гбит/с LVDS (при передаче по дифференциальной паре ввода-вывода); схема цифрового управления согласованием: возможность подключения последовательных или параллельных согласующих резисторов; встроенная технология синхронизации источника сигнала – ChipSync™: способность побитного выравнивания встроена во все блоки ввода/вывода; отдельные входы/выходы для источников региональных тактовых сигналов; встроенная логика последовательно-параллельного и параллельно-последовательного преобразования данных и делитель частоты; поддержка сетевых и телекоммуникационных интерфейсов на скорости до 1 Гбит/с; поддержка интерфейсов к высокоскоростным модулям внешней памяти: DDR, DDR2,SDRAM, QDR-II, RLDRAM-II, FCRAM-II;

6) гибкие логические ресурсы: увеличена скорость работы конфигурационного логического блока до 40% по сравнению с предыдущими семействами Virtex; до 200 000 логических ячеек; до 178 176 регистров с входом разрешения тактового сигнала; до 178 176 таблиц преобразований; мультиплексоры, расширяющие возможности логики, и регистры в блоках ввода-вывода; каскадирование регистров сдвига или распределенной памяти;

7) конфигурация: криптографическая защита битового потока алгоритмом AE8; определение и исправление ошибок в битовом потоке; 4 режима загрузки; поддержка .JTAG интерфейса;

8) технология производства 90-нм КМОП процесс с металлизацией медью;

9) напряжение питания ядра 1.2 В;

10) корпус кристалла с шариковыми выводами, типа Flip-Chip;

11) аппаратный модуль высокоскоростного трансивера RocketIO MGT (только FX): скорость передачи от 622 Мбит/с до 11.1 Гбит/с; 8Ь/10Ь, 64Ь/66Ь кодер и декодер; поддержка объединения каналов; формирование и проверка CRC; программируемый передающий предварительный корректор; программируемый компенсатор на приеме; детектирование наличия/отсутствия сигнала на приеме; драйвер режима ожидания на передаче; динамическая реконфигурация модуля;

12) ядро RISC процессора фирмы IBM PowerPC 405 (только FX): работа на частоте до 450 МГц; пятиступенчатый конвейер данных; 16 КБ кэш инструкций; 16 КБ кэш данных; контроллер доступа к блочной памяти кристалла; возможность настройки отношения рабочих частот процессора и процессорной шины; дополнительный интерфейсный модуль (APU), обеспечивающий высокоскоростной обмен данными между процессором PowerPC 405 и сопроцессором, реализованным на логике: может работать на разных тактовых частотах; поддерживает автономные инструкции: нет остановов конвейера; 32-битные инструкции и 64-битные данные;

13) трехрежимный Ethernet MAC модуль (только FX): совместимость со стандартом 1EEE 802.3; 10, 100 и 1000 Мбит/с; поддержка автоопределения режима работы; фильтрация принимаемых адресов; единое решение для 1000 Base-X совместно с модулем RocketIO MGT; поддержка нескольких физических интерфейсов PHY через ресурсы ввода/вывода; отдельные интерфейсы на приеме и передаче для сбора статистической информации; поддержка пакета JUMBO; 3 интерфейса управления.

2. Разработка архитектуры системы

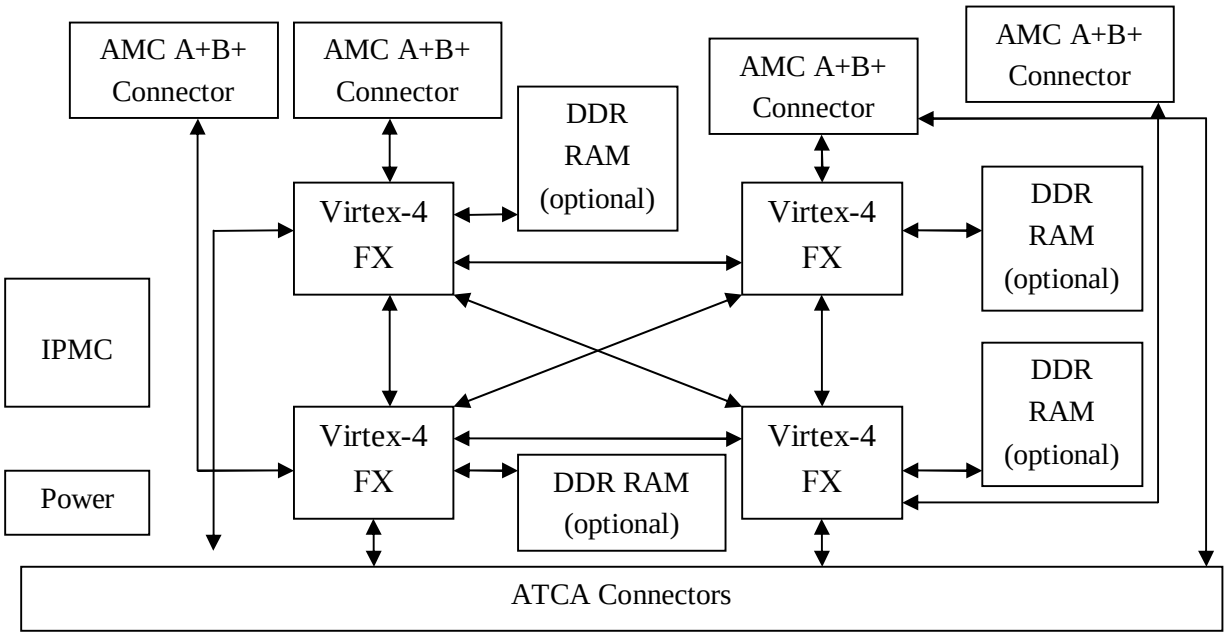


Рис. 1. Аппаратная структура несущей платы

Базовый модуль системы состоит из несущей платы, выполненной в соответствии со спецификацией AMC.0 Cutaway Carrier Boatd, и набора AMC модулей, оборудованных MMC контроллерами. Несущая плата объединяется с AMC модулями посредством LVDS-выводов. Несущая плата также оборудована микросхемами памяти DDR SDRAM для каждой ПЛИС и 1PMC контроллером, как показано на рис. 1. Несущая плата строится на базе ПЛИС группы FX.

Все AMC модули способны работать синхронно, что обеспечивает колоссальную производительность на математических задачах, например, при матричной обработке. Компоненты модуля объединяются по схеме полной сети (full mesh).

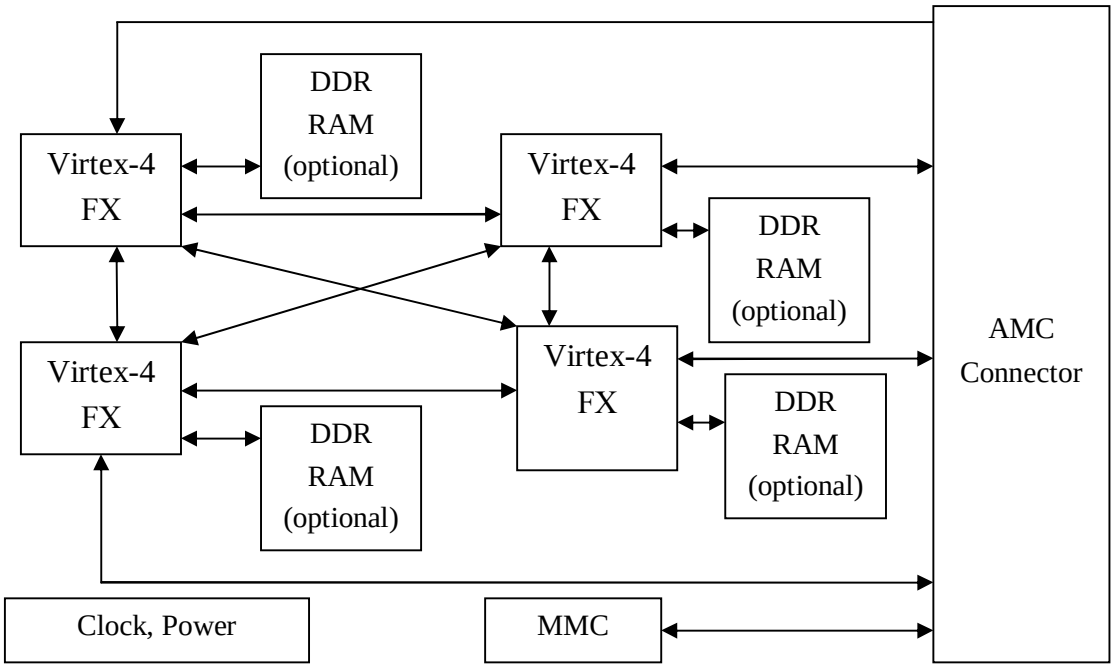


Рис. 2 Аппаратная архитектура AMC модуля

На AMC модули могут устанавливаться ПЛИС LX, SX или FX, обеспечивая значительную гибкость архитектуры системы в зависимости от решаемой задачи, как показано на рис. 2. На каждый AMC модуль устанавливается 4 ПЛИС, MMC контроллер для обеспечения работы по протоколу IPMI.

Внутренняя структура ПЛИС и ее программная архитектура адаптированы для решения широкого круга задач. Два ядра PowerPC получают доступ к встроенной двухпортовой оперативной памяти, контроллеру DMA и свитчеру. Для подключения внешней памяти, расположенной на плате AMC модуля, используется встроенный контроллер DDR SDRAM, как показано на рис. 3. Свитчер, установленный на AMC модуле, взаимодействует со свитчером несущей платы через интерфейс MGT/LVDS.

Структура ПЛИС несущей платы отличается наличием доступа по интерфейсу Gigabit Ethernet и портами ввода/вывода MGT/LVDS, обеспечивающими взаимодействие несущих модулей в шельфе, как показано на рис. 3.

Шельф в стандарте ATCA представляет собой сложную систему, обеспечивающую питание и охлаждение модулей, вентиляцию и межсоединения несущих плат. Существует несколько топологий межсоединений, которые отличаются надежностью, сложностью реализации – в частности, одноранговая звезда, двойная звезда и полная сеть. При использовании топологии двойной звезды (Dual Star) необходима плата центрального свитчера, но скорость обмена может достигать 10 Гбит на одну дифференциальную пару. Если используется топология полной сети (Full Mesh), то скорость обмена ограничивается 1 Гбит на дифференциальную линию, но не требуется центральный свитчер, что удешевляет систему.

Программная модель ПЛИС включает в себя компоненты доступа к вычислительным ядрам DSP, PPC, а также к интерфейсным контроллерам, памяти и внешним AMC-модулям, как показано на рис. 4. Программное обеспечение системы состоит из непосредственно системного встраиваемого ПО для ПЛИС, компонентов конфигурирования, интеграции и управления. Система оборудуется операционной системой реального времени, компоненты которой оптимизированы для работы в многопроцессорной многозадачной среде.

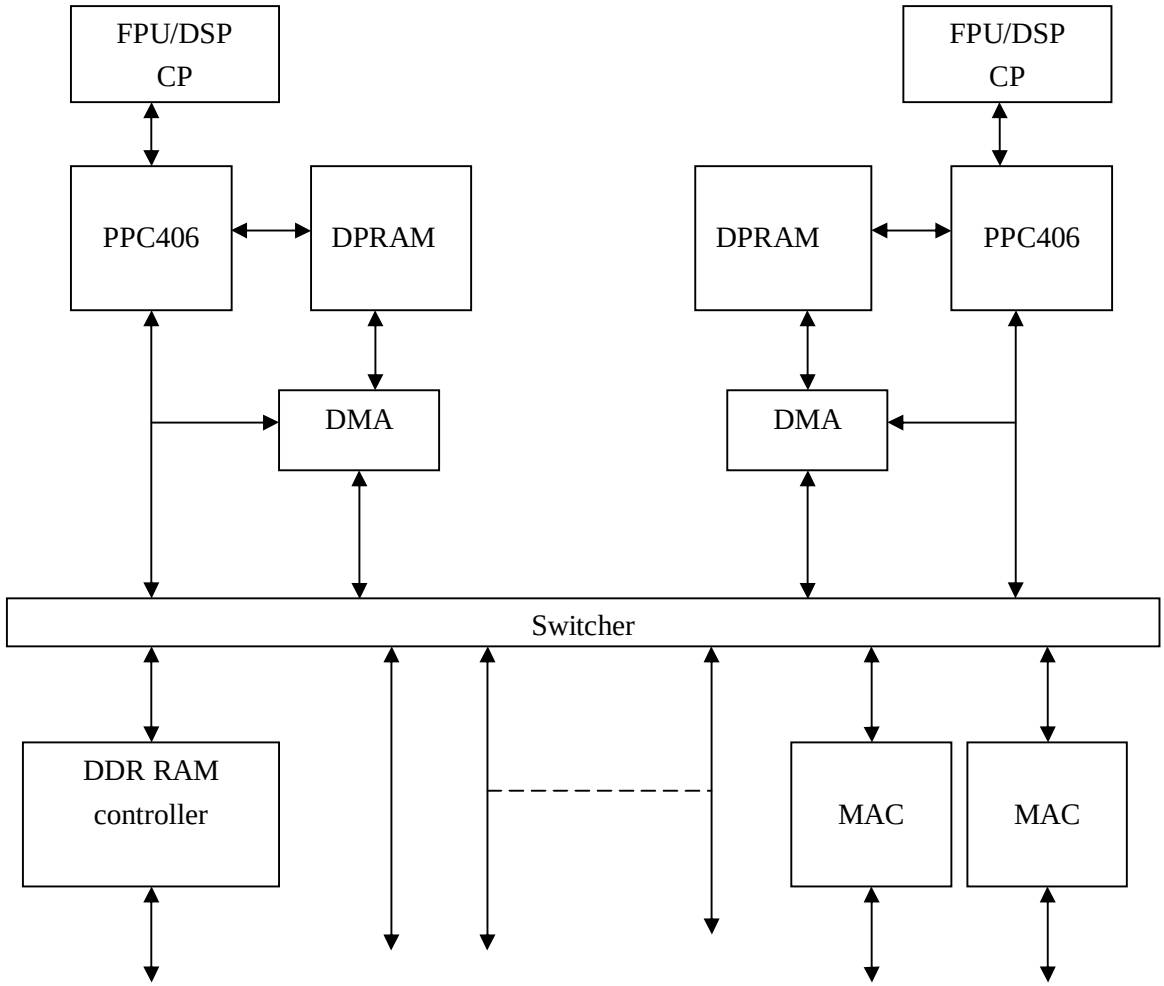


Рис. 3. Вариант структуры ПЛИС AMC-модуля

Отдельно выступают средства отладки приложений, адаптированные к архитектуре системы и поддерживающие ее особенности:

- системное ПО для поддержки функционирования оборудования, которое выполняет задачу интеграции базовых модулей в единый специализированный вычислитель, поддержки совместного функционирования компонентов системы;
- библиотеки параметризируемых IP Cores для ПЛИС, адаптированных к архитектуре системы;
- существующий или разрабатываемый заново wizard для EDK;
- IDE;
- отладочные средства, обеспечивающие расширение и функциональное дополнение, адаптацию к сложной параллельной архитектуре с настраиваемыми параметрами (ChipScope Pro).

Для решения целевых задач используется следующее системное программное обеспечение:

- мультипроцессорная, мультизадачная ОС;
- библиотека функций поддержки IP Cores;
- средства контроля, управления, реконфигурации системы.

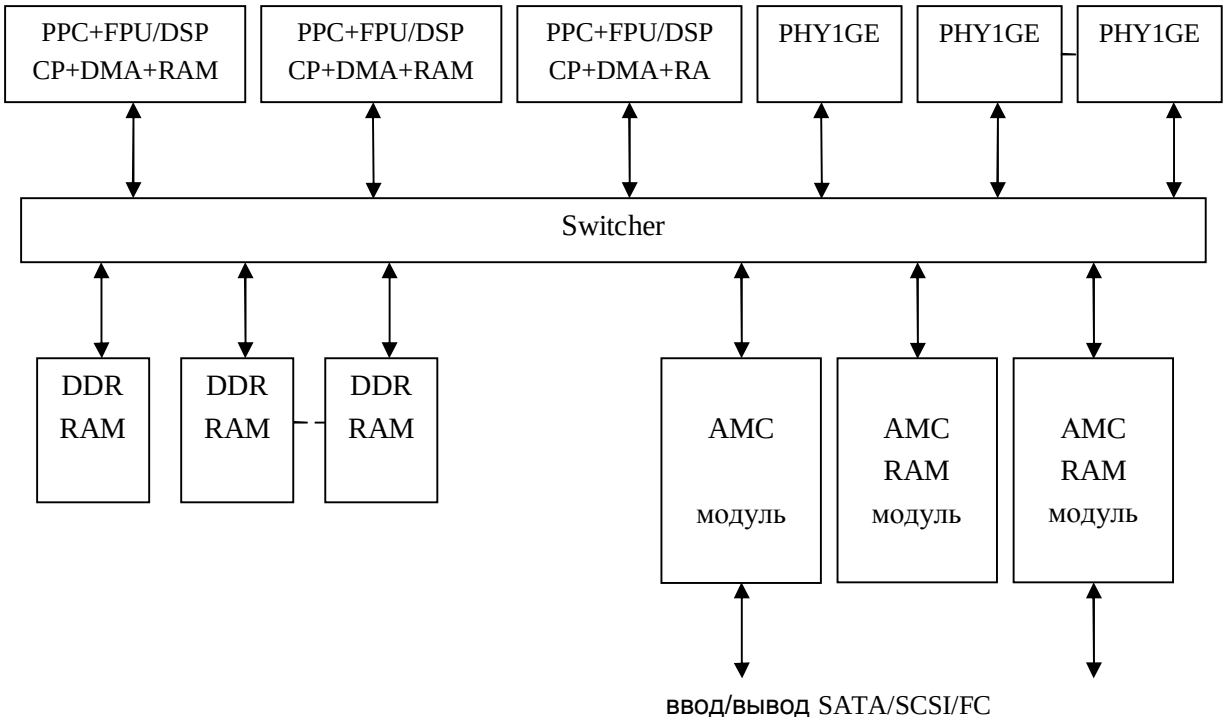


Рис. 4. Программная модель ПЛИС

Литература

1. Семенов А. Технология Advanced TCA. // Телекоммуникации. 2005. №1.
2. Сорокин И. Compact PCI – стандарт, открытый во всех отношениях. // Компоненты и технологии. 2004. №10.
3. Гук М. // КомпьютерИнфо (СПб). 1998. 4–6.
4. Шипулин С.Н., Храпов В.Ю. Особенности проектирования цифровых схем на ПЛИС // Chip News. 1996. № 5. С. 40–43.
5. Шипулин С.Н., Храпов В.Ю. Основные тенденции развития ПЛИС. // Электронные компоненты. 1996. № 3–4. С. 26.
6. Кнышев Д.А., Кузелин М.О. ПЛИС фирмы «Xilinx»: описание структуры основных семейств. М.: Додэка-XXI, 2001.
7. Мальцев П.П., Гарбузов Н.И., Шарапов А.П., Кнышев Д.А. Программируемые логические ИМС на КМОП-структурах и их применение. М.: Энергоатомиздат, 1998.
8. Кузелин М. ПЛИС CPLD компании Xilinx с малым потреблением. Серия CoolRunner. // Компоненты и технологии. 2001. № 5.

МОДЕЛИРОВАНИЕ ЛОГИЧЕСКИХ НЕИСПРАВНОСТЕЙ В ЦИКЛАХ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ ПРОГРАММ

К.В. Петров

Научный руководитель – доктор технических наук, профессор О.Ф. Немолочнов

В статье в качестве метода анализа свойств испытательных последовательностей, используемых при тестировании программ с циклами, предлагается модернизированный метод моделирования логических неисправностей. Изменения в метод были внесены с учетом проблем, возникающих при моделировании неисправностей в программах, содержащих циклы.

Введение

Тестирование и верификация программного продукта могут быть сведены к тестированию вычислительного процесса, порождаемого программой при интерпретации ее команд вычислительной машиной. Вычислительный процесс задается в виде графо-аналитической модели как множество вершин графа и связей между ними. Задачи тестирования и, соответственно, верификации сводятся к обходу вершин и связей с помощью специально подобранных испытательных (тестовых) последовательностей входных наборов и анализу результатов вычислений на этих наборах.

Вне зависимости от методов генерации тестовых последовательностей необходимо проводить анализ контролирующих свойств тестов. С этой целью в рассмотрение было введено понятие логической неисправности. Логическая неисправность – это искажение логики работы программы. В выходных данных программы логическая неисправность будет проявляться в виде ошибки [1].

Для решения задачи анализа тестовых последовательностей было предложено моделировать логические неисправности определенного вида. Моделирование неисправностей сводится к модификации команд условной передачи управления в соответствии с логическими неисправностями путем замены их на команды безусловной передачи управления. Проявление логических неисправностей фиксируется путем сравнения результатов выполнения программы без неисправностей (эталонные результаты) с результатами моделирования работы программы с внесенными в нее неисправностями [2].

Если в процессе тестирования были обнаружены все моделируемые неисправности, или, иначе говоря, если совокупность испытательных последовательностей отвечает условиям полноты контроля заданного множества логических неисправностей, то такая совокупность и образует тест вычислительного процесса программы гарантированного качества [3].

При наличии в тестируемой программе циклов возникают трудности с внесением в нее неисправностей, так как это может привести к появлению бесконечных циклов и связанных с ними аварийных ситуаций. В статье рассматривается данная проблема и предлагается один из возможных способов ее решения.

Метод моделирования логических неисправностей

Логические неисправности разделяют на простые и сложные. Под простой неисправностью понимается такая неисправность, которая не порождает новых дуг на графе вычислительного процесса. Соответственно, к сложным относятся все неисправности, порождающие новые дуги [3].

Для анализа качества тестовых последовательностей следует использовать специально генерируемые логические неисправности особого вида. Это неисправности должны вноситься по очереди в каждую из команд условного перехода, содержащуюся в тестируемой программе. Внесение (моделирование) неисправностей будет, как прави-

ло, сводиться к замене таких команд на команды безусловной передачи управления. В общем случае каждой команде условного перехода (КУП), содержащейся в тестируемой программе, могут соответствовать две моделируемые неисправности:

– неисправность, при которой всегда происходит переход по указанному в команде адресу;

– неисправность, при которой перехода по указанному адресу никогда не происходит, и всегда выполняется команда, располагающаяся в памяти непосредственно за КУП.

Очевидно, что в первом случае условие перехода всегда становится истинным, во втором – ложным. Поэтому будем называть такие неисправности «константная 1» и «константный 0».

Поясним смысл такого моделирования. Замена условного перехода безусловным приводит к исчезновению одной из дуг, выходящей из соответствующей условной вершины на графе, задающем вычислительный процесс тестируемой программы. Отметим, что новая дуга при этом не появляется (за исключением одного особого случая, разбираемого ниже), и потому моделируемые неисправности относятся к простым.

Если тестовая последовательность, качество которой мы оцениваем, не выявит исчезновение дуги, т.е. среди выходных данных не будет ошибочных, то ее следует дополнить набором, подобранным специально с целью обнаружения внесенной неисправности. Можно утверждать, что полнота контроля всех неисправностей такого вида будет определять полноту контроля всех разветвлений вычислительного процесса программы [3]. Фактически в данном методе используются приемы мутационного тестирования (внесения неисправностей) для определения полноты покрытия ветвей [4].

Проблемы при моделировании неисправностей в циклообразующих КУП

Как правило, часть команд условного перехода, содержащихся в программе, используется для организации циклов. Внесение изменений в такие команды может привести к различным последствиям. В одних случаях моделирование неисправностей приведет к исчезновению цикла (т.е. исчезновению обратной петли). Данная ситуация не представляет никакой угрозы, так как понятно, что в этом случае программа благополучно завершится. Однако, если в результате изменения КУП стало «константным 0» условие выхода из цикла, то это означает, что неисправность превратила цикл в бесконечный – наверняка, если это условие выхода было единственным, и потенциально, если существуют еще и другие условия прекращения работы цикла. Теперь, если система моделирования неисправностей запустит данную модификацию тестируемой программы, то с той или иной долей вероятности придется аварийно прекращать работу обеих программ. Такого поведения, конечно, необходимо избегать.

Существует также опасность, что в неконтролируемом цикле произойдет, например, выход за границы массива при записи в него, что приведет к еще более ранней аварийной ситуации. Интересно, что в этом случае – в отличие от предыдущего – моделируемая неисправность является фактически сложной, а не простой. Выход за границы массива можно истолковать как появление новой связи между программой и данными. Таким образом, модификация КУП может в отдельных случаях приводить к внесению в программу сложных логических неисправностей, так как именно новая дуга на графе вычислительного процесса является признаком сложной неисправности.

Итак, мы видим, что при моделировании неисправностей в КУП могут возникнуть аварийные ситуации двух видов:

– для итерационных процессов – бесконечное увеличение времени вычисления; бороться с такими авариями можно с помощью отсечек по времени;

– для переборных процессов – бесконечное увеличение длин массивов, приводящее к искажению и затиранию данных и программного кода.

Возникает закономерный вопрос: нужно ли моделировать такие неисправности? Напомним, что необходимо добиться полноты контроля всех разветвлений программы. С этой точки зрения и следует подойти к решению данной проблемы.

Рассмотрим цикл с предусловием, причем в этом цикле не предусмотрен дополнительный выход (рис. 1).

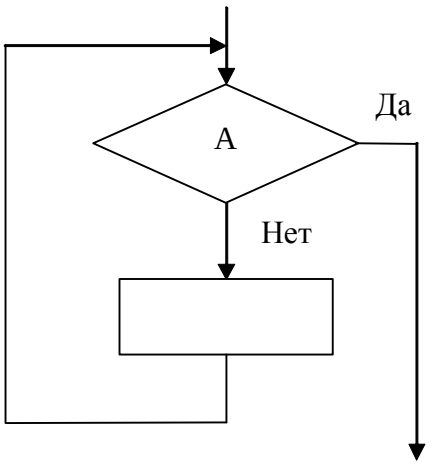


Рис. 1. Цикл с одним выходом

В данном случае неисправность $A \equiv true$ должна быть промоделирована, чтобы проверить, выполняется ли цикл хотя бы однажды. Однако, если цикл выполняется конечное число раз, то в последнем случае условие A будет с неизбежностью истинно. Таким образом, моделировать неисправность $A \equiv false$ не имеет смысла, так как программа после одной или нескольких проверок A все равно попадет на ветвь, где $A \equiv true$. Итак, существует только 2 варианта развития событий, и о них можно судить только по обнаружению неисправности $A \equiv true$.

1) Программа выполнилась, и неисправность $A \equiv true$ не была обнаружена. Это означает, что цикл не был выполнен ни разу.

2) Программа выполнилась, и неисправность $A \equiv true$ была обнаружена, т.е. цикл был выполнен хотя бы раз.

Потому можно утверждать, что в данном случае можно обойтись без моделирования неисправности $A \equiv false$.

Если рассматривать цикл с постусловием, то с точки зрения контроля разветвлений ненужность внесения такого рода неисправности представляется очевидной: цикл в любом случае выполняется хотя бы однажды, и даже моделирование $A \equiv true$ кажется избыточным. Следовательно, в случае цикла с постусловием вообще отпадает необходимость вносить неисправности в КУП, отвечающую за переход к новой итерации.

Но существуют более сложные ситуации, когда обе простые неисправности в циклообразующей КУП должны моделироваться. Рассмотрим следующий пример, где возможны два различных выхода из цикла (рис. 2).

Здесь среди четырех возможных неисправностей существуют одна «проблемная»: $B \equiv true$. На первый взгляд, можно решить, что есть и другая аналогичная неисправность – $A \equiv false$, однако запрет выхода из цикла через проверку условия A не приводит к аварии в данном случае: выход все равно произойдет, когда массив закончится, потому что так сформулировано условие B. Для другого цикла аналогичной структуры возможно, что и неисправность $A \equiv false$ преобразует его в бесконечный.

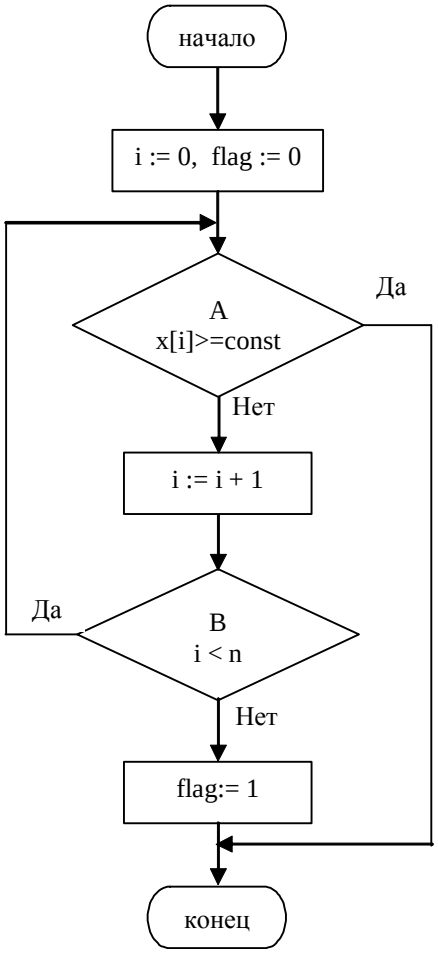


Рис. 2. Цикл с двумя выходами

Внесение неисправности $B \equiv true$ необходимо для достижения полноты контроля ветвлений. Действительно, в приведенном примере существует ветвь, проверить покрытие которой можно лишь путем моделирования данной неисправности.

Пусть, например, в рассматриваемом примере $n = 2, const = 10$, массив $x = \{5, 10\}$. Достаточно ли одного такого тестового набора для достижения полноты покрытия ветвей в данном случае? С точки зрения безопасности мы можем промоделировать неисправности $A \equiv true$, $A \equiv false$ и $B \equiv false$. Покажем, что предложенный тестовый набор проверяет все три эти неисправности. Проверить это можно, проверяя значения выходных переменных; в данном случае это i и $flag$. В таблице приведены значения этих переменных после выполнения исходной программы и трех ее модифицированных вариантов; при этом в качестве входных данных используется массив $x = \{5, 10\}$.

Переменные	R (эталон)	R (A ≡ true)	R (A ≡ false)	R (B ≡ false)
i	1	0	2	1
flag	0	0	1	1

Таблица. Значения выходных переменных при внесении различных неисправностей

Легко видеть, что наборы выходных данных различны во всех четырех случаях. Следовательно, предложенный тестовый набор проверяет каждую из внесенных нами неисправностей. Если мы не станем моделировать новые неисправности, его следует признать достаточным для исчерпывающего тестирования этого фрагмента программы.

Однако очевидно, что данный набор не вызывает выполнение ветви, на которой производится присваивание $flag := 1$. Проверить это можно только путем моделирования опасной неисправности $B \equiv true$. Так как набор $x = \{5, 10\}$ не сможет проверить эту неисправность, то в этом случае результаты выполнения программы не будут отличаться от эталонных. Интересно, что, в свою очередь, проверка этой неисправности означала бы как раз образование бесконечного цикла. Таким образом, именно зацикливание программы должно в данном случае свидетельствовать о высоком качестве тестового набора. Но мы, как уже говорилось, не можем допустить возникновения аварийных ситуаций.

Практический способ решения

Чтобы справиться с проблемами, неизбежно возникающими при модификации условий выхода из цикла, и в то же время иметь возможность моделировать такие неисправности для достижения полноты контроля всех разветвлений тестируемой программы, предлагается использовать следующий способ.

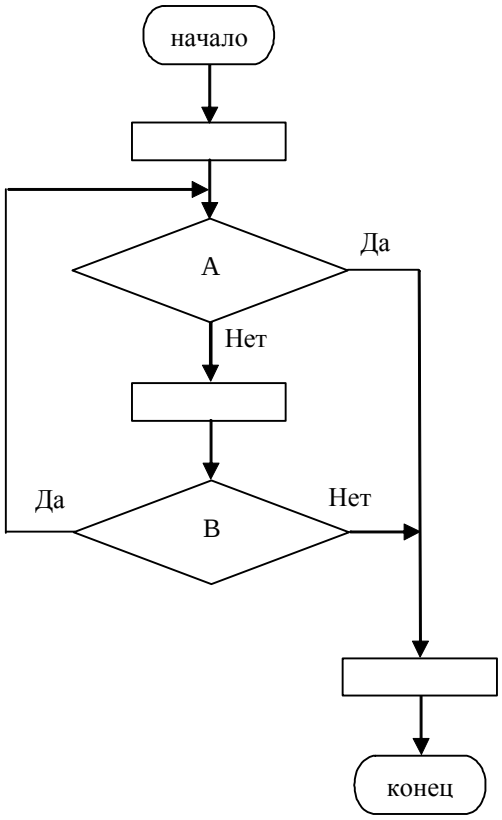


Рис. 3. Исходный цикл

Как мы видели, нам необходимо узнать, проверяет ли данный тестовый набор вносимую неисправность, что эквивалентно зависанию программы. Однако для этого нам не нужно действительно физически навсегда замыкать цикл. Нам нужно только показать, что такое замыкание в данных обстоятельствах неизбежно. Это равнозначно тому, что при завершении последней «штатной» итерации будет произведен очередной переход к началу цикла. Если после этого мы допустим исполнение хотя бы одной команды, то уже не сможем гарантировать безопасность работы всей системы.

Тогда будем иначе моделировать данную неисправность. Это приведет к внесению дополнительных изменений в тестируемую программу, однако и обезопасит ее ис-

полнение. Вместо замены КУП, обслуживающей обратную связь цикла, на команду безусловного перехода мы должны будем вставить в машинный код программы другую команду безусловного перехода, сразу вслед за рассматриваемой КУП. После выполнения этого перехода будет выведено сообщение об обнаруженной неисправности, и модифицированная программа завершится. Следовательно, в том случае, когда выход из цикла происходит именно здесь (т.е. внесение неисправности «константная 1» привело бы к бесконечному циклу), мы узнаем об этом без зависания и аварий. В том же случае, если цикл был покинут другим способом, неисправность обнаружена не будет, что соответствует действительности. На рис. 3 изображен исходный цикл, а на рис. 4 – цикл после внесения неисправности.

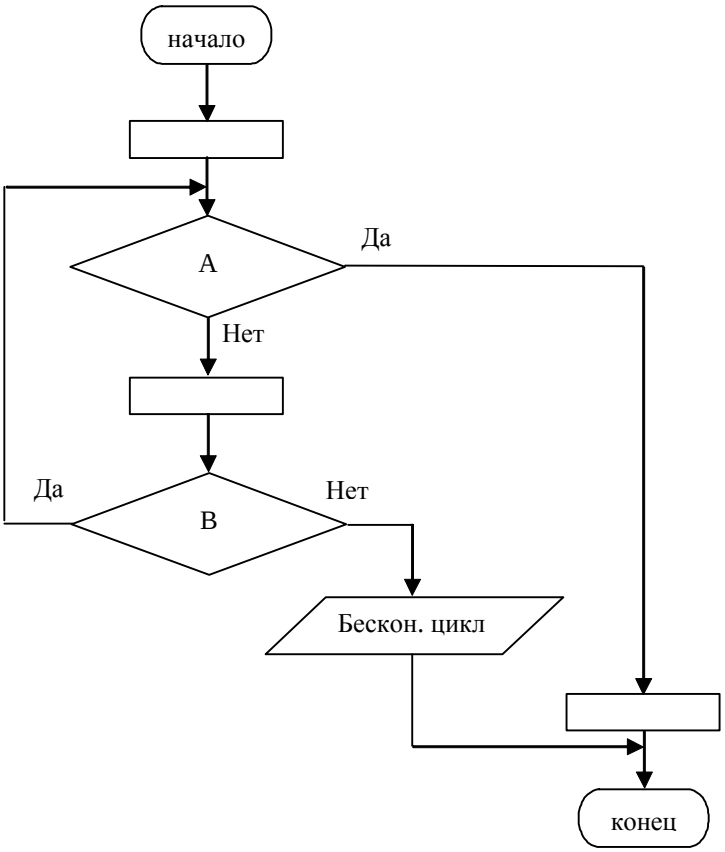


Рис. 4. Цикл с внесенной неисправностью B ≡ true

Итак, было предложено решение, позволяющее моделировать неисправности, аналогичные неисправности $B \equiv true$ в нашем примере. Нужно еще рассмотреть и возможность моделирования неисправностей типа $A \equiv false$ (невыполнение второго условия выхода из цикла), которые, как уже говорилось, тоже могут приводить к авариям. Так как это неисправность «константный 0», то моделирование ее представляется несложным: нужно изменить адрес перехода в модифицируемой КУП таким образом, чтобы после выполнения этого перехода так же, как и в предыдущем случае, на экран выводилось сообщение об обнаруженной неисправности (рис. 5).

Обобщая, можно предложить следующую последовательность действий при моделировании потенциально опасных неисправностей.

1. Необходимо обнаружить все циклы, существующие в программе. Тогда мы узнаем все циклообразующие команды условного перехода и будем моделировать опасные неисправности в них так, как это было сделано выше для неисправности $B \equiv true$.

2. В каждом цикле необходимо найти все КУП, которые контролируют другие возможные точки выхода из данного цикла. После этого мы будем моделировать опасные неисправности в этих командах так, как это было сделано выше для неисправности $A \equiv false$.

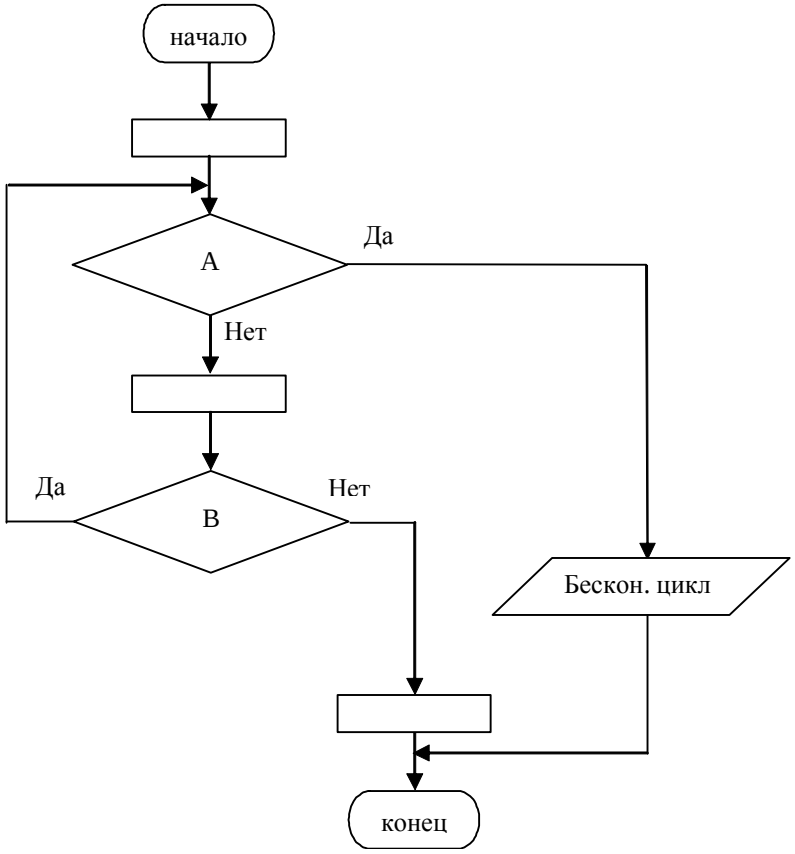


Рис. 5. Цикл с внесенной неисправностью A ≡ false

Заключение

В статье в качестве метода анализа свойств испытательных последовательностей, используемых при тестировании программ с циклами, был предложен модернизированный метод моделирования логических неисправностей. Изменения в метод были внесены с учетом проблем, возникающих при моделировании неисправностей в программах, содержащих циклы.

Литература

1. Немолочнов О.Ф. Методы технической диагностики. Л.: ЛИТМО, 1977.
2. Немолочнов О.Ф., Зыков А.Г., Лаздин А.В., Поляков В.И. Верификация в исследовательских, учебных и промышленных системах. // Научно-технический вестник СПб ГИТМО(ТУ), 2003.
3. Немолочнов О.Ф., Зыков А.Г., Поляков В.И., Кулагин В.С., Петров К.В. Логические неисправности вычислительных процессов программ. // Труды 9-й научно-технической конференции «Майоровские чтения». СПб: СПбГУ ИТМО, 2005.
4. http://www.ise.gmu.edu/~ofut/rsrch/abstracts/practical.html

ДЕКЛАРАТИВНЫЙ ПОДХОД К ВЛОЖЕНИЮ И НАСЛЕДОВАНИЮ АВТОМАТНЫХ КЛАССОВ

А.А. Астафуров

Научный руководитель – кандидат технических наук Д.Г. Шопырин

В работе предлагается декларативный подход к реализации автоматных объектов в рамках современных объектно-ориентированных императивных языков программирования со статической проверкой типов. Отличительной особенностью предлагаемого подхода является возможность использования наследования и вложения макросостояний.

Введение

Всякий раз, когда возникает необходимость реализации автоматов в объектно-ориентированных языках, мы сталкиваемся с различными подходами к решению этой задачи. Подходы к реализации автоматных систем можно разделить на императивные и декларативные.

Наиболее распространенным императивным подходом является паттерн проектирования State Design Pattern [1, 2]. Основными плюсами данного подхода являются распределение специфического поведения между состояниями, а также выполнение переходов в явном виде, непосредственно в коде. Главным недостатком является необходимость создания сложной иерархии при наличии большого числа состояний у автомата, которая может быть устранена при использовании шаблона Decorator Design Pattern [2, 3]. Кроме того, при императивном подходе в коде автоматного объекта присутствуют такие лишние детали, как явное делегирование вызова из контекста конкретному состоянию или вызов вложенного автомата.

Программа является декларативной, если она описывает то, чем она должна быть в конечном итоге, а не то, как именно этого достигнуть. Например, веб-страницы декларативны, так как они описывают то, как страница должна выглядеть (шрифт, размеры, текст, картинки), а не то, как на самом деле она должна быть отображена на экране компьютера. Этот подход сильно отличается от традиционного императивного подхода таких языков, как Fortran, C, Java, которые требуют от программиста конкретного алгоритма для выполнения. Иными словами, императивные программы делают алгоритм явным, оставляя цель неявной. Декларативные программы же делают цель явной, оставляя алгоритм неявным [4]. Существуют подходы, позволяющие реализовывать автоматы в полностью декларативном виде – например, подходы на основе XML. Плюсом таких подходов является полная изоморфность модели, а минусом – то, что они плохо применимы в реальных условиях.

В данной работе предлагается подход, являющийся компромиссом между декларативным и императивным подходами. От декларативного подхода берется возможность декларативного описания структуры автомата, от императивного подхода – описание логики переходов, так как она является частью автоматного кода и является императивной по своей природе.

При помощи декларативного метаязыка можно сконструировать автомат, описав его состояния. При этом достаточно легко удается решать задачу вложения автоматов, избавляя программиста от необходимости в явном виде указывать вызовы вложенных автоматов – все вызовы будут сделаны при помощи специальной библиотеки, контролирующей автомат и осуществляющей перехват вызовов. При помощи декларативного подхода удается также решить задачу наследования, позволяя наследовать автоматы, перекрывая их состояния, которые, в свою очередь, также могут наследоваться. Сама же возможность наследования позволяет по-новому взглянуть на автомат, рассматривая его как самостоятельный объект и применяя к нему принципы объектно-ориентированного дизайна. Все эти качества, совершенно необходимые для более эффективной реа-

лизации автоматов, к сожалению, отсутствуют в известных в настоящее время шаблонах и библиотеках.

Данная статья описывает способ применения декларативного автоматного программирования в современных объектно-ориентированных императивных языках со статической проверкой типов, иллюстрируя описанные подходы примерами программ на языке C#.

Описание подхода

В подходе используется модифицированная модель Statecharts. Основная разница между Statecharts [5] и SWITCH-технологией [6] состоит в том, что в SWITCH-технологии явно вводится понятие автомата. Семантика используемых в SWITCH-технологии графов переходов близка к семантике языка Statecharts, но не эквивалентна ей. В SWITCH-технологии вводится понятия автомата и вложения автоматов, но отсутствуют понятия вложенных и ортогональных состояний. Вложение и ортогонализация состояний в SWITCH-технологии реализуется посредством вложения автоматов и введения понятия «система автоматов» [7]. SWTICH-технология удобнее с точки зрения документации: при использовании вложения автоматов не возникает проблемы с нотацией, как это имеет место при использовании вложенных ортогональных состояний. Так, например, в языке Statecharts возникают трудности с расположением названия макросостояния, содержащего вложенные ортогональные состояния [5].

В данном подходе автомат и вложенные в него состояния рассматриваются как набор макросостояний. Действительно, ведь у состояния нет никакой специфики по отношению к автомату. Макросостояние может включать в себя другие макросостояния с их состояниями. С точки зрения программной реализации библиотеки автоматного программирования между состоянием и автоматом также нет никакой разницы. В рамках библиотеки, разрабатываемой в данном подходе, в системе будет присутствовать класс состояния, который, помимо стандартных свойств, таких как текущее состояние, вложенные макросостояния и т.д., реализует интерфейс, описывающий допустимые входные воздействия. Стоит также отметить, что в диаграмме UML 2 Sate Machine (State Chart в UML 1) [8] понятие автомата на диаграмме так же отсутствует: существуют только состояния, которые могут включать другие состояния, таким образом, решая вопрос вложения автоматов.

Благодаря отождествлению понятий состояния и автомата, а также введению термина «макросостояние», определение вложения следует читать так: «при получении сообщения состоянием это сообщение сначала передается всем его вложенным состояниям, а затем выполняется действие, связанное с этим сообщением». В настоящей работе мы будем считать, что входное воздействие сначала передается всем вложенным состояниям, а затем выполняется действие внутри состояния.

В рамках работы также рассматривается проблема наследования автоматов. Разрабатываемый подход должен позволять наследовать автоматы, переопределяя состояния и правила перехода между ними. Наследование автоматных объектов основано на перегрузке состояний базового автоматного объекта. Производный объект должен перегрузить поведение базового объекта как минимум в одном из его состояний. В производный объект могут быть добавлены новые состояния и переходы между ними [7].

Декларативный подход в рамках императивного языка

Несмотря на императивность многих современных языков программирования, в них тоже можно применять декларативный подход. Для этого необходимо инкапсулировать все недекларативные детали внутри библиотеки или модели, создавая ощущение

декларативности. В зависимости от языка и платформы это может быть достигнуто разными способами. Например, в языке Java и платформе Microsoft .NET существует механизм отражения (*англ. reflection*), позволяющий получать доступ к компонентам класса во время исполнения. Это позволяет давать декларативное описание поведения классов, а затем, во время исполнения, добавлять императивную составляющую.

Для более детального рассмотрения реализации декларативного подхода в императивном языке в качестве примера возьмем язык C# и платформу Microsoft .NET. Наличие такой конструкции, как атрибут, позволяет естественно применить декларативный подход в этом языке. Атрибуты в C# – это конструкции, позволяющие добавлять метаинформацию как к членам класса, так и к самим классам. В дальнейшем информация об атрибутах может быть получена во время исполнения библиотекой, инкапсулирующей недекларативные составляющие программы при помощи отражения.

Рассмотрим механизм выполнения декларативной программы в императивной среде. В рамках платформы .NET возможны следующие основные варианты:

- использование Context Bound Objects – применение встроенного в CLR механизма по перехвату сообщений, поступающих объекту;
- инструментализация сборок – модификация скомпилированного байт-кода с добавлением императивной составляющей, необходимой для выполнения программы в среде Microsoft CLR.

Context Bound Object (привязанный к контексту объект, CBO) – специальный объект, позволяющий контролировать все вызовы, поступающие к нему из контекста. В терминах Microsoft CLR контекст – это окружение, устанавливающее правила, которым будут следовать объекты, в нем находящиеся [9]. Таким образом, если объект привязан к контексту, он также привязан и к его правилам. Контекст создается при активации объекта. Общение объекта с внешним миром происходит при помощи сообщений. Среда Microsoft CLR предоставляет механизмы, позволяющие встраиваться в цепочку обработки сообщений. Это необходимо для их перехвата, а также для генерации или модификации сообщений, проходящих через границу контекста.

Таким образом, при использовании Context Bound Object мы можем контролировать вызовы, а значит, при использовании метаинформации, полученной из атрибутов, мы можем «добавлять» императивную часть в программу во время выполнения. Вся эта работа происходит незаметно для прикладного программиста и реализуется стандартными средствами CLR.

Отметим, что использование CBO оправдано в тех случаях, когда производительность не критична. CBO также выгодно использовать при создании прототипа будущей декларативной системы.

Другим подходом к добавлению императивной составляющей в декларативный код является инструментализация сборок. Инструментализация сборок подразумевает модификацию байт-кода сборки, полученной в результате компиляции. Как и везде, в качестве управляющей информации используются метаданные, добавленные при помощи атрибутов. Однако в этом случае вся работа делается на этапе разработки, а не во время выполнения. Пользуясь метаданными из атрибутов, служебная подпрограмма модифицирует байт-код при помощи Reflection.Emit (для .NET), добавляя в методы вызовы других методов, обеспечивающих конкретную реализацию декларативным директивам.

Данный подход хорош для продуктов, критичных к скорости, так как все дополнительные вызовы делаются напрямую, без дополнительных затрат на управление очередью сообщений, как это происходит в CBO. Явным минусом такого подхода являются трудности при отладке приложения. Из-за того, что байт-код сборки модифицируется, код, полученный в результате этой модификации, начинает расходиться с отладочной информацией, сгенерированной на этапе компиляции.

Реализация на платформе .NET

Для реализации декларативного подхода в императивных языках C# и Visual Basic.NET была разработана библиотека классов DOME (Declarative Object Machines Extension). На рис. 1 приведена диаграмма классов библиотеки DOME.

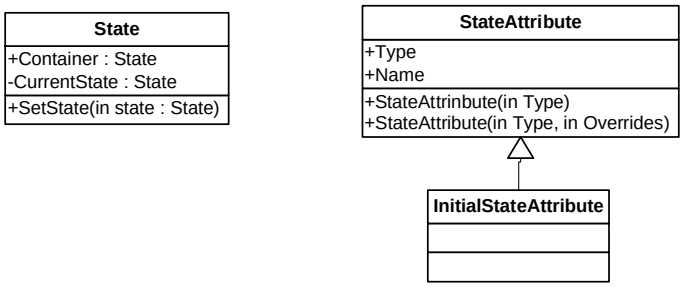


Рис. 1. Основные классы библиотеки

- `State` – базовый класс для состояний.
  - `Container` – экземпляр объемлющего класса State. Данное поле необходимо для доступа к объемлющему состоянию при вложении состояний.
  - `CurrentState` – текущее вложенное состояние. Используется при вложении состояний, а также непосредственно в автомате.
  - `SetState(Type state)` – меняет текущее состояние. В качестве параметра нужно указать тип (класс) состояния, в которое нужно перейти.
- `StateAttribute` – атрибут, применяемый к классам `State` и `Machine`, для описания возможных состояний автомата или объемлющего состояния.
  - `Type` – .NET CLR – тип (класс) состояния
  - `Name` – имя состояния. На данный момент система его автоматически устанавливает в строковое представление значения `Type` для удобства использования атрибута. Считается, что присутствие двух экземпляров одного состояния является бессмысленным, так что конфликта имен состояний при именовании при помощи строкового представления `Type` возникнуть не может.
  - *Конструктор* – Первый конструктор принимает тип (класс) состояния. Перегруженный конструктор имеет два параметра и предоставляет механизм перекрытия состояний. Для этого необходимо в качестве `Type` указать тип (класс) нового состояния, а в качестве `Overrides` указать тип (класс) состояния, которое будет перекрыто. В ходе создания экземпляра атрибута, в случае перекрытия, будет также произведена проверка того, что перекрываемое состояние присутствует выше в иерархии.
- `InitialStateAttribute` – имеет то же значение, что и `StateAttribute`, и применяется для обозначения стартового состояния автомата или стартового вложенного состояния.

В качестве первого примера рассмотрим простейший автомат радара, состоящий из двух состояний: ON и OFF (рис. 2).

Для начала определим 3 интерфейса. Интерфейс `IOn` для состояния ON будет содержать только один метод E1 – т.е. это единственное событие, которое мы обрабатываем, находясь в этом состоянии. По такому же принципу создадим интерфейс `IOff`. Интерфейс `IRadar` будет описывать весь автомат и наследовать интерфейсы `IOn` и `IOff`:

```
public interface IOn
{
void E0();
}

public interface IOff
{
void E1();
}
public interface IRadar : IOn, IOff
{ }
```

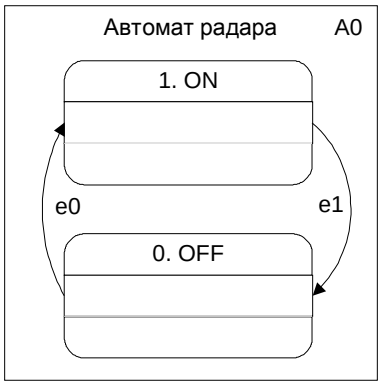


Рис. 2. Автомат радара, состоящий из двух состояний

Далее необходимо реализовать классы состояний ON и OFF:

```
    class On : State, IOn
{
public void E1()
{
Container.SetState(typeof(Off));
}
}
class Off : State, IOff
{
public void E1()
{
Container.SetState(typeof(On));
}
}
```

Поле Container является ссылкой на автомат, которому принадлежит данное состояние. Метод `SetState(Type)` меняет текущее состояние автомата.

Наконец, реализуем сам автомат радара, наследуясь от класса `State` и реализовав интерфейс `IRadar`:

```
[State(typeof(On)), State(typeof(Off))]
class Radar : State, IRadar
{
public void E0() { }
public void E1() { }
}
```

Особое внимание стоит обратить на атрибуты класса Radar. Именно они указывают на то, что автомат `Radar` будет содержать в себе 2 состояния, `On` и `Off`, реализованные в соответствующих классах.

Во время выполнения программы все вызовы к `Radar` будут перехватываться системой и передаваться методу текущего состояния с идентичной сигнатурой, а затем будут выполнен метод самого автомата. В случае отсутствия у текущего состояния метода с идентичной сигнатурой, вызов перейдет непосредственно к методу автомата.

Реализация вложения

Для описания вложения рассмотрим более сложный пример – автомат героя-бойца из компьютерной игры. Боец реагирует на два события – таймер для обновления состояния и нажатие на кнопку «вверх» для прыжка. В прыжке боец меняет состояние: отталкивается, летит, падает, встает на землю.

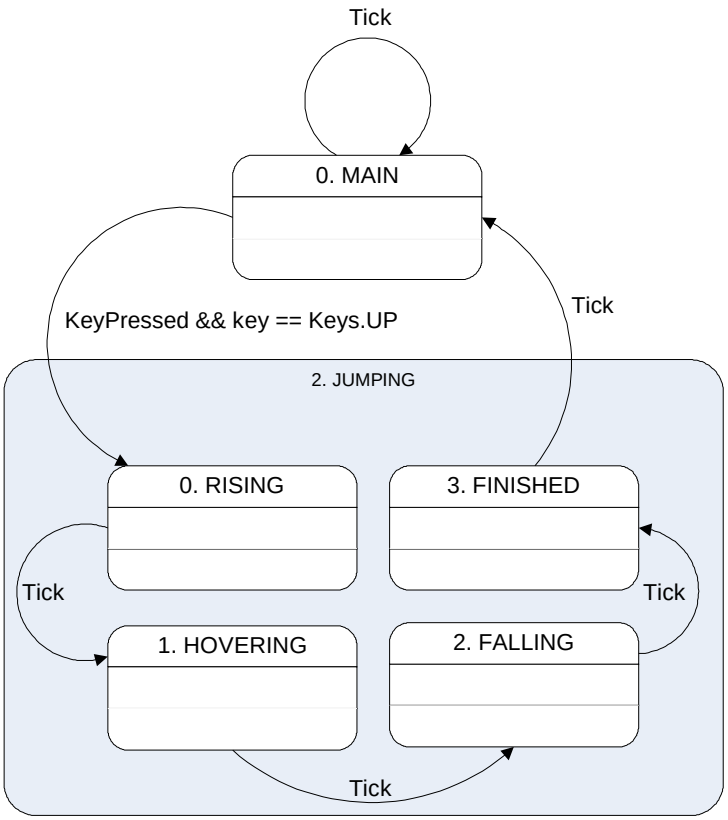


Рис. 3. Автомат бойца с детализацией состояния прыжка

Как видно на рис. 3, состояние `Jumping` имеет вложенные состояния: `Rising`, `Falling`, `Hovering` и `Finished`. Рассмотрим вложение в исходном коде:

```
[InitialState(typeof(Jumping.Rising)),
State(typeof(Jumping.Falling)),
State(typeof(Jumping.Hovering)),
State(typeof(Jumping.Finished))]
public class Jumping : State, IFighter
{
public void Tick()
{
if (CurrentState is Finished)
Container.SetState(typeof(Fighter.Main));
}

public void ButtonPressed(Keys key)
```

```
{}

public bool InAir()
{
return true;
}

public class Rising : State, ITickable
{
public void Tick()
{
Console.WriteLine(“rising”);
Container.SetState(typeof(Hovering));
}
}

public class Hovering : State, ITickable
{
public virtual void Tick()
{
Console.WriteLine(“hovering”);
Container.SetState(typeof(Falling));
}
}

public class Falling : State, ITickable
{
public void Tick()
{
Console.WriteLine(“falling”);
Container.SetState(typeof(Finished));
}
}

public class Finished : State, ITickable
{
public void Tick()
{
//
}
}
}
```

В этом случае событие будет сначала передано текущему вложенному состоянию, а затем обработано соответствующим методом состояния `Jumping`. Как и в предыдущем примере, вызов передается при помощи перехвата сообщения к Context Bound Object.

Реализация наследования

При помощи декларативного подхода удается очень легко и интуитивно понятно описывать наследование автоматов. Для этого достаточно унаследовать класс от базового автомата. Для перекрытия тех или иных состояний необходимо описать это в метаданных. Разумеется, базовые состояния можно наследовать так же, как и автоматы, перекрывая их вложенные состояния или добавляя новые.

В качестве примера поставим задачу перекрытия одного из состояний в автомате `Fighter` из предыдущего примера. Для усложнения задачи предположим, что необходимо перекрыть состояние `Hovering`, вложенное в состояние `Jumping`. Для этого

необходимо создать наследника класса Fighter, назовем его EasternFighter, и атрибутами описать, что мы перекрываем состояние Fighter.Jumping:

```
[State(typeof(EasternJumping), typeof(Fighter.Jumping))]
 public class EasternFighter : Fighter
 {
 }
```

Действительно, перекрывая вложенное состояние Hovering, мы, тем не менее, перекрываем и само состояние Jumping. Теперь необходимо создать наследника класса-состояния Jumping, чтобы, собственно, перекрыть его вложенное состояние Hovering новым состоянием EasternHovering:

```
 [State(typeof(EasternHovering), typeof(Fighter.Jumping.Hovering))]
 public class EasternJumping : Fighter.Jumping
 {
 }
```

Наконец, необходимо реализовать само состояние EasternHovering, создав наследника от Jumping.Hovering.

```
    public class EasternHovering : Fighter.Jumping.Hovering
 {
 public override void Tick()
 {
 Console.WriteLine("Eastern Hovering");
 Container.SetState(typeof(Fighter.Jumping.Falling));
 }
 }
```

Таким образом, когда новый автомат будет вызван, при переходе в состояние Jumping будет использован новый класс EasternJumping, который, в свою очередь, использует перекрытое состояние EasternHovering, о чем сообщает при протоколировании на консоль.

Заключение

В рамках работы был предложен декларативный подход к реализации автоматных объектов. В работе описаны способы реализации декларативного подхода в императивных языках, а также предложена библиотека, позволяющая реализовать данный подход на платформе Microsoft .NET. Использование библиотеки проиллюстрировано примерами реализации наследования и вложения с ее использованием.

Данный подход к реализации автоматов помогает решить проблему совместного использования декларативных и императивных языков, позволяя применять декларативный объектно-ориентированный подход к дизайну автоматов в императивных языках. Благодаря тому, что данная концепция описана на уровне парадигмы декларативного программирования без привязки к конкретному языку программирования, изложенные подходы можно применять и в других языках, позволяющих реализовать декларативный подход, например, Java.

В рамках будущей деятельности планируется также более детально рассмотреть вопрос создания библиотеки, подобной DOME в языке Java, а также подробнее изучить вопрос виртуальных и невиртуальных состояний и их перекрытия.

Литература

1. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. СПб: Питер, 2001.
2. Adamczyk P**.** The Anthology of the Finite State Machine Design Patterns // The 10th Conference on Pattern Languages of Programs, 2003.
3. Odrowski J. and Sogaard P. Pattern Integration – Variations of State // Proceedings of PLoP96.
4. Czarnecky K. and Eisenecker U.W. Generative Programming Methods, Tools and Applications. Addison-Wesley, 2000.
5. Harel D. Statecharts: A visual formalism for complex systems // Sci.Comput. Program. 1987. Vol. 8. P. 231–274.
6. Шалыто А.А. SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб: Наука, 1998.
7. Шопырин Д.Г. Методы объектно-ориентированного проектирования и реализации программного обеспечения реактивных систем./ СПб.: СПбГУ ИТМО, 2005.
8. Буч Г., Рамбо Дж., Джекобсон А. UML. Руководство пользователя./ М.: ДМК, 2000. 432 с.
9. Box D. and Sells C. Essential .NET Vol. 1, The Common Language Runtime. Addison-Wesley, 2002.

АВТОМАТНОЕ ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ДИНАМИЧЕСКИХ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ

О.Г. Степанов

Научный руководитель – кандидат технических наук Д.Г. Шопырин

В работе рассматривается проблема переноса диаграмм переходов, принятых в SWITCH-технологии, в исполняемый код. Для решения данной проблемы предлагается использование динамических языков программирования, возможности которых позволяют добиться изоморфности диаграммы переходов и соответствующего программного кода. Разработан формальный метод преобразования диаграмм переходов в исполняемый код, а также практическая реализация этого метода в виде библиотеки STROBE на языке Ruby.

Введение

Для проектирования и разработки реактивных систем часто используется SWITCH-технология, также известная как «автоматное программирование» или «программирование с явным выделением состояний» [1]. Одной из основных частей SWITCH-технологии является графический язык, позволяющий описывать поведение систем со сложным поведением в терминах состояний и переходов между ними и связей между этими системами.

При использовании SWITCH-технологии в разработке программного обеспечения важной частью разработки является реализация поведения, описанного на диаграммах переходов, на целевом языке программирования. Для решения этой задачи традиционно используется один из следующих трех подходов.

1. *Полностью ручное программирование*. Одним из простейших общепринятых методов ручного программирования является следующий: текущее состояние системы хранится в переменной интегрального или перечислимого типа, а основная логика программы сосредоточена внутри одного или нескольких операторов switch, определяющих действия программы в зависимости от текущего состояния [2]. Другим популярным методом является использование паттерна программирования State, впервые описанного в [3]. Несмотря на такие преимущества этой группы методов, как высокая производительность и полный контроль над получаемым кодом, она имеет существенные недостатки – низкая читаемость кода и большая трудоемкость.
2. *Автоматическая генерация кода по диаграмме переходов*. Обычно при этом подходе генерируется код, аналогичный тому, который получается при использовании ручного программирования. Недостатками этого подхода являются низкая читаемость кода (связанная с тем, что в качестве целевого языка используется императивный язык, например Java, и при переносе теряется информация, специфичная для логики диаграмм переходов), малая степень контроля над получаемым кодом, невозможность ручного изменения получаемого кода и привязанность к конкретному формату входных данных, с использованием которого задается исходная диаграмма переходов (формат файлов Visio, XML [4, 5] или другой).
3. *Ручное написание кода с использованием специальной библиотеки*. В этом случае происходит перенос диаграммы переходов в вызовы специальной библиотеки, которая по этим инструкциям строит внутреннее представление диаграммы переходов. Затем, согласно этому представлению, происходит исполнение логики описанного автомата. Основным преимуществом подхода является то, что вызовы библиотеки отражают семантику диаграммы переходов (каждый вызов может, например, соответствовать объявлению состояния или перехода), что позволяет создавать читаемый код, который легко поддерживать. Также некоторые библиотеки ориентированы на конкретные виды взаимодействия автоматного и объектно-ориентированного кода, что позволяет более гладко комбинировать эти подходы к програм-

мированию. Основными недостатками являются низкая производительность некоторых из таких систем (существуют системы, основанные на метапрограммировании и статической генерации кода [6, 7], которые позволяют достичь высокой производительности), а также невозможность описать ряд конструкций диаграмм переходов, используя ограниченный синтаксис целевого языка.

В настоящей работе предлагается развитие третьего подхода, расширяя его с точки зрения «качества» и читаемости получаемого кода. Разработанный подход использует динамические языки программирования, т.е. языки, позволяющие менять и дополнять код программы во время выполнения. В работе показано, как свойства динамических языков могут быть использованы для увеличения читаемости кода, строящего модель системы, особенно в части формирования условий переходов.

Для формализации процесса перенесения диаграмм переходов в код была разработана операционная семантика (правила интерпретации) диаграмм переходов SWITCH-технологии. Предложенная семантика позволяет верифицировать правильность воспроизведения спроектированного поведения.

Материал статьи организован следующим образом: в первом разделе дано краткое описание SWITCH-технологии, предложена операционная семантика диаграмм переходов и описана проблема их переноса в исполняемый код. Во втором разделе предложен подход к решению этой проблемы, базирующийся на формальном преобразовании диаграмм переходов в исполняемый код с использованием свойств динамических языков программирования, а также описана библиотека STROBE, реализующая разработанный подход.

1. Операционная семантика диаграмм переходов

Одна из проблем при использовании SWITCH-технологии, возникающая при реализации поведения, описанного на диаграммах состояний, на различных языках программирования – обеспечение корректности переноса логики системы при сохранении исходных обозначений и структуры описания.

Рассмотрим пример использования диаграмм переходов. В работе [8] описана система управления лифтом, основанная на автоматах, в ходе реализации которой был разработан автомат управления кнопкой вызова лифта на нижнем этаже (*A11*). Его диаграмма переходов представлена на рис.

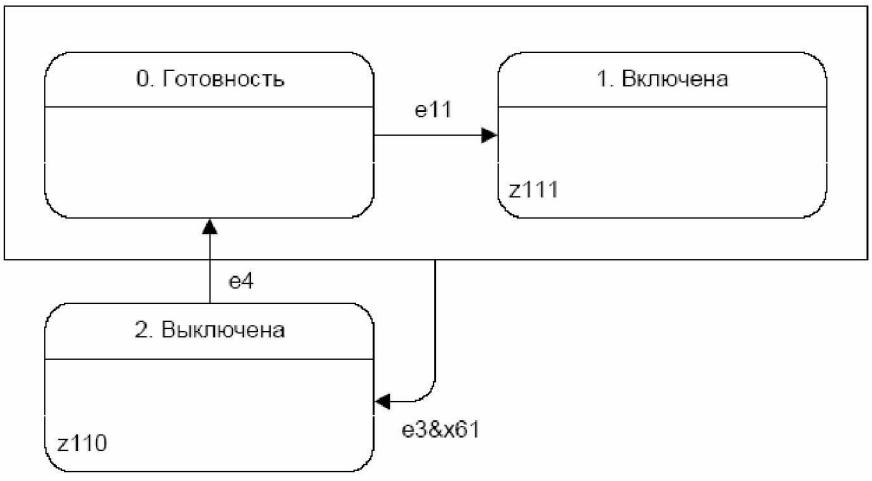


Рис. Диаграмма переходов автомата управления кнопкой вызова лифта

Из диаграммы видно, что этот автомат имеет три состояния: «Готовность» (кнопка на этаже подсвечена, ее можно нажать), «Включена» (кнопка нажата) и «Выключе-

на» (кнопку нажать нельзя). Переход в состояние «Включена» происходит по нажатию кнопки (событие *e11*), переход в состояние «Готовность» – по событию *e4* (разрешение на включение лампы в кнопке) от головного автомата, переход в состояние «Выключена» – по событию *e3* (выключение лампы в кнопке) от головного автомата при условии, что лифт находится на нижнем этаже (определяется переменной *x61*).

Для обеспечения корректности переноса логики требуется формально описать семантику диаграмм переходов и метод их преобразования в исполняемый код. Это позволит верифицировать получаемый код на соответствие логике исходной диаграммы. В условиях сложного графического синтаксиса диаграмм оптимальным описанием является создание операционной семантики. Ниже предлагается операционная семантика для диаграмм переходов в SWITCH-технологии, аналогичная семантике UML-диаграмм состояний, описанной в работе [9].

Для начала определим основные свойства диаграмм переходов.

1. Диаграмма изображает одно или несколько состояний системы и переходы между ними.
2. Состояния на диаграмме могут быть объединены в группы; группы могут быть вложены друг в друга; состояния внутри группы равноправны.
3. Переходы могут начинаться в состоянии или в группе состояний, заканчиваться только в состоянии (переходы, начинающиеся в группе состояний, называются *групповыми переходами*); переходы могут начинаться и заканчиваться в одном и том же состоянии.
4. Каждое состояние помечено следующими атрибутами:
   a. имя состояния,
   b. номер состояния (нумерация начинается с нуля),
   c. действия по входу в состояние,
   d. вложенные автоматы (возможно с номерами воздействий, с которыми они вызываются).
5. Переход может быть помечен следующими атрибутами:
   a. условие перехода,
   b. действия на переходе,
   c. приоритет перехода.

Обработка события происходит следующим образом: в качестве текущего события выставляется обрабатываемое. Затем перебираются переходы, выходящие из текущего состояния и содержащих его групп в порядке приоритета (сначала рассматриваются переходы с меньшими номерами). Для каждого перехода вычисляется условие, которое является логической формулой. Эта формула может использовать следующие переменные:

1. *ei* – имеет значение «истина», если текущее событие – *ei*, и «ложь» иначе;
2. *yi* – имеет значение равное номеру состояния автомата *Ai*;
3. *xi* – значение переменной *xi*.

Выполняется первый переход (взятый в указанном выше порядке), для которого значение условия истинно. Выполнение перехода состоит из следующих шагов:

1. выполняются действия на переходе (вызываются указанные выходные воздействия в порядке их следования на диаграмме);
2. текущим выставляется состояние, в котором заканчивается переход;
3. если произошла смена состояния (текущее состояние до начала обработки события отличается от состояния, в котором заканчивается переход), вызываются действия по входу в состояние;
4. вызываются вложенные автоматы: если для автомата указан номер события, он вызывается с этим событием; иначе – с событием *e0*.

Для описанного выше автомата A11 применение семантики дает следующие инструкции:

1. в качестве текущего установить обрабатываемое событие;
2. если текущее состояние – «Готовность» и текущее событие – *e11*, то осуществляется переход в состояние «Включена», при этом производится выходное воздействие *z111* (включить лампу в кнопке);
3. если текущее состояние – «Готовность» или «Включена», текущее событие – *e3* и при этом значение переменной *x61* – «истина», осуществляется переход в состояние «Выключена», при этом производится выходное воздействие *z110* (выключить лампу в кнопке);
4. если текущее состояние – «Выключена» и текущее событие – *e4*, производится переход в состояние «Готовность».

2. Реализация автоматных систем на языке Ruby

Для решения задачи переноса диаграмм переходов в исполняемый код предлагается использовать динамические языки программирования. Отличительными свойствами этих языков, позволяющими упростить перенос диаграмм переходов, являются следующие:

1. возможность динамической генерации кода;
2. возможность использования замыканий (совокупностей функций и данных в данном лексическом контексте).

С помощью динамической генерации кода можно создавать индивидуальный для каждой диаграммы переходов код выполнения автомата, что позволит увеличить производительность, а совместно с замыканиями позволяет переносить условия переходов в код на целевом языке практически без изменений.

Для практической реализации предложенного подхода был выбран язык Ruby [10], разработанный Юкихиро Мацумото (Yukihiro Matsumoto). Это интерпретируемый динамический язык программирования, обладающий следующими свойствами:

1. простой синтаксис;
2. объектная ориентированность;
3. поддержка механизма mixin-ов через включение модулей;
4. поддержка замыканий;
5. легкая переносимость (среда исполнения Ruby может быть запущена под большинством современных операционных систем);
6. возможность интеграции с кодом на других языках программирования.

Приведем исполняемый код,, соответствующий приведенному выше примеру, на языке Ruby с использованием библиотеки STROBE, описанной ниже.

```
# Подключение библиотеки STROBE
require 'strobe/automaton'
module Elevator
  # Декларация класса автомата
  class A11 < Strobe::Automaton
    # Декларация внутренней переменной x61
    attr_accessor :x61
    # Декларация событий
    inputs :e3, :e4, :e11
    # Декларация выходных воздействий
    outputs :z110, :z111
    # Начало группы состояний
    begin_group
      # Состояние "Готовность"
      state :ready
```

```
      # Переход в состояние "Включена"
      transition :to => :on,
                 :if => lambda { e11 }
    # Состояние "Включена"
    state :on,
          :output_actions => :z111
    # Групповой переход в состояние "Выключена"
    group_transition :to => :off,
                     :if => lambda { e3 && x61 }
  end_group
  # Состояние "Выключена"
  state :off,
        :output_actions => :z110
    # Переход в состояние "Включена"
    transition :to => :ready,
               :if => lambda { e4 }
 end
end
```

В этом коде внутри класса *A11*, объявленного как автоматный путем наследования от библиотечного класса Strobe::Automaton, последовательно определена диаграмма состояний автомата *A11*. Инструкции *inputs*, *outputs*, *state*, *transition* и *group_transition* объявляют события, выходные воздействия, обычный и групповой переход, соответственно. Инструкции *begin_group* и *end_group* объединяют все декларированные между ними состояния в группу.

Представленный выше пример был построен с использованием формального метода, позволяющего перенести любую диаграмму переходов в исполняемый код на Ruby. Схема метода – следующая:

1. для каждой диаграммы создается новый класс с именем, совпадающим с именем автомата (возможно использовать модули чтобы избежать конфликтов имен);
2. внутри класса с помощью специальных конструкций описывается диаграмма переходов (каждое выражение соответствует описанию одной сущности на диаграмме: состоянию, переходу, группе состояний и т.д.);
   2.1. при описании состояний им присваиваются идентификаторы, соответствующие их именам на диаграмме;
   2.2. при описании переходов ссылки на состояния производятся по именам;
   2.3. при описании перехода условие перехода переносится с диаграммы в виде выражения, использующего те же самые переменные, что и исходное выражение на диаграмме;
3. класс автомата связывается с другими подсистемами через подписку на выходные воздействия.

Ключевыми моментами использования языка Ruby являются полностью декларативное описание поведения системы и автоматическая генерация специальных методов, позволяющих переносить диаграмму переходов с минимумом изменений. Покажем это на примере. Допустим, в диаграмме есть переход, условие которого задается формулой $e1 \vee e0 \wedge (x1 \vee (y2 = 3))$. В коде программы он будет представлен в виде выражения e1 || e0 && (x1 || (y2 == 3)). При вычислении условия перехода это выражение будет вычислено в контексте автоматного класса, что позволит использовать сгенерированные по описанию автомата (списку входных и выходных воздействий, связям с другими автоматами) методы e0, e1, x1 и y2. Генерируются следующие методы:

1. для каждого входного воздействия – метод с именем воздействия, который возвращает значение «истина», если сейчас автомат обрабатывает соответствующее воздействие, и «ложь» во всех других случаях;

2. для каждой внутренней переменной – метод с именем переменной, который возвращает текущее ее значение;
3. для других автоматов в системе – переменная с префиксом y, возвращающая номер состояния соответствующего автомата.

При попытке практической реализации предложенного подхода возникает ряд проблем:

1. обеспечение возможности вызова методов описания диаграммы переходов непосредственно в теле класса, наряду со стандартными декларациями языка;
2. обеспечение выполнения условий переходов в нужном контексте;
3. поддержка наличия нескольких экземпляров одного и того же автомата и обеспечение связи между автоматами через переменные *yi*.
4. поддержка интеграции с программами на других языках, а также обеспечение возможности управления физическими устройствами;
5. обеспечение приемлемой производительности получаемого кода.

Автором разработан метод описания автоматов, решающий большинство поставленных проблем. Данный метод реализован в библиотеке STROBE, которая описана ниже.

Разработана библиотека STROBE, позволяющая декларативно задавать диаграммы переходов на языке Ruby. Для программиста эта библиотека предоставляет дополнительный набор инструкций (реализованных в виде методов), позволяющих поэлементно определять диаграммы переходов (в число инструкций входят, например, *state* для определения состояния и *transition* для определения перехода). Методы используют технологию именованных параметров (при вызове инструкции значение каждого параметра связывается с параметром явно по имени), что позволяет увеличить читаемость кода (примерами имен параметров, использованных в примере выше, могут являться *:to* или *:output_actions*).

Процесс описания автомата при использовании библиотеки STROBE соответствует описанному в предыдущем пункте, а также имеет следующие особенности:

1. внешняя конфигурация автомата (входные и выходные воздействия, связи с другими автоматами) описывается в начале класса;
2. состояния описываются последовательно, в порядке их нумерации на диаграмме переходов;
3. переходы из данного состояния описываются непосредственно после описания самого состояния;
4. связь с другими компонентами осуществляется путем подписки на конкретные выходные воздействия;
5. связь с другими автоматами по переменным состояния (*yi*) осуществляется путем явного связывания экземпляров автомата и имен переменных в блоке описания внешней конфигурации автомата;
6. для выполнения нескольких экземпляров одного автомата независимо используются система доменов, описанная ниже.

Библиотека позволяет перенести в код на Ruby любую синтаксически верную диаграмму переходов, а также перенести несколько автоматов и связать их. Возможна также интеграция с модулями на других языках программирования, в том числе модулями управления физическими объектами. При этом описание автомата изоморфно диаграмме состояний и понятно без дополнительных инструкций и описаний.

Для обеспечения независимого выполнения нескольких экземпляров автомата используется система доменов – логических областей, к одной из которых может быть приписан экземпляр автомата при создании. Каждый экземпляр автомата имеет уникальный в пределах домена идентификатор, по умолчанию равный имени класса. Та-

ким образом, ссылаться на другие автоматы можно по имени класса, используя стандартные алгоритмы разрешения ссылок языка.

Производительность решения находится на одном уровне с динамически выполняемыми решениями третьей группы (интерпретирующими внутреннее представление диаграммы переходов). Этот показатель можно улучшить путем полной генерации кода обработки входных воздействий, что невозможно осуществить в рамках языка Ruby, сохранив читаемость кода, в связи с отсутствием доступа к модели кода и невозможностью производить подстановку замыканий в сгенерированный код. Тем не менее, по всем остальным показателям предложенный подход решает проблемы, описанные в конце третьего раздела.

Заключение

В настоящей работе рассматривается проблема переноса диаграмм переходов автоматов, разработанных согласно SWITCH-технологии, в исполняемый код. Рассмотрены основные направления решения этой проблемы, предложен подход, развивающий одно из этих направлений. Разработан формальный метод преобразования диаграмм переходов в исполняемый код, описаны отличительные свойства этого метода – декларативная структура кода, его изоморфность исходной диаграмме, в особенности в области задания условий переходов. Описаны основные проблемы, возникающие при практической реализации разработанного подхода, и предложена конкретная реализация в виде библиотеки STROBE на языке Ruby, решающая большинство этих проблем.

Литература

1. Шалыто А.А. SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб: Наука, 1998. 628 с.
2. Шалыто А.А., Туккель Н.И. Реализация автоматов при программировании событийных систем // Программист, 2002. №4. С. 74–80.
3. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. СПб: Питер, 2001. 368 с.
4. Бондаренко К.А., Шалыто А.А. Разработка XML–формата для описания внешнего вида видеопроигрывателя с использованием конечных автоматов. СПб: СПбГУ ИТМО. 2003. // http://is.ifmo.ru, раздел «Статьи».
5. Gurov V.S., Mazin M.A., Narvsky A.S., Shalyto A.A. UniMod: Method and Tool for Development of Reactive Object-Oriented Programs with Explicit State Emphasis // Proceedings of St. Petersburg IEEE Chapters, 2005. V.2. P. 106–110.
6. Abrahams D., Gurtovoy A. C++ Template Metaprogramming. Addison Wesley, 2004.
7. Шопырин Д.Г., Шалыто А.А. Объектно-ориентированный подход к автоматному программированию // Информационно-управляющие системы, 2003. № 5. С. 29–39.
8. Наумов А.С., Шалыто А.А. Система управления лифтом. // http://is.ifmo.ru, раздел «Статьи».
9. Гуров В.С., Мазин М.А., Шалыто А.А. Операционная семантика UML–диаграмм состояний в программном пакете Unimod /// Тез. докл. Всероссийской научно-метод. конф. «Телематика’2005».
10. Thomas D., Fowler C., Hunt A. Programming Ruby. Second Edition. Pragmatic Bookshelf, 2004.

ОТНОШЕНИЕ НАСЛЕДОВАНИЯ ДЛЯ ТИПОВ СО СЛОЖНЫМ ПОВЕДЕНИЕМ

Н.И. Поликарпова
Научный руководитель – кандидат технических наук Д.Г. Шопырин

В работе рассматривается объектно-ориентированный подход к моделированию и спецификации сущностей со сложным поведением; строится определение типа со сложным поведением, предлагается метод формальной спецификации и определение отношения наследования для таких типов.

Введение

В процессе создания программного обеспечения часто возникает необходимость реализации сущностей со *сложным поведением*. Таким свойством обладают устройства управления, сущности, отвечающие за взаимодействие других сущностей между собой и с пользователем, сущности, реализующие сетевые протоколы, и многие другие.

Можно сказать, что объект обладает сложным поведением, если в ответ на возникновение некоторого *события* он может совершить одно из нескольких *действий*. Выбор действия зависит от информации, хранимой внутри и вне данного объекта на текущем шаге вычисления, а также на всех предыдущих шагах. При наивной программной реализации таких сущностей возникают избыточные переменные, называемые *флагами*, которым не соответствуют никакие элементы предметной области. Единственное предназначение флагов – участвовать в многочисленных, запутанных конструкциях ветвления, реализующих логику поведения. Это решение трудно для понимания, подвержено ошибкам и практически не расширяемо.

Вместо этого предлагается описывать объект со сложным поведением, приписав ему некоторое множество *управляющих состояний*, в каждом из которых поведение объекта является «простым» и может быть описано непосредственно. Вся логика поведения оказывается сосредоточенной в механизме *переходов* между состояниями. Связь управляющих состояний с поведением и механизм переходов удобно описывать с помощью *конечного автомата* с выходами [1]. Поэтому такой подход к описанию сложного поведения часто называют *автоматным*.

Те или иные модификации диаграмм переходов конечных автоматов применяются для описания сложного поведения в ряде известных методологий спецификации компьютерных систем, таких как UML [2], Statemate [3], SWITCH-технология [4]. Две последних из них не являются объектно-ориентированными. Например, в подходе Statemate в качестве первого шага спецификации концептуальной модели системы предлагается осуществить ее *функциональную декомпозицию* – представить систему в виде набора взаимосвязанных *деятельностей*. После этого поведение всей системы в целом предлагается описать в терминах деятельностей при помощи диаграммы Statechart, которая является компактным описанием конечного автомата.

Язык моделирования UML, напротив, основан на объектно-ориентированных концепциях. В объектно-ориентированной методологии основным видом декомпозиции является *объектная* – представление системы в виде набора взаимодействующих *абстрактных типов данных* (АТД). В UML для обозначения этой абстракции используется термин *класс*, однако автор предпочитает применять этот термин по отношению к реализации абстракций в некотором объектно-ориентированном языке программирования. Несмотря на основополагающую разницу в методологии, UML предполагает использование диаграмм Statechart тем же образом, что и Statemate – для описания поведения всей системы в целом или произвольной ее части.

С точки зрения автора, действительно объектно-ориентированным подходом к описанию сложного поведения было бы выделение из совокупности АТД, описываю-

щей систему, типов, моделирующих сущности со сложным поведением, и применение автоматного подхода для описания поведения каждого из этих типов в отдельности. Таким образом, мы приходим к понятию *автоматизированного абстрактного типа данных* (ААТД) – типа данных, содержащего конечный автомат, управляющий его поведением.

При спецификации компьютерной системы с использованием автоматизированных типов данных сложное поведение инкапсулируется – пользователь может обращаться с ААТД так же, как с обыкновенным типом. Это позволяет применять для построения системы как целого привычные объектно-ориентированные приемы, такие как порождение подтипов и композиция объектов. Предлагаемый подход способствует большей модульности специфицируемой системы и повторному использованию реализованных типов со сложным поведением.

В дальнейших разделах настоящей работы вводится понятие автоматизированного абстрактного типа данных. Предлагается метод спецификации ААТД, основанный на сочетании существующих языков спецификации типов данных и конечных автоматов, и описывается алгоритм преобразования спецификации ААТД в спецификацию обыкновенного типа данных. Эквивалентность двух спецификаций позволяет легко переносить на ААТД свойства и отношения, характерные для обыкновенных типов данных. Из всех рассматриваемых обычно отношений между типами отношение «*является подтипом*» (или отношение *наследования*) трактуется наиболее неоднозначно, несмотря на то, что оно повсеместно используется на практике. В статье [5] были даны два формальных определения отношения наследования. В настоящей работе рассматривается применение одного из них к автоматизированным типам данных.

Понятие автоматизированного абстрактного типа данных

Прежде всего, уточним, что понимается в настоящей работе под абстрактным типом данных.

Определение 1. *Абстрактным типом данных* называется пара

$\tau = \langle T, M \rangle$,

где T – множество значений (или вычислительных состояний), $M = \{m \mid m = \langle i, Args, Ret, f \rangle\}$ – конечное множество методов. Каждый метод включает в себя имя i, набор типов аргументов $Args$, тип возвращаемого значения Ret и, наконец, исполняющую функцию $f : T \times \{Obj : Args\} \to \{T \times \{Obj : Ret\}\}^*$ (запись $\{Obj : Types\}$ означает множество всех объектов, принадлежащих к соответствующим типам или типу).

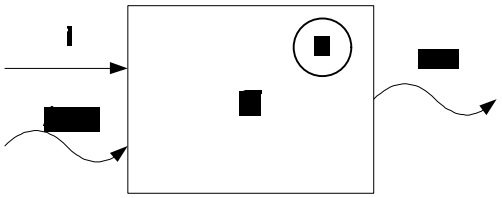


Рис. 1. Абстрактный тип данных

Абстрактный тип данных схематично изображен на рис. 1. Совокупность отвечающих методу типов $Args$ и Ret определяет его сигнатуру, а исполняющая функция – его семантику. Как следует из определения, исполняющая функция не всегда однозначно определяет новое вычислительное состояние и возвращаемое значение. Для некоторых значений аргументов она может быть не определена, а для других существует

несколько вариантов результата (это оставляет определенную свободу реализации АТД в программе).

Далее будем обозначать метод с именем i через m_i, типы его аргументов и возвращаемого значения – через $m_i.Args$ и $m_i.Ret$, соответственно. Если необходимо явно указать его принадлежность к множеству методов абстрактного типа τ, будем писать $\tau :: m_i$.

Если абстрактный тип данных является моделью сущности со сложным поведением, исполняющие функции его методов обычно трудны для понимания и спецификации. В терминах данного выше понятия сложного поведения они содержат описания нескольких простых действий, объединенные условными конструкциями. В соответствии с автоматным подходом можно специфицировать простые функции-действия отдельно и приписать их типу *управляющий автомат*, задача которого – осуществлять диспетчеризацию вызовов функций-действий в зависимости от имени метода (события) и от управляющего состояния. Чтобы различать функции-действия, удобно присвоить им собственные имена. Именованные функции-действия образуют множество методов некоторого нового типа данных, который, по отношению к первичному типу, моделирующему сущность со сложным поведением, будем называть *вложенным*. Таким образом, модель сущности со сложным поведением представляется в виде совокупности *управляющей* и *вычислительной* компонент. Эта конструкция получила название *разомкнутого* автоматизированного абстрактного типа данных (его отличие от замкнутого ААТД обсуждается ниже).

Определение 2. *Разомкнутым автоматизированным абстрактным типом данных* будем называть пару $\langle A,\tau \rangle$, где A – управляющий автомат, τ – вложенный абстрактный тип данных.

Управляющий автомат представляет собой пятерку

$$A = \langle S, E, Z, s_0, \Delta \rangle,$$

где S – конечное множество управляющих состояний, E – конечное множество входных символов (или событий), Z – конечное множество действий, $s_0 \in S$ – начальное (стартовое) состояние, $\Delta : S \times E \to S \times Z$ – управляющая функция.

Управляющую функцию можно разложить на две компоненты ζ и δ, где $\zeta : S \times E \to Z$ – функция действий, $\delta : S \times E \to S$ – функция переходов.

Пусть I – множество имен методов вложенного типа τ, тогда $Z \subset I$.

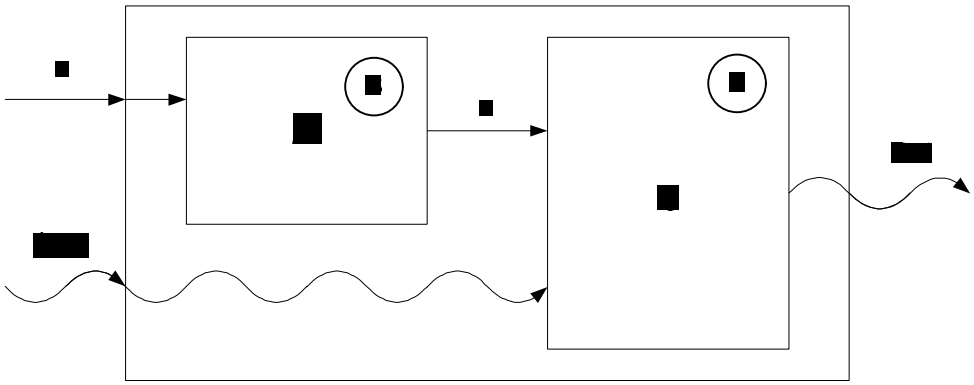


Рис. 2. Разомкнутый ААТД

Разомкнутый ААТД изображен на рис. 2. Прямыми стрелками на рисунке обозначена передача данных, множество значений которых заведомо конечно, а волнистыми

стрелками – передача элементов потенциально бесконечных множеств. Как следует из рис. 2, управляющий автомат преобразует только события, но не влияет на аргументы методов, и вообще, управляющая функция автомата оперирует исключительно конечными множествами. Это важное свойство позволяет задавать ее значения в явном виде для всех возможных аргументов (с помощью таблицы или диаграммы переходов). Именно поэтому конечные автоматы удобны для описания логики сложного поведения.

Целью автора было определить автоматизированный тип данных как частный случай обыкновенного АТД. Покажем, что разомкнутый ААТД $\langle A,\tau \rangle$ действительно является типом данных (назовем его τ'). В качестве множества значений типа τ' возьмем декартово произведение $T \times S$ множества значений вложенного типа и множества управляющих состояний автомата. Именами для методов нового типа будут служить события ААТД. Каковы будут сигнатуры этих методов? Если $\forall e \in E\ \exists Args_e : \forall s \in S\ m_z.Args = Args_e$, где $z = \zeta(s,e)$, можно определить набор типов аргументов метода m_e как $Args_e$. То же справедливо и для типа возвращаемого значения. Однако это – слишком сильное требование: в действительности, для правильной работы любого метода автоматизированного типа аргументы m_e должны только содержать достаточно информации для вызова любого из сопоставляемых ему методов m_z, и наоборот, возвращаемое значение любого из методов m_z должно быть пригодно для возврата из m_e.

Пусть построен метод m_e типа τ'. В некотором управляющем состоянии s он сопоставляется методу m_z с именем $z = \zeta(s,e)$ вложенного типа τ. Разумно выдвинуть для этих методов следующее *правило сигнатур*:

- *Контравариантность аргументов*. Количество аргументов m_e и m_z одинаково. Пусть α_i набор типов аргументов m_e, а β_i – соответствующий набор для m_z. Тогда $\forall i\ \alpha_i < \beta_i$ [1].
- *Ковариантность результата*. Оба метода m_e и m_z одновременно либо возвращают результат, либо нет. Пусть результат существует. Тогда, если результат m_e имеет тип γ, а результат m_z – тип δ, то $\delta < \gamma$.

Аналогичное правило сигнатур используется при определении понятия подтипа в работе [5]. Взяв его за основу, можно определить сигнатуру любого метода m_e типа τ' следующим образом:

$$\forall k\ m_e.Args_k = \min_{s \in S} \tau :: m_{\zeta(s,e)}.Args_k$$

$$m_e.Ret = \max_{s \in S} \tau :: m_{\zeta(s,e)}.Ret$$

Если общего подтипа соответствующих типов аргументов (или общего супертипа типов возвращаемых значений) не существует, тип τ' не может быть построен и автоматизированный тип $\langle A,\tau \rangle$ является некорректным.

Отметим, что при спецификации автоматизированного типа данных правило сигнатур можно использовать как для построения интерфейса ААТД по известному интерфейсу вложенного типа, так и наоборот. Кроме того, если интерфейсы обоих типов и управляющая функция автомата заданы, можно использовать правило сигнатур для проверки их непротиворечивости.

Исполняющие функции методов τ' строятся тривиальным образом из исполняющих функций методов τ, соответствующих различным управляющим состояниям.

---

[1] Под < подразумевается отношение «является подтипом»

Существует широкий класс сущностей со сложным поведением, для которого модель разомкнутого ААТД оказывается непригодной. Во-первых, вычислительной мощности конечного автомата может быть недостаточно для описания выделенной разработчиком управляющей компоненты автоматизированного типа. Во-вторых, даже в том случае, когда логика поведения может быть описана конечным автоматом, в нем могут присутствовать группы «однотипных» состояний. Такие состояния не несут в себе уникальной информации, их можно различить, зная вычислительное состояние объекта автоматизированного типа. Наличие групп однотипных состояний свидетельствует о неправильном распределении обязанностей между управляющей и вычислительной компонентами.

В этих случаях можно ввести в ААТД обратную связь – позволить управляющему автомату получать определенные сведения о вычислительном состоянии и руководствоваться ими при вычислении управляющей функции. Напрямую оперировать значениями вложенного типа управляющий автомат не способен, поскольку их может быть бесконечно много. В связи с этим вводится конечное множество X *наблюдаемых вычислительных состояний* (или *наблюдений*), каждому из которых соответствует некоторая совокупность значений вложенного типа. Текущее наблюдение передается в качестве аргумента управляющей функции автомата.

На практике иногда требуется, чтобы текущее наблюдение зависело не только от вычислительного состояния, но и от аргументов вызываемого метода автоматизированного типа. При этом необходимо приписать собственное множество наблюдений X_e каждому событию. В дальнейшем будем рассматривать именно этот случай, так как он является наиболее общим.

Каким образом управляющий автомат опрашивает текущее наблюдение? Будем считать, что для каждого управляющего состояния s и события e ему известен *предикат* p – имя некоторого метода-наблюдателя[2] вложенного типа, возвращающего текущее наблюдение. Отметим, что в настоящей работе управляющий автомат ААТД всюду предполагается детерминированным. Чтобы сохранить это свойство при наличии предикатов, необходимо потребовать, чтобы соответствующие им методы вложенного типа были *однозначно специфицированы*, т.е. чтобы их исполняющие функции имели вид $f : D \times \{Obj : Args\} \to \{Obj : Ret\}$ в некоторой известной области.

Автоматизированный абстрактный тип данных с обратной связью (рис. 3) назовем *замкнутым* по аналогии с замкнутой системой управления, которая также отличается от разомкнутой наличием обратной связи.

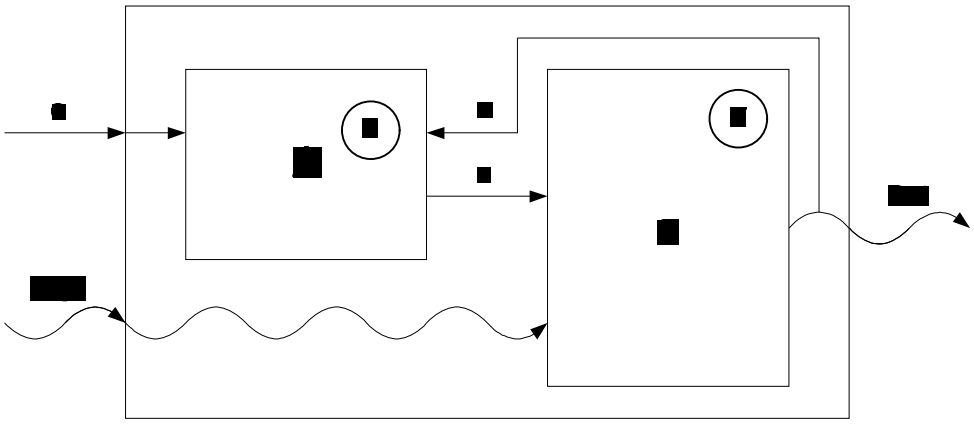


Рис. 3. Замкнутый ААТД

[2] *Наблюдателем* (*observer*) называется метод, не изменяющий текущего значения

Определение 3. *Замкнутым автоматизированным абстрактным типом данных* будем называть пару $\langle A,\tau \rangle$, где A – управляющий автомат, τ – вложенный абстрактный тип данных.

Управляющий автомат представляет собой восьмерку

$$A = \langle S, E, X, Z, P, s_0, \pi, \Delta \rangle,$$

где S – конечное множество управляющих состояний, E – конечное множество входных символов (или событий), $X = \bigcup_{e \in E} X_e$ – конечное множество наблюдаемых вычислительных состояний (или наблюдений), Z – конечное множество действий, $P = \bigcup_{e \in E} P_e$ – конечное множество предикатов, $s_0 \in S$ – начальное (стартовое) состояние, $\pi : S \times E \to P_e$ – функция предикатов, $\Delta : S \times E \times X_e \to S \times Z$ – управляющая функция.

Пусть I – множество имен методов вложенного типа τ, тогда $Z \subset I$, $P \subset I$, причем $\forall e \in E \; X_e = \{Obj : \max_{s \in S} m_{\pi(s,e)}.Ret\}$.

Для типов аргументов методов-предикатов действует то же правило контравариантности, что и для методов-действий.

В заключение рассмотрения модели автоматизированного абстрактного типа данных отметим особенности ее практического применения для спецификации сущностей со сложным поведением. Во-первых, на практике действиям и предикатам автоматизированного типа не всегда бывает удобно сопоставлять ровно по одному методу вложенного типа (обычно их может быть любое количество). Во-вторых, правила сигнатур для методов-действий и методов-предикатов могут оказаться слишком сильными. Например, они запрещают методу-действию иметь меньше аргументов, чем имеет соответствующий метод автоматизированного типа, или, наоборот, иметь «лишний» аргумент, который в зависимости от состояния и события связывается с определенной константой. Обе эти проблемы решаются введением *фиктивных* методов-действий и методов-предикатов, которые удовлетворяют всем требованиям модели ААТД, но не обозначаются в спецификации и реализации в явном виде. Фиктивные методы строятся из действительных методов вложенного типа путем их композиции, последовательного вызова, связывания аргументов и других простых операций. Если фиктивный метод получается достаточно сложным, в целях улучшения ясности спецификации его следует оформить как отдельный действительный метод.

Спецификация ААТД

В этом разделе предлагается метод формальной спецификации автоматизированных абстрактных типов данных. Отметим, что при этом новой нотации не вводится – для полного описания ААТД достаточно объединить некоторый уже существующий язык спецификации конечных автоматов с выходами, с помощью которого будет описываться управляющая компонента, и некоторый язык спецификации абстрактных типов данных, с помощью которого будет описываться вложенный тип.

Для спецификации конечных автоматов в настоящей работе для простоты используются обыкновенные *диаграммы переходов* (*state-transition diagrams*, STD). Спецификации абстрактных типов данных выполняются на языке Larch [6]. Larch состоит из двух уровней, первый из которых используется для описания множеств значений типов, а второй – непосредственно для спецификации АТД. Синтаксис второго уровня Larch зависит от целевого языка программирования, для которого предназначена спецификация. В настоящей работе будем использовать упрощенный вариант Larch/C++ [7].

Спецификация абстрактного типа данных включает следующую информацию:

- имя типа;
- описание множества значений типа;
- для каждого из методов типа:
  - имя метода;
  - сигнатуру метода;
  - семантику метода в терминах *предусловий* и *постусловий*.

Для иллюстрации предлагаемого метода спецификации ААТД рассмотрим классический пример сущности со сложным поведением – эмулятор турникета (Coin Machine) [8, 9]. Турникет принимает монеты любого достоинства. Если суммарной стоимости опущенных пользователем турникета монет достаточно для прохода, дверь открывается, и он может пройти. Если он опустил больше, чем нужно, турникет поблагодарит его, выведя соответствующее сообщение на экран. Если пользователь попытается пройти, не заплатив, турникет подаст сигнал тревоги. Кроме того, турникет может быть сломан, о чем также выводится сообщение на экран. Управляющая компонента турникета может быть описана с помощью диаграммы переходов, изображенной на рис. 4.

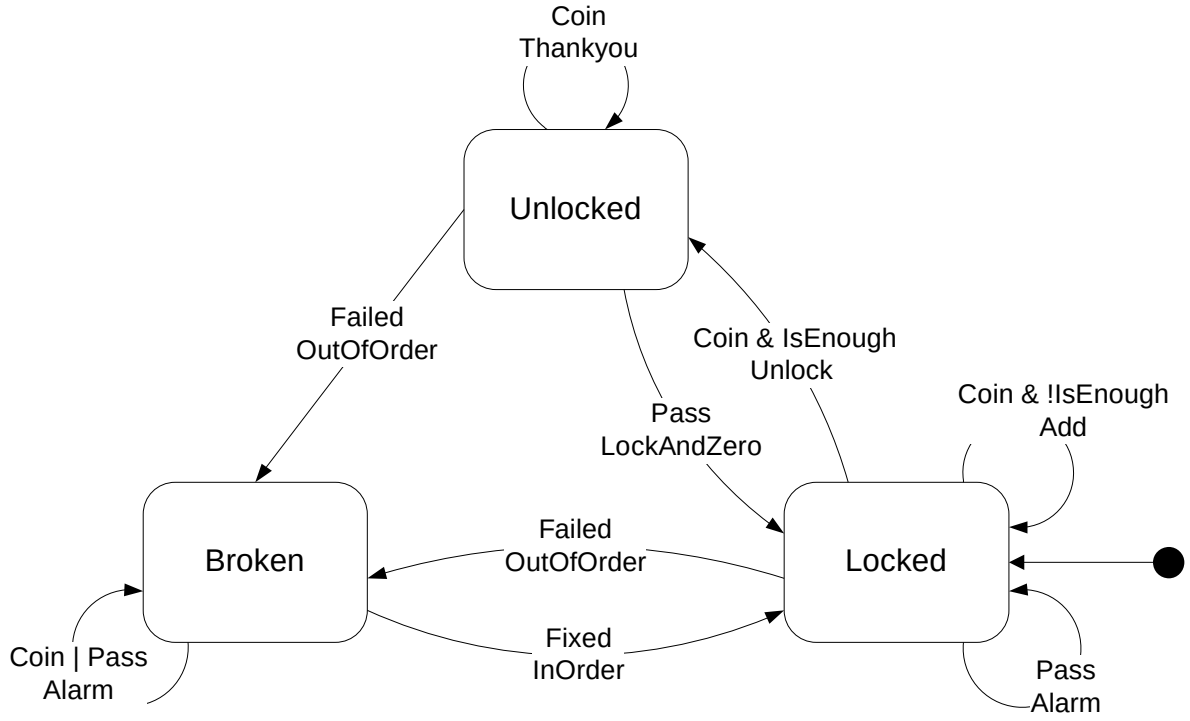


Рис. 4. Диаграмма переходов управляющего автомата Coin Machine

Из диаграммы переходов следует, что управляющий автомат турникета состоит из следующих компонент:

$S = \{\texttt{Locked}, \texttt{Unlocked}, \texttt{Broken}\}$,

$E = \{\texttt{Coin}, \texttt{Pass}, \texttt{Failed}, \texttt{Fixed}\}$,

$X = X_{\texttt{Coin}} = \{0,1\}$,

$Z = \{\texttt{Unlock}, \texttt{LockAndZero}, \texttt{Thankyou}, \texttt{OutOfOrder}, \texttt{InOrder}, \texttt{Alarm}, \texttt{Add}\}$,

$P = P_{\texttt{Coin}} = \texttt{IsEnough}$,

$s_0 = \texttt{Locked}$.

Множество наблюдений $X_{\texttt{Coin}}$ – это множество значений, возвращаемых предикатом `IsEnough`. В том, что этот предикат действительно возвращает логические значения (и, следовательно, их можно использовать в булевом выражении, помечающем дугу диаграммы переходов), можно убедиться из приведенной ниже спецификации вложенного типа турникета `Coin Machine Nested`.

```
class CoinMachineNested {
    spec bool doorOpen;     // Открыта ли дверь
    spec string msg;        // Сообщение на экране
    spec int sum;           // Текущая накопленная сумма
    spec const int cost;    // Стоимость прохода
public:
    void Unlock() {
        requires: true;
        modifies: doorOpen;
        ensures: doorOpen' == true;
    }
    void LockAndZero() {
        requires: true;
        modifies: doorOpen, sum;
        ensures: (doorOpen' == false) && (sum' == 0);
    }
    void Thankyou() {
        requires: true;
        modifies: msg;
        ensures: msg' == "Thank You";
    }
    void OutOfOrder(string failure) {
        requires: true;
        modifies: msg;
        ensures: (msg'.find("Out of Order") != string::npos) &&
            (msg'.find(failure) != string::npos);
    }
    void InOrder() {
        requires: true;
        modifies: msg;
        ensures: msg' == "In Order";
    }
    void Alarm() {
        requires: true;
    }
    void Add(int value) {
        requires: value > 0;
        modifies: sum;
        ensures: sum' == sum^ + value;
    }
    bool IsEnough(int value) {
        requires: value > 0;
        ensures: result == (sum + value >= cost);
    }
}
```

В приведенном примере множество значений абстрактного типа описывается при помощи набора *переменных спецификации* [7]. В описании семантики методов ключевое слово `requires` обозначает предусловие. Постусловие представляет собой конъюнкцию выражения `ensures` и утверждения, что значения никаких объектов, кроме указанных в выражении `modifies`, не изменяются. Через $x\texttt{^}$ обозначается значение объекта x перед началом исполнения метода, а через $x\texttt{'}$ – в момент его завершения.

Связь между диаграммой переходов управляющего автомата и спецификацией вложенного типа обеспечивается идентичностью имен методов вложенного типа и со-

ответствующих им действий и предикатов на диаграмме. Если управляющий автомат имеет большое количество состояний и переходов, на диаграмме удобнее использовать сокращенные имена событий, действий и предикатов (например, e_i, z_i и x_i соответственно, как в нотации SWITCH). В этом случае для описания ААТД дополнительно понадобится *словарь сокращений*, отображающий связь между полными и сокращенными именами.

Отметим, что при рассматриваемом подходе к спецификации автоматизированных типов данных можно легко заменить обыкновенные диаграммы переходов диаграммами Statechart или графами переходов из SWITCH-технологии. С точки зрения последней, спецификация вложенного типа вместе со словарем сокращений заменяет и дополняет часть схемы связей автомата, отвечающую за связи с объектом управления. При этом вместо неформального описания предикатов и действий на схеме связей приводится их формальная спецификация в терминах пред- и постусловий. С точки зрения подхода Statemate спецификация вложенного типа представляет собой функциональный вид системы, т.е. заменяет Activity Chart.

В разд. 0 было показано, что автоматизированный абстрактный тип данных может быть преобразован в обыкновенный АТД. Описанный выше алгоритм преобразования можно переформулировать в терминах спецификаций. Таким образом, от спецификации автоматизированного типа, использующей нотации Larch и STD, можно перейти к спецификации на Larch. Результирующая спецификация будет описывать тип данных, поведения которого с точки зрения пользователя не отличается от поведения исходного автоматизированного типа.

Итак, пусть имеется автоматизированный абстрактный тип данных $\langle A,\tau \rangle$ (в общем случае замкнутый) с именем `Name`. Пусть диаграмма переходов задает вид его управляющего автомата $A = \langle S, E, X, Z, P, s_0, \pi, \Delta \rangle$, а спецификация вложенного типа τ задает его имя, множество значений, а также имена, сигнатуры и семантику его методов в терминах предусловий и постусловий. Предусловие метода с именем *i* будем обозначать $m_i.pre$, постусловие – $m_i.post$. В случае, если метод однозначно специфицирован и возвращает значение (напомним, что это справедливо для всех методов-предикатов), будем считать, что его постусловие содержит утверждение вида $result = ret\_value$. Таким образом, подстановка $m_i.post[v / ret\_value]$ будет означать, что возвращаемое методом значение равно v.

Построим спецификацию абстрактного типа данных τ' следующим образом.

- В качестве имени τ' выберем `Name`.
- Множество значений τ' опишем как декартово произведение множества значений τ и множества S. В частности, если множество значений описывается в терминах переменных спецификации, введем перечислимый тип σ с множеством значений S. Тогда набор переменных спецификации типа τ' будет содержать переменные из τ и переменную $state : \sigma$.
- Для каждого события $e \in E$ добавим к спецификации τ' метод, такой что:
  - имя метода совпадает с e;
  - сигнатура метода определяется по формулам
    $\forall k \; m_e.Args_k = \min\limits_{s \in S} \left\{ \tau :: m_{\pi(s,e)}.Args_k, \min\limits_{x \in X_e} \tau :: m_{\zeta(s,e,x)}.Args_k \right\}$,
    $m_e.Ret = \max\limits_{s \in S, x \in X_e} \tau :: m_{\zeta(s,e,x)}.Ret$;
  - семантика метода определяется по формулам:

$$m_e.pre = \bigvee_{s\in S}\left(\begin{array}{l}(state^{\wedge} = s)\wedge \tau :: m_{\pi(s,e)}.pre \wedge \\ \wedge \bigvee_{x\in X_e}\left((\tau :: m_{\pi(s,e)}.post[x/ret\_value]) \wedge (\tau :: m_{\zeta(s,e,x)}.pre)\right)\end{array}\right),$$

$$m_e.post = \bigwedge_{s\in S}\left(\begin{array}{l}(state^{\wedge} = s) \Rightarrow \\ \Rightarrow \bigwedge_{x\in X_e}\left(\tau :: m_{\pi(s,e)}.post[x/ret\_value] \Rightarrow \left(\tau :: m_{\zeta(s,e,x)}.post \wedge (state' = \delta(s,e,x))\right)\right)\end{array}\right).$$

Отметим, что в спецификациях типов в настоящей работе не рассматриваются *формирователи* (*creators*) – функции, создающие объекты данного абстрактного типа. Формирователи часто специфицируются отдельно от самого АТД, как, например, в работе [5]. При преобразовании автоматизированного абстрактного типа данных $\langle A,\tau\rangle$ в обыкновенный тип τ' спецификации формирователей c' типа τ' можно получить из спецификаций формирователей c типа τ, изменив постусловия следующим образом:

$c'.post = c.post \wedge (result'.state = s_0)$.

После преобразования автоматизированного типа данных в обыкновенный АТД теряется информация об управляющих состояниях или, другими словами, разделение описания логики сложного поведения и описания его семантики. Однако, если спецификация ААТД приведена к такому виду, для нее могут быть доказаны различные свойства и отношения, характерные для обыкновенных типов данных и определенные в терминах их спецификации. Если при рассмотрении этих свойств учесть потерянную информацию о состояниях, свойства абстрактных типов можно переформулировать в терминах ААТД и в дальнейшем использовать непосредственно.

Отношение наследования

Интуитивное требование к абстрактному типу данных, являющемуся наследником (подтипом) некоторого другого типа, выражается в *Принципе подстановки Лисков* (*Liskov Substitution Principle*, LSP). Он заключается в том, что объект подтипа может быть использован в программе везде, где ожидается объект супертипа. Этот принцип является общепризнанным, однако его смысл зависит от модели вычислений. В работе [5] приведены два формальных конструктивных определения отношения наследования в терминах спецификаций АТД на языке Larch, каждое из которых в некотором смысле удовлетворяет LSP. В настоящей работе ограничимся рассмотрением одного из них (использующего *функцию расширения*), поскольку оно требует меньше модификаций исходного языка спецификаций абстрактных типов данных.

Одним из существенных для пользователя атрибутов абстрактного типа τ является его *инвариант* I_τ – свойство значения x_ρ, которое выполняется для любого объекта $x:\tau$ в любом вычислительном состоянии программы ρ. В [5] показано, что в отсутствие формирователей невозможно использовать индукцию для доказательства наличия у типа тех или иных инвариантов, поэтому предлагается обозначать инвариант в спецификации АТД в явном виде.

С учетом всего вышесказанного, определение отношения наследования через функцию расширения выглядит следующим образом.

Определение 4. Тип $\omega = \langle W, M\rangle$ является *подтипом* (*наследуется* от) $\tau = \langle T, N\rangle$ (пишут $\omega < \tau$), если существует функция абстракции $A : W \to T$, функция переименования $R : M \to N$ и функция расширения $E : M \times Obj^* \to Prog$, такие, что:

1. выполняется *правило инварианта*: $\forall w \in W \quad I_{\omega(w)} \Rightarrow I_{\tau(A(w))}$;

2. для каждого метода $\omega :: m \in dom(R)$ и соответствующего ему метода $\tau :: n = R(m)$ выполняется:
   - *правило сигнатур* (разд. 0);
   - *правило семантики*. Для всех объектов $x : \omega$:
   $$n.pre[A(x\hat{}) / x\hat{}] \Leftrightarrow m.pre,$$
   $$m.post \Rightarrow n.post[A(x\hat{}) / x\hat{}, A(x') / x'];$$
3. для каждого метода $\omega :: m \notin dom(R)$, значения его аргументов $a : m.Args$ и объекта $x : \omega$ должна быть определена программа $prog = E(x.m(a))$, обладающая следующими свойствами:
   - $prog$ принимает на вход набор объектов a и объект x;
   - $prog$ вызывает методы либо из $dom(R)$, либо методы типов, отличных от ω;
   - пусть метод m и программа $prog$ начинают выполняться в вычислительном состоянии программы ρ_1, причем $m.pre$ выполняется в ρ_1, m завершает исполнение в ρ_2, а $prog$ – в ψ; тогда $A(x_{\rho_2}) = A(x_{\psi})$.

Рассмотрим теперь применение этого определения к типам τ' и ω', построенным из автоматизированных типов данных $\langle A^1, \tau \rangle$ и $\langle A^2, \omega \rangle$, соответственно (для простоты рассмотрим случай, когда оба ААТД разомкнуты). Управляющие автоматы автоматизированных типов имеют вид $A^i = \langle S^i, E^i, Z^i, s_0{}^i, \Delta^i \rangle$, $i = 1,2$. Их вложенные типы имеют вид $\tau = \langle T, N \rangle$, $\omega = \langle W, M \rangle$.

Чтобы тип ω' являлся подтипом τ', во-первых, необходима функция абстракции A', соблюдающая инвариант. Из соображений разделения управляющей и вычислительной компонент ААТД, будем считать, что в данном случае эта функция может быть разложена на две составляющие – функцию абстракции вложенных типов $A : W \to T$ и функцию абстракции состояний $S : S^2 \to S^1$. Разумно считать, что инвариант автоматизированного типа данных совпадает с инвариантом его вложенного типа, поэтому правило инварианта можно сформулировать, используя только функцию A:

$\forall w \in W \quad I_{\omega(w)} \Rightarrow I_{\tau(A(w))}$.

Во-вторых, требуется функция переименования $R : E^2 \to E^1$. Она может быть частичной, т.е. автоматизированный подтип может обрабатывать больше событий, чем его автоматизированный супертип. В-третьих, для всех $E^2 \notin dom(R)$ требуется определить значение функции расширения E.

Рассмотрим теперь взаимосвязь между методом m с именем e типа ω' и соответствующим ему методом n с именем $R(e)$ родительского типа τ'. В соответствии с определением отношения наследования действует правило контравариантности аргументов вида

$\forall k \; n.Args_k < m.Args_k$,

или, в терминах аргументов методов вложенных типов,

$\forall k \; \min\limits_{s \in S^1} n_{\zeta^1(s,R(e))}.Args_k < \min\limits_{s \in S^2} m_{\zeta^2(s,e)}.Args_k$.

Таким же образом можно записать правило ковариантности результата:

$m.Ret < n.Ret$,

$\max\limits_{s \in S^2} m_{\zeta^2(s,e)}.Ret < \max\limits_{s \in S^1} n_{\zeta^1(s,R(e))}.Ret$.

Кроме того, для рассматриваемой пары методов m и n действует правило семантики. В соответствии с ним для любого объекта $x:\omega'$:

$n.pre[A'(x\wedge)/x\wedge] \Leftrightarrow m.pre$.

Обозначая через *state* управляющее состояние объекта x, а через $y:\omega$ вычислительную компоненту x, это выражение можно записать так:

$$\bigvee_{s\in S^1}\left((S(state\wedge)=s)\wedge n_{\zeta^1(s,R(e))}.pre[A(y\wedge)/y\wedge]\right)\Leftrightarrow\bigvee_{s\in S^2}\left((state\wedge=s)\wedge m_{\zeta^2(s,e)}.pre\right).$$

Поскольку управляющее состояние может одновременно иметь только одно значение, из приведенного выше выражения следует:

$$\forall s\in S^2\ m_{\zeta^2(s,e)}.pre \Leftrightarrow n_{\zeta^1(S(s),R(e))}.pre[A(y\wedge)/y\wedge].$$

Аналогичные рассуждения можно привести относительно постусловий:

$$m.post \Rightarrow n.post[A'(x\wedge)/x\wedge, A'(x')/x'],$$

$$\bigwedge_{s\in S^2}\left((state\wedge=s)\Rightarrow\left(m_{\zeta^2(s,e)}.post\wedge(state'=\delta^2(s,e))\right)\right)\Rightarrow$$

$$\Rightarrow\bigwedge_{s\in S^1}\left((S(state\wedge)=s)\Rightarrow\left(n_{\zeta^1(s,R(e))}.post[A(y\wedge)/y\wedge, A(y')/y']\wedge(S(state')=\delta^1(s,R(e)))\right)\right),$$

$$\forall s\in S^2\ m_{\zeta^2(s,e)}.post \Rightarrow n_{\zeta^1(S(s),R(e))}.post[A(y\wedge)/y\wedge, A(y')/y'],$$

и, кроме того, при абстракции состояний сохраняется функция переходов:

$\forall s\in S^2\ S(\delta^2(s,e))=\delta^1(S(s),R(e))$.

Наконец, рассмотрим функцию расширения. По определению, каждому событию $\varepsilon\in E^2\setminus dom(R)$, объекту $x:\omega'$ и набору значений аргументов $a:m_\varepsilon.Args$ сопоставляется программа *prog*. В этой программе разрешается вызывать только «посторонние» методы или методы с именами $e\in dom(R)$. Поскольку каждый метод m_e на самом деле вызывает один из методов вложенного типа $m_{\zeta^2(\cdot,e)}$ и вычисляет новое управляющее состояние, требование существования программы *prog* можно заменить следующим: $\forall s\in S^2\ \exists prog_s\in Prog$, такая что:

- $prog_s$ вызывает только «посторонние» методы или методы вида $m_{\zeta^2(\cdot,e)}$, где $e\in dom(R)$;
- пусть метод $m_{\zeta^2(s,\varepsilon)}$ и программа $prog_s$ начинают выполняться в вычислительном состоянии программы ρ_1, причем $m_{\zeta^2(s,\varepsilon)}.pre$ выполняется в ρ_1, $m_{\zeta^2(s,\varepsilon)}$ завершает исполнение в ρ_2, а $prog_s$ – в ψ; тогда $A(y_{\rho_2})=A(y_\psi)$.

Исходя из приведенных рассуждений, можно сформулировать простое условие на автоматизированные типы $\langle A^1,\tau\rangle$ и $\langle A^2,\omega\rangle$, достаточное для того, чтобы соответствующие им обыкновенные типы τ' и ω' были связаны отношением наследования в смысле определения Б. Лисков через функцию расширения. Это условие будем рассматривать как определение отношения наследования для разомкнутых автоматизированных абстрактных типов данных.

Определение 5. Разомкнутый ААТД $\langle A^2,\omega\rangle$ ($\omega=\langle W,M\rangle$) является *подтипом* разомкнутого ААТД $\langle A^1,\tau\rangle$ ($\tau=\langle T,N\rangle, A^i=\langle S^i,E^i,Z^i,s_0^{\,i},\Delta^i\rangle$), если существует функция абстракции состояний $S:S^2\to S^1$ и функция переименования событий $R:E^2\to E^1$, такие, что:

- $S(s_0^{\,2})=s_0^{\,1}$
- $\forall s\in S^2\ \forall e\in dom(R)\ \ S(\delta^2(s,e))=\delta^1(S(s),R(e))$.

Кроме того, ω является подтипом τ (в смысле определения через функцию расширения), и для их функции переименования $R_{\omega\tau}$ справедливо

$\forall s \in S^2 \; \forall e \in dom(R)\; R_{\omega\tau}(\zeta^2(s,e)) = \zeta^1(S(s), R(e))$.

Заключение

В работе подробно рассмотрено понятие автоматизированного абстрактного типа данных. По мнению автора, эта концепция наилучшим образом отражает объектно-ориентированный подход к моделированию сущностей со сложным поведением. ААТД представляет собой тип данных, логика поведения которого описывается конечным автоматом, а семантика – вложенным типом данных. Предложен метод спецификации автоматизированных типов данных, а также алгоритм преобразования описания ААТД в спецификацию эквивалентного ему обыкновенного типа. В работе подробно рассмотрено отношения наследования для типов со сложным поведением, предложено простое конструктивное определение этого отношения для разомкнутых автоматизированных абстрактных типов данных.

Попытки перенести свойства, характерные для абстрактных типов данных, на сущности со сложным поведением делались и раньше, например, в работе [10]. Однако эти свойства рассматривались в применении к конечным автоматам. С нашей же точки зрения, автомат – лишь одна из двух компонент сущности со сложным поведением, и если говорить о наследовании, композиции или обобщенности, то в применении именно к абстрактному типу, описывающему сущность целиком.

В дальнейшем автор планирует определить отношение наследования для замкнутых ААТД, а также рассмотреть применение к автоматизированных типам и их экземплярам других объектно-ориентированных техник моделирования и проектирования.

Литература

1. Хопкрофт Дж., Мотвани Р., Ульман Дж. Введение в теорию автоматов, языков и вычислений. М.: Вильямс, 2002. 528 с.
2. Буч Г., Рамбо Дж., Джекобсон И. Язык UML: Руководство пользователя. СПб: Питер, 2004. 430 с.
3. Harel D., Polity M. Modeling Reactive Systems with Statecharts. The Statemate Approach. New York: McGraw-Hill, 1998. 258 p.
4. Шалыто А.А. SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб: Наука, 1998. 628 с.
5. Liskov B., Wing J. Family Values: A Behavioral Notion of Subtyping // ACM Trans. Program. Lang. Syst., 1994.
6. Guttag J.V., Horning J.J. Larch: Languages and Tools for Formal Specification. New York: Springer, 1993. 571 p.
7. Leavens G.T. Larch/C++ An Interface Specification Language for C++ // http://www.cs.iastate.edu/~leavens/larchc++.html
8. Martin R. Three-Level FSM. Pattern Languages of Program Design, 1995.
9. Yacoub S.M., Ammar H.H. Finite State Machine Patterns. // Proceedings of EuroPLoP. 1998.
10. Sane A., Campbell R. Object-Oriented State Machines: Subclassing, Composition, Delegation, and Genericity. // Proceedings of OOPSLA. 1995.

МЕТАПРОГРАММИРОВАНИЕ НА ОСНОВЕ ТЕКСТОВОГО ПРЕПРОЦЕССОРА

В.Ю. Лоторейчик

Научный руководитель – кандидат технических наук Д.Г. Шопырин

В работе предлагается ряд мер по усовершенствованию препроцессора языка C/C++. Предлагаемые расширения позволят как повысить уровень безопасности использования препроцессора, так и значительно увеличить его выразительную мощность. В частности, предлагаемые усовершенствования призваны расширить возможности и упростить использование метапрограммирования на основе препроцессора.

Введение

Препроцессор – это программа, которая обрабатывает входные данные и формирует выходные данные, используемые как входные другой программой. Наиболее часто в программировании встречается пример, когда препроцессор создает из исходного кода программы входной поток информации для компилятора. Такой препроцессор принято называть текстовым. Основные применения текстового препроцессора в программировании – это написание макросов, условная компиляция и включение файлов. В данной статье будет рассмотрен только механизм макросов. Вызовы макросов в коде программы текстовый препроцессор заменяет на заранее описанный или формируемый по определенным правилам текст (код), при этом для формирования текста подстановки могут выполняться различные нетривиальные действия.

Хотя текстовый препроцессор применяется широко в различных средах разработки, но в этой статье будем касаться только препроцессора языка C++. Создатель языка C++ Б. Страуструп не рекомендует пользоваться препроцессором, потому что он создает проблемы для многих инструментов разработки, но иногда его использование вполне оправдано. Описанный на данный момент стандартом C++ препроцессор достаточно беден и не позволяет писать макросы, выполняющие сложные действия для формирования текста макроподстановки (замена макроса на некоторый программный код). В работе предлагается ряд мер по усовершенствованию препроцессора C++. Расширенный препроцессор языка C++ будем для краткости называть CPP#.

Аналогичные разработки:

- Camlp4 [1];
- Generic Preprocessor [2].

В Camlp4 присутствует механизм рекурсии. Оба препроцессора позволяют выполнять вычисления на стадии обработки кода препроцессором. Основные преимущества CPP# по сравнению с Camlp4 и Generic Preprocessor – это высокая безопасность и возможность структурировать макросы. Также с помощью CPP# метапрограммировать гораздо удобней, так как он изначально проектировался с этой целью.

Проблематика

Развитие механизма макросов препроцессора языка C++ на данный момент приостановлено в основном из-за того, что он является небезопасным. Возможно, какие-то изменения в препроцессоре появятся в C++0x, который выйдет приблизительно в 2009 году. Совпадение имен макросов с именами переменных или функций приводит к нежелательным результатам, при этом препроцессор C++ не может этого отследить. Особенно опасно, когда имя некоторой функции совпадает с именем макроса с тем же количеством аргументов. Также препроцессор C++ не может отследить совпадение имен макросов между собой. Такие случаи возникают, когда разные файлы создаются разными программистами. Искусственным решением может быть использование в именах

макросов длинных префиксов, которые обозначают принадлежность их к некоторой библиотеке или группе. В механизм макросов необходимо внести структурированность. Конечно, у макросов есть и другие недостатки, но все же на данный момент опасность их использования и отсутствие структурированности являются главными.

После того, как мы повысим безопасность и структурированность макросов, можно расширять синтаксис и семантику макроязыка в C++, чтобы получить обновленный, более гибкий инструмент для выполнения сложных действий на стадии обработки кода препроцессором. Эти сложные действия можно условно разделить на две части: вычисления на стадии препроцессинга и метапрограммирование. Метапрограммирование – это написание программ, которые генерируют другие программы или их части. Мы ставим перед собой цель так расширить синтаксис и семантику препроцессора C++, чтобы с его помощью можно было легко писать метапрограммы. Приведем примеры использования метапрограммирования на базе макросов.

1. Построение таблицы значений некоторой функции. Использование макросов позволяет осуществлять данную операцию на стадии препроцессинга.
2. Автоматическое написание схожих по структуре фрагментов кода. Использование макросов позволяет избежать ошибок, связанных с дублированием, и повышает читаемость программы.

В состав пакета Boost входит библиотека Boost.Preprocessor [3], позволяющая метапрограммировать и выполнять вычисления на стадии препроцессинга в рамках старого препроцессора C++, но она обладает большими ограничениями. Например, складывать можно лишь те числа, сумма которых не превосходит 256. Реализацию этих макросов можно посмотреть в исходных кодах библиотеки BOOST [3].

Повышение уровня безопасности

Первоочередной задачей является повышение уровня безопасности использования препроцессора. Для сохранения обратной совместимости с уже существующим кодом безопасный синтаксис действует внутри директив `#safe` и `#safe-end`. Перед именами макросов при их вызове в безопасном синтаксисе ставится служебный символ ‘#’, что позволяет обращаться к макросам явно. Директивы `#safe` и `#safe-end` должны обязательно идти в паре.

```
#define A 5
#safe
int a = #A; //нормальный вызов, #A раскроется в число 5
 int b = A; //препроцессор пропустит это

#safe-end
```

Безопасный синтаксис решает проблему совпадения имен макросов с именами переменных или функций. Текст программы, не заключенный между вышеприведенными директивами, обрабатывается согласно синтаксису стандартного препроцессора C++. Можно принудительно заставить препроцессор перейти в стандартный синтаксис с помощью директив `#unsafe` и `#unsafe-end.` Вызовы макросов внутри других макросов осуществляются без префикса.

Введение в механизм макросов пространств имен позволяет решить проблему совпадения имен макросов между собой, а также больше их структурирует. Для объявления пространства имен используется директива CPP `#namespace.` После этой директивы идет название пространства имен. Описание пространства имен заканчивается директивой `#namespace-end,` присутствие которой необходимо. Доступ к макросам, объявлен-

ным внутри пространства имен, осуществляется через префикс в виде названия этого пространства имен, после которого ставится диез, а затем само имя макроса.

```
#namespace math
#define sqr(x) (x) * (x)
#namespace-end
#safe
int a = #math#sqr(5); //присвоится 25
#safe-end
```

Внутри пространства имен доступ к макросам, в нем объявленным, идет напрямую, без префикса. Также CPP+ допускает вложенные пространства имен. Доступ к макросам вложенных пространств имен идет через присоединение соответствующего количества префиксов, разделенных символом диез (например: `#math#geom#ANGLE).` Для подключения пространства имен используется директива `#using`, после которой через запятую перечисляются названия тех пространств имен, которые будут подключены. В области текста программы, где подключено пространство имен, доступ к его макросам осуществляется без префикса. Пространства и подпространства имен образуют древовидную структуру, подобную XML или обычным пространствам имен C++.

В CPP# добавлен механизм изоляции. Механизм изоляции основан на идеях, предложенных Б. Страуструпом в работе [4]. В области кода программы, заключенного между директивами препроцессора `#scope` и `#scope-end,` не видны макросы, которые объявлены вне этого блока. После того, как программист написал директиву `#scope,` он может быть уверен в том, что коллизий с макросами у него не будет. Макросы, объявленные внутри некоторого `scope`-блока, видны только внутри этого блока. Пример:

```
#safe
#define A 10
#define B 9

#scope //отключает все объявленные снаружи макросы
int A = 7; //объявление целого числа A
#define B 7
#define C 99
int x = #B; //x становится равным 7
#scope-end

int x = #A; //x становится равным 10
int y = #B; //y становится равным 9
int z = #C; //ошибка: C не определено

#safe-end
```

Предлагаемый в настоящей работе механизм изоляции предоставляет также различные способы взаимодействия пространств имен и блоков изоляции. `Scope`-блоки могут быть вложены друг в друга – это может применяться в крупных проектах. Иногда может быть необходимо получить доступ к некоторым объявленным снаружи макросам внутри `scope`-блока. Для этого в CPP# используется директива `#import.` По-

сле неё идет перечисление через запятую тех макросов или пространств имен, доступ к которым необходимо получить внутри scope-блока.

Расширение функциональных возможностей препроцессора

В целях расширения функциональных возможностей препроцессора в настоящей работе предлагаются следующие механизмы:

- механизм рекурсии;
- механизм вычислений на стадии обработки кода препроцессором;
- механизм специализации макросов;
- механизм перегрузки макросов;
- механизм передачи в макрос переменного числа параметров.

Механизм рекурсии. Рассмотрим пример вычисления факториала на стадии препроцессинга:

```
#include <iostream>

#safe

#define FACTORIAL(num) $(num * FACTORIAL($(num – 1)))
#define FACTORIAL(#(0)) 1 /* специализация */

int main()
{
     std::cout<<#FACTORIAL(5);
}

#safe-end
```

Первое расширение касается внесения в макросы возможности рекурсии. В теле макроса может использоваться тот же самый макрос. Выполняется рекурсивная макроподстановка. Рекурсия была бы бесконечной, но за счет специализации макросов имеется возможность сделать ее конечной (механизм специализации будет описан ниже). Рассмотрим, как работает рекурсивный механизм:

```
#define A 2 ## A
```

При вызове макроса без аргументов A препроцессор сначала раскрывает макрос A, находящийся в теле A. Процесс происходит до тех пор, пока не исчерпается стек препроцессора. `A -> 2 ## A -> 2 ## 2 ## A -> 2 ## 2 ## 2 ## A -> ...` количество итераций зависит от реализации препроцессора.

Рекурсия может быть ограничена с помощью механизма специализации и вычислений на этапе препроцессинга. Макрос, который будет печатать число из заданного числа двоек, можно написать следующим образом.

```
#safe
#define A(num) 2 ## A($(num – 1))
//специализация макроса A
#define A(#(0))
...
printf(“%d”, #A(5)); //выдаст число из 5 двоек
```

```
...
#safe-end
```

Макровызов #A(10) работает следующим образом: 2 ## A($(10 – 1)) -> 2 ## A(9) -> 2 ## 2 ## A($(9 – 1)) ->...-> 2 ## … ## 2 ## A(0) -> (вместо A(0) подставляется пустая строка, так как для значения аргумента равного нуля написана специализация) в итоге получается 2 ## 2 ## … ## 2. После этого выполняется конкатенация. В итоге получается 22…2.

В том случае, если одним из аргументов первого макроса при вызове является другой макрос, сначала раскрывается первый (внешний) макрос. Может потребоваться, чтобы внешний макрос раскрывался позже внутреннего. Если при объявлении некоторый аргумент макроса заключается в фигурные скобки, то это значит, что этот аргумент будет обработан препроцессором раньше.

```
#safe
#define A({x}) x + 1
#define B 2
…
int a = #A(#B) //B раскрывается раньше, чем A
…
#safe-end
```

Рекурсия — очень мощный инструмент. Он позволяет реализовывать циклы. Как говорит Л. Питер Дойч, «Итерация – от человека, рекурсия – от Бога» [5].

Механизм вычислений на стадии обработки кода препроцессором. При реализации рекурсии возникает необходимость в целочисленных вычислениях. Необходимость в других вычислениях меньше, но также есть. Аргументом оператора $() может быть:

1. целочисленное выражение,
2. вещественное выражение,
3. булевское выражение,
4. выражение, содержащее строки.

Оператор $() выполняет вычисление выражения, которое является его аргументом, и подставляет результат. Если оператор $() сам является аргументом некоторого макроса (#A($(10-1))), то он раскрывается раньше, чем вызывается макрос. Аналогично, если у оператора $() в выражении встречается макрос, то он раскрывается прежде, чем начнется вычисление выражения.

В соответствующих выражениях допустимы лишь применимые к ним операции и операнды. Например, в целочисленных выражениях допустимы операции сложения, умножения, деления нацело, взятие остатка, унарный минус и логические операции. CPP# сам определяет, с каким типом выражения он имеет дело.

```
#define SUM(a, b) $(a + b)
#define MUL(a, b) $(a * b)
```

Макрос SUM может не только складывать два вещественных или целых числа, но и выполнять конкатенацию строк. Все зависит от аргументов.

Механизм специализации макросов. Механизм специализации необходим для реализации рекурсии. Однако этот механизм может быть полезен и сам по себе. Синтаксически механизм специализации макросов похож на механизм специализации шаблонов. Основная идея состоит в том, чтобы вызовы макросов с определенными значе-

ниями параметров раскрывались согласно особой, специально написанной для них подстановке. Если для некоторого макровызова есть две возможности по раскрытию, то выбирают наиболее подходящую возможность. Рассмотрим пример.

```
#define MACRO(x, y, z) $(x + y + z)
#define MACRO(#(<=0), #(<=0), #(<=0)) 0
```

Если вызвать макрос `MACRO` со всеми численными аргументами, не превосходящими нуля, то будет вызвана специализация, и вместо макроса будет подставлено 0.

Опишем формальные правила, по которым выбирается специализация. Для численных и строковых аргументов допустимы такие специализации аргумента, как равенство и строгое или нестрогое неравенство. В случае строковых аргументов сравнение происходит согласно лексикографическому порядку. Для булевских аргументов допустима только специализация равенство. Если некоторый набор аргументов подошел под две различные специализации макроса, возникает вопрос о том, какую из специализаций выбрать.

1. Выбирается та специализация, где большее количество аргументов специализировано.
2. Если некоторый вызов удовлетворяет двум специализациям, в которых специализировано одинаковое количество аргументов, то выбирается та, где больше специализаций-равенств.
3. Возможны также ситуации, когда нет признаков, по которым можно отдать предпочтение одной из специализаций. В этом случае препроцессор выдает сообщение об ошибке.

Следующий макрос ставит запятую, если ему в качестве аргумента передается число больше нуля, ничего не ставит, если передается ноль, и ставит точку, если передается отрицательное число. Если в качестве аргумента передается не число, то обе специализации игнорируются, и макрос всегда раскрывается в запятую.

```
#define COMMA_IF(n) ,
#define COMMA_IF(#(0))
#define COMMA_IF(#(<0)) .
```

Сравнивать аргументы можно с выражениями, содержащими другие макросы.

Иногда бывает полезно в специализации получить значение аргумента. Тогда обращение к нему можно выполнить через его номер. Нумерация аргументов начинается с нуля. Используется специальный оператор препроцессора CPP# диез. Например, #(0) является синонимом значения первого аргумента. Доступ к аргументам через их номера крайне полезен для макросов с переменным числом аргументов. Пример:

```
#define MACRO(x, y, z) $(x+y+z)
#define MACRO(#(<=0), y, #(<=0)) #(0)
```

Специализация макроса получает доступ к значению первого аргумента через его номер.

Механизм перегрузки макросов. Еще один инструмент – это перегрузка макросов. Он позволяет писать разный код для макросов с одинаковым названием, но разным количеством аргументов. Пример:

```
#define PRINT(a, b) cout<<(a)<<(b)
#define PRINT(a, b, c) cout<<(a)<<(b)<<(c)
```

Перегрузка является общепринятым механизмом в современных языках программирования, и ее отсутствие значительно обедняет выразительную силу препроцессора.

Механизм передачи в макросы переменного числа параметров. Синтаксис и семантика CPP# допускает написание макросов с переменным числом аргументов. При объявлении переменного числа аргументов вместо последнего аргумента такого макроса и исключительно только вместо последнего ставится $$. Перед $$ могут идти другие аргументы макроса. При вызове на месте $$ ставится любое число аргументов, в том числе их может и не быть. Если макрос с переменным количеством аргументов вызывает другой макрос или самого же себя рекурсивно, то вызванному макросу можно передать $$ как аргумент, и он будет совпадать с тем, что было передано исходному макросу при вызове. Если в вышеизложенном случае одним из аргументов, соответствующих $$, был другой макрос, то он раскроется перед новым вызовом. Макросы с переменным количеством аргументов могут быть полезны при работе с массивами на стадии обработки кода препроцессором. Служебное слово $$len возвращает количество аргументов, стоящих на месте $$. Доступ к аргументам осуществляется по номеру аргумента через оператор препроцессора $$(). $$first и $$last обозначают соответственно первый и последний аргументы в перечислении. A $$tail отрезает у перечисления первый элемент и возвращает оставшуюся часть. Рассмотрим пример макроса, вычисляющего сумму некоторого количества чисел.

```
    //Вспомогательные макросы
    #define SUM_(num, $$) $($$first + SUM_($(num - 1), \
$$tail))
    #define SUM_(#(=0), $$) $$(0)
    #define SUM_(#(<0), $$)
    //Макрос, вычисляющий сумму
    //Обратите внимание, что $$len возвращает количество
//аргументов у макроса SUM_ на месте $$, а не у макроса //SUM
    #define SUM({$$}) SUM_($($$len - 1), $$)
    //Макрос, генерирующий арифметическую прогрессию
    #define SEQ(n) n, SEQ($(n - 1))
    #define SEQ(#(=0)) 0
    //Применение - вычисление суммы арифметической прогрессии
    #safe
    int sum10 = #SUM(#SEQ(10));
    #safe-end
```

Отметим, что раскрытие SEQ(10) происходит перед вызовом SUM_ и получившаяся последовательность подставляется на место $$, поэтому $$len возвращает 10, а не 1.

С помощью макросов с переменным количеством аргументов можно решать разнообразные нетривиальные задачи: искать минимум в произвольном количестве аргументов, сортировать массивы и т.д.

Примеры использования

Перейдем к рассмотрению более сложных примеров применения обновленного препроцессора. Метапрограммирование на основе текстового препроцессора можно использовать не только для заполнения массивов значений функций и других подобных операций. Даже при использовании всей мощи шаблонного метапрограммирования и библиотек, позволяющих метапрограммировать, в некоторых случаях при программи-

ровании на C++ требуется большое количество повторений похожего по форме кода. Рассмотрим структуру данных из работы [7]:

```
template <class T0, class T1, class T2>
struct tiny_size
: mpl::int_<3> {};
```

Помимо повторяющегося фрагмента в списке параметров вышеуказанного шаблона, для него в работе [7] написаны три специализации, которые имеют много общего:

```
template <class T0, class T1>
struct tiny_size<T0, T1, none>
: mpl::int_<2> {};

template <class T0>
struct tiny_size<T0, none, none>
 : mpl::int_<1> {};

template <>
struct tiny_size<none, none, none>
 : mpl::int_<0> {};
```

Предпочтительно такой «механический» код не писать руками, а генерировать автоматически. Это позволит избежать ошибок и сделает код более гибким. Реализуем шаблон из предыдущего раздела для 4 параметров-типов. Сначала запишем само объявление:

```
#define TINY_MAX_SIZE 4

#define ENUM_PARAMS(num, T) ENUM_PARAMS(#(num – 1), T) \
,T ## $(num – 1)
#define ENUM_PARAMS(#(0), T)

template <ENUM_PARAMS(TINY_MAX_SIZE, class T)>
struct tiny_size
 : mpl::_<TINY_MAX_SIZE> {};
```

Макрос `ENUM_PARAMS` используется для генерации перечисления параметров. Например, `ENUM_PARAMS(5, class T)` раскроется в `class T0, class T1, ..., class T4.`

Теперь перейдем к написанию макропрограммы, которая сгенерирует все специализации выше описанного шаблона. Для этого необходимо написать вспомогательные макросы.

Первый макрос `ENUM_PARAMS` уже написан. Нам понадобится макрос пунктуации:

```
#define COMMA_IF(n) ,
#define COMMA_IF(#(0))
```

Этот макрос ставит запятую, если его аргумент не является целым числом, равным нулю. Кроме того, необходим макрос повторения:

```
#define REPEAT(num, ACTION) ACTION(num - 1) \
REPEAT($(num – 1), ACTION)
```

```
#define REPEAT(#(0), ACTION)
```

Этот макрос повторяет другой макрос `ACTION num` раз, причем в качестве аргумента для `ACTION` он подставляет номер текущей итерации. Теперь перейдем непосредственно к написанию макропрограммы.

```
#define TINY_print(num) none,
#define TINY_print(#(1)) none
#define TINY_print(#(0))

#define TINY_size(n)                    \
 template <ENUM_PARAMS(n, class T)>           \
  struct tiny_size<                  \
      ENUM_PARAMS(n, T)                 \
      COMMA_IF(n)                       \
     REPEAT($(TINY_MAX_SIZE – n), TINY_print);   \
 >                                  \
      : mpl::int_<n> {};

#safe
#REPEAT(#TINY_MAX_SIZE, #TINY_size)
#safe-end
```

Заключение

В работе предложен ряд мер по усовершенствованию препроцессора языка C/C++. Для повышения безопасности использования препроцессора предложен явный синтаксис макроподстановки, пространства имен для макросов и блоки изоляции. Для повышения выразительной силы препроцессора предложены следующие усовершенствования:

- механизм рекурсии;
- механизм вычислений на стадии обработки кода препроцессором;
- механизм специализации макросов;
- механизм перегрузки макросов;
- механизм передачи в макрос переменного числа параметров.

Целями дальнейшей работы являются реализация препроцессора CPP#, а также разработка библиотеки макросов, аналогичной библиотеке Boost.Preprocessor.

Литература

1. http://caml.inria.fr/pub/docs/manual-camlp4/index.html
2. http://www.nothingisreal.com/gpp/gpp.html/
3. http://boost-consulting.com/tmpbook/preprocessor.html
4. http://www.open-std.org/
5. Страуструп Б. Язык программирования C++ . М.: БИНОМ, 2005. С. 203–205.
6. Ахо А., Сети Р., Ульман Дж. Компиляторы: принципы, технологии, инструменты. М., СПб, Киев: Вильямс, 2003. С. 22–37.
7. Abrahams D., Gurtovoy A. C++ Template Metaprogramming: Concepts, Tools, and Techniques from Boost and Beyond. Addison-Wesley Professional, 2004. Appendix A.
8. http://www.solarix.ru/

МЕТОДЫ КОНТРОЛЯ КАЧЕСТВА ОБСЛУЖИВАНИЯ В МУЛЬТИСЕРВИСНЫХ СЕТЯХ

М.Ю. Будько

Научный руководитель – кандидат технических наук, доцент Г.П. Жигулин

Рассматриваются методы управления качеством обслуживания в компьютерных сетях. Определяются области их применения, достоинства и недостатки. Предлагается метод, позволяющий обеспечить требуемое качество обслуживания в сетях, не поддерживающих специальных механизмов резервирования ресурсов.

Введение

Контроль качества является актуальной задачей в вычислительных сетях. Современные информационные технологии все больше ориентируются на передачу потоков мультимедийных данных реального времени. Это заставляет владельцев компьютерных сетей внедрять технологии, обеспечивающие качество обслуживания. Следовательно, производители сетевых устройств стремятся закладывать в них соответствующие возможности. Сложность применяемых технических решений влияет на стоимость оборудования и услуг, которые на нем предоставляются. Под мультисервисными сетями в настоящей работе понимаются сети, одновременно использующиеся для передачи данных, не предъявляющих требований к полосе пропускания, и трафика, критичного к задержкам.

При передаче на большое расстояние важной является поддержка сквозного качества обслуживания на протяжении всего маршрута. Задача усложняется, когда потоковые данные проходят через сети, принадлежащие разным провайдерам. Операторы могут применять несовместимые механизмы качества обслуживания или вообще их не поддерживать. Таким образом, возникает необходимость анализа достоинств, недостатков, области применения методов контроля качества обслуживания.

Обзор методов контроля качества обслуживания

Методы контроля качества обслуживания предназначены для управления такими сетевыми параметрами, как:

1) задержка (delay), т.е. время, необходимое пакетам для прохождения от источника к получателю;
2) джиттер (jitter) – время, на которое изменяется значение задержки;
3) пропускная способность (throughput) – скорость передачи пакетов;
4) уровень потери пакетов (packet loss rate).

Организация IETF (Internet Engineering Task Force), отвечающая за спецификацию протоколов и архитектуры сети Internet, выделяет две группы услуг, в зависимости от требуемого качества обслуживания:

1) дифференцированные услуги (differentiated services);
2) интегрированные услуги (integrated services).

В первом случае потоки информации обслуживаются в соответствии с тем приоритетом, который они имеют. При этом не гарантируется выполнение каких-либо требований к значениям сетевых параметров. Во втором случае предлагается механизм, позволяющий резервировать ресурсы на всем протяжении маршрута. Благодаря этому обеспечивается необходимое качество обслуживания.

Ограничениями дифференцированного подхода является то, что, несмотря на высокий приоритет, данные все равно могут быть подвержены непредсказуемым задержкам при перегрузках в сети. Преимущества состоят в том, что не требуется сложного и дорогого оборудования для построения сети. При интегрированном подходе обеспечи-

вается предварительное планирование и резервирование ресурсов. Это влияет на стоимость оборудования, однако гарантирует качество услуг.

В IP-сетях, изначально работающих по принципу наибольших усилий, возможно предоставление качественных услуг для передачи аудио, видео и другого чувствительного трафика. Но не вся сетевая инфраструктура поддерживает соответствующие механизмы. Следовательно, актуальной является задача разработки способов, позволяющих с минимальными затратами и на существующей технической базе предоставлять сравнимые по качеству услуги. В [2] предлагается, а в [1] и [3] подробно описывается метод статистического анализа состояний вызовов. Его использование в сетях IP-телефонии позволяет реализовать подход, при котором качество услуги повышается не за счет изменения инфраструктуры, а за счет улучшенного управления. Следовательно, можно пробовать распространить этот подход и на другие виды услуг.

Метод управления качеством от источника

На рис. 1 представлена наиболее распространенная схема работы пользователя в сети Интернет.

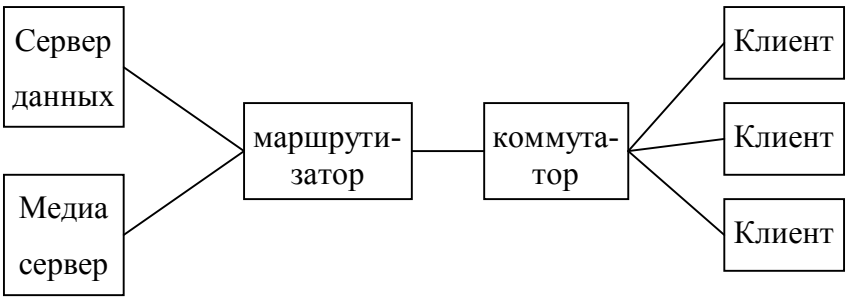


Рис. 1. Типичная схема сети

Под сервером данных понимается любой источник информации, не предъявляющий особых требований к задержкам и другим сетевым параметрам, а под медиа-сервером – источник потоков информации, чувствительной к состоянию сети. Будем считать, что ни один из участников обмена трафиком не поддерживает возможности резервирования ресурсов, но поддерживает дифференцированное обслуживание по приоритетам. В контексте настоящей работы основное внимание будет уделяться мультимедийным потокам информации, поэтому схема передачи обычных данных анализироваться не будет.

Участки, на которых возможны перегрузки, представлены на рис. 2 пунктирной линией.

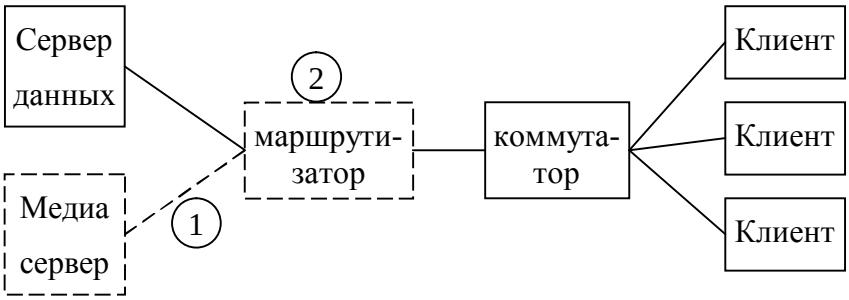


Рис. 2. Узкие места в сети при передаче мультимедиа потоков

Рассмотрим причину возникновения перегрузок на участке (1). Предположим, что медиа-сервер передает потоки мультимедиа, каждый из которых занимает определен-

ную полосу пропускания. Тогда график загрузки линии (1) будет выглядеть как на рис. 3.

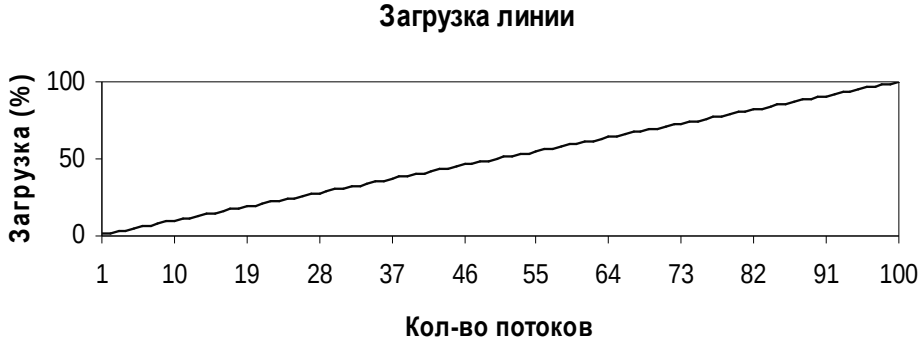


Рис. 3. Загрузка линии (1)

При этом делается предположение, что один поток передается с интенсивностью 1 Мбит/с, а общая пропускная способность канала связи – 100 Мбит/с. В результате на одной подобной линии связи возможна передача приблизительно 100 потоков. При необходимости сгенерировать большее количество потоков возникнут ошибки на уровне приложений.

Увеличение количества потоков возможно только за счет использования сжатия или увеличения степени сжатия, если оно уже применялось. Наиболее приемлемой является ситуация, когда приложение-сервер способно самостоятельно оценивать загрузку линии связи и динамически изменять степень сжатия. В этом случае график загрузки линии будет похож на рис. 4.

Рис. 4. Увеличение числа потоков за счет применения сжатия

При достижении некоторого уровня загрузки канала скорости потоков уменьшаются. Это позволяет повысить количество передаваемых потоков.

Основные проблемы, возникающие при таком подходе:

1) правильная оценка ситуации в сети;
2) наличие механизмов, позволяющих динамически увеличивать или уменьшать степень сжатия данных.

На участке (2) перегрузки будут возникать, если объемы информации, поступающие от серверов, будут превышать возможности линии связи, идущей к пользователям. Однако при наличии механизмов оценки ситуации в сети и использовании дифференцированного обслуживания возможно снижение загрузки за счет повышения степени сжатия. Для этого необходимо использовать дополнительные программные средства мониторинга состояния сети и качества предоставляемой услуги, как показано на рис. 5.

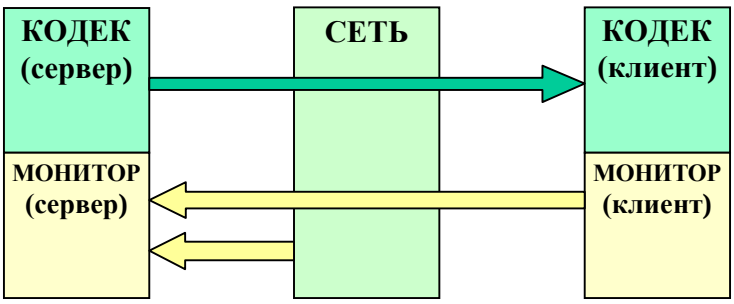


Рис. 5. Схема взаимодействия

Их задача состоит в определении не только качества работы сети путем измерения параметров ее функционирования, но и влияния этих параметров на потребительские свойства предоставляемой мультимедиа-услуги. Это возможно с помощью измерения таких параметров, как:

1) количество поврежденных (восстановленных) пакетов на стороне клиента,
2) количество пропущенных кадров при воспроизведении,
3) реальная частота кадров.

Заключение

Можно сделать вывод, что сети, не приспособленные для передачи мультимедиа-потоков, также возможно использовать для предоставления сервиса с достаточно высоким качеством. Однако это потребует внедрения специальных способов управления сетью.

Литература

1. Будько М.Ю. Выбор показателей, используемых для оценки качества работы сети IP-телефонии // Тез. докл. XII Всерос. научно-метод. конф. «Телематика'2005». СПб: СПбГУ ИТМО, 2005.
2. Будько М.Ю. Оценка качества передачи голоса в сетях IP-телефонии // Труды II конференции молодых ученых СПбГУ ИТМО. СПб: СПбГУ ИТМО, 2005.
3. Будько М.Ю. Оценка качества работы многоуровневой VoIP-сети // Научно-технический вестник СПбГУ ИТМО. Вып. 20. СПб.: СПбГУ ИТМО, 2005.
4. Столингс В. Современные компьютерные сети. 2-е изд. СПб: Питер, 2003.
5. Шринивас В. Качество обслуживания в сетях IP. М.: Вильямс, 2003.

АНАЛИТИЧЕСКАЯ МОДЕЛЬ ПОДСИСТЕМЫ ШИФРОВАНИЯ ТРАНСПОРТНОГО ПОТОКА

М.В. Костин

Научный руководитель – доктор технических наук, профессор Т.И. Алиев

Статья содержит детальное описание аналитической модели подсистемы шифрования транспортного потока, передаваемого в широковещательном однонаправленном канале связи. Аналитическая модель позволяет рассчитывать основные характеристики подсистемы при определенных исходных параметрах и заданном внешнем воздействии.

Введение

Подсистема шифрования транспортного потока, передаваемого в широковещательном однонаправленном канале связи (спутниковом, эфирном или кабельном), является основополагающей частью системы условного доступа (СУД) к информационным сервисам [3]. В процессе своего функционирования подсистема шифрования взаимодействует с другими компонентами СУД [4].

Основной функцией подсистемы шифрования информационных сервисов является применение криптографического алгоритма 1 уровня (E_1) только к информационным сервисам, определенным подсистемой управления пользователями и сервисами. Каждому сервису должен соответствовать отдельный ключ шифрования (K_{e1}), предоставляемый подсистемой генерации и управления ключами и обновляющийся с определенным периодом (I_1).

Все идентификаторы информационных сервисов (для DVB-совместимых систем это значения PID [5]), требующих шифрования, а также соответствующие этим идентификаторам актуальные ключи 1 уровня хранятся в специализированной таблице. Добавление нового сервиса выглядит как добавление новой строки в эту таблицу. Обновление ключа заключается в изменении информации в соответствующем поле. При принятии нового транспортного пакета идентификатор, содержащийся в заголовке пакета, последовательно сравнивается со всеми значениями PID в таблице. Если находится строка с таким же идентификатором, то из нее выбирается ключ 1 уровня, который используется для шифрования полезной нагрузки транспортного пакета. Если строка не найдена, то пакет передается в канал связи без шифрования.

Актуальность моделирования подсистем шифрования обусловлена необходимостью решения задач анализа и синтеза систем условного доступа в свете наметившейся в последнее время тенденции использования широковещательных каналов для трансляции различных информационных сервисов в цифровых форматах. В частности, отсутствие доступных и эффективных российских СУД является основным фактором, сдерживающим рост рынка цифрового телевидения и радиовещания.

Аналитическая модель должна устанавливать однозначные зависимости между основными параметрами и характеристиками с учетом задаваемого внешнего воздействия. При этом модель должна быть проста в использовании и должна являться удобным инструментом для проектирования и анализа подсистем шифрования.

Разработка аналитической модели

На первом этапе разработки аналитической модели определим основные исходные параметры системы, характеристики и параметры внешнего воздействия, детальный список и описание которых приведены в [4]. При разработке данной модели мы абстрагируемся от ряда особенностей системы, в частности, не будем уделять внимание аспектам защищенности и криптоустойчивости системы. Кроме того, введем ряд допу-

щений, основные из которых касаются особенностей входящего потока пакетов и распределения времени обслуживания в узлах системы.

Параметры

1. Размер транспортного пакета, S_{tp} (Б).
2. Время обработки одного транспортного пакета, требующего шифрования, v_1 (с).
3. Время обработки одного транспортного пакета, не требующего шифрования, v_2 (с).
4. Время обработки сообщения от подсистемы генерации ключей, v_3 (с).
5. Размер полезной нагрузки транспортного пакета, LP_l (Б).

Характеристики

1. Задержка, вносимая системой шифрования в передачу транспортного потока и выражаемая полным временем нахождения транспортного пакета, требующего шифрования, в подсистеме, T_{delay} (с).
2. Пропускная способность подсистемы, C (пакетов/с).
3. Требуемый объем оперативной памяти, R (байт).

Внешнее воздействие

1. Скорость входящего транспортного потока, V_{ts} (б/с).
2. Интенсивность поступления транспортных пакетов, требующих шифрования, на вход подсистемы, λ_1 (1/с).
3. Интенсивность поступления транспортных пакетов, не требующих шифрования, на вход подсистемы, λ_2 (1/с).
4. Интенсивность поступления сообщений от подсистемы генерации ключей, $\lambda_3(1/c)$.

Для определения основных динамических характеристик подсистемы скремблирования транспортного потока, таких как производительность подсистемы, загрузка и время пребывания транспортного пакета в подсистеме, предполагается использовать математический аппарат теории массового обслуживания. Это обусловлено тем фактом, что данную подсистему можно достаточно легко представить в виде разомкнутой сети массового обслуживания (РСеМО) с двумя узлами (рис. 1). Теоретически размер очереди в данной системе ограничивается размером свободной оперативной памяти и может быть вычислен как отношение размера свободной оперативной памяти к размеру памяти, требуемой для размещения одной заявки.

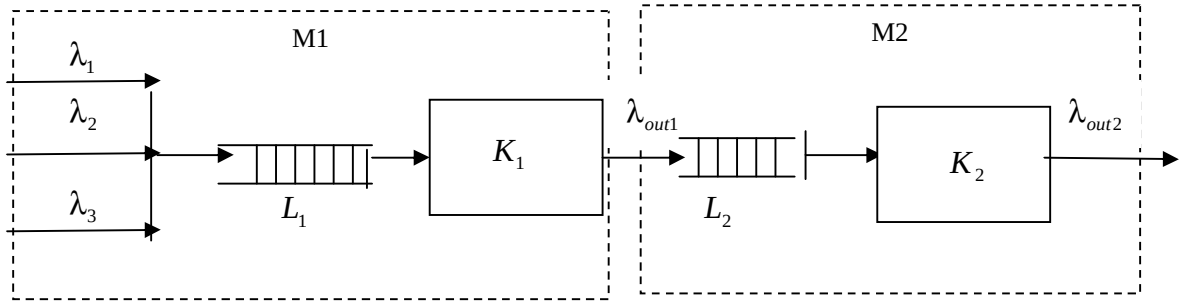


Рис. 1. Модель подсистемы шифрования

В данном случае на вход системы поступает неоднородный, стационарный, ординарный поток заявок без последействия [1, 2], которые могут быть трех типов:

1) транспортные пакеты, требующие шифрования, с интенсивностью λ_1,
2) транспортные пакеты, не требующие шифрования, с интенсивностью λ_2,
3) заявки на обновление ключей 1 уровня с интенсивностью λ_3.

Необходимо отметить, что сумма интенсивностей поступления заявок первого и второго класса является постоянной величиной и в DVB-совместимых системах зависит от скорости входящего транспортного потока и размера транспортного пакета:

$$\begin{cases} \lambda_1 + \lambda_2 = const, \\ \lambda_1 + \lambda_2 = \dfrac{V_{ts}}{S_{tp}}. \end{cases} \quad (1)$$

На первом этапе все заявки обрабатываются шифратором K_1 в соответствии с некоторым алгоритмом [4]. На выходе K_1 получается поток, содержащий как простые, так и зашифрованные пакеты. Этот поток с интенсивностью λ_{out1} поступает в очередь заявок L_2 в соответствии с дисциплиной буферизации FIFO. Этот поток является простейшим одномерным и обслуживается прибором K_2 с целью выравнивания скоростей входящего и исходящего транспортных потоков. В общем случае M_2 реализует так называемый алгоритм «дырявого ведра» [2]. Основное условие нормального функционирования подсистемы шифрования:

$$\lambda_1 + \lambda_2 = \lambda_{out2}. \quad (2)$$

Основной задачей анализа предложенной модели является определение таких характеристик, как полное время задержки транспортного пакета в системе шифрования и требуемый размер оперативной памяти от исходных параметров.

В общем случае полное время задержки будет определяться суммой средних времен пребывания заявок (транспортных пакетов) в СМО М1 и СМО М2, а требуемый размер оперативной памяти – суммой средних длин очередей L_1 и L_2 и их среднеквадратичных отклонений:

$$T_{delay} = u_{cp1} + u_{cp2}, \quad (3)$$

$$R = l_1 + \sigma_1 + l_2 + \sigma_2. \quad (4)$$

Рассмотрим более подробно СМО М1. Каждый тип заявок в М1 может быть максимально полно определен совокупностью следующих параметров:

1) интенсивность поступления заявок на вход системы, $\lambda_i\,(1 \le i \le 3)$,
2) средняя длительность обслуживания заявки в системе, $\nu_i\ (1 \le i \le 3)$,
3) приоритет обслуживания, $\pi_i (1 \le i \le 3)$, $\pi_i \in \Pi : \Pi = \{1,2\}$.

Средние длительности обслуживания заявок всех типов являются постоянными величинами и для заявок 2 и 3 типов зависят только от производительности прибора K_1. Длительность обслуживания заявок первого типа v_1, т.е. транспортных пакетов, требующих шифрования, зависит также от размера полезной нагрузки транспортного пакета LP_l и трудоемкости используемого алгоритма шифрования.

В данной системе приоритеты обслуживания являются относительными, т.е. более приоритетные заявки не прерывают обслуживание менее приоритетных, причем заявки 1 и 2 типа являются равноприоритетными (табл. 1).

Тип заявки	Приоритет
1	2
2	2
3	1

Таблица 1. Приоритеты заявок

Поскольку на вход системы поступают заявки различных типов, у которых различаются как интенсивности поступления, так и средние длины обслуживания, то можно сделать вывод о многомерности входящего потока заявок. В этом случае загрузка прибора потоком заявок *i*-го типа будет составлять

$$\rho_i = \lambda_i \nu_i \ , \tag{5}$$

а суммарная загрузка шифратора будет составлять

$$P = \sum_{i=1}^{3} \rho_i \ . \tag{6}$$

Средняя длина очереди заявок *i*-го типа в системе может быть вычислена с помощью формулы Литтла [1, 2] :

$$l_i = \lambda_i w_i \ , \tag{7}$$

где среднее время ожидания заявки *i*-го типа в очереди w_i, с учетом дисциплины обслуживания с относительными приоритетами, вычисляется по формуле

$$w_1 = w_2 = \frac{\sum_{j=1}^{3} \lambda_j \sigma_j^2}{2(1-\rho_3)(1-P)}, \tag{8}$$

$$w_3 = \frac{\sum_{j=1}^{3} \lambda_j \sigma_j^2}{2(1-P)}, \tag{9}$$

где σ_j^2 – дисперсия времени обслуживания заявки *j*-го типа. Для экспоненциального распределения $\sigma_j^2 = v_j^2$. Тогда среднее время нахождения в системе М1 заявки *i*-го типа может быть вычислено как сумма среднего времени пребывания заявки в очереди и среднего времени обслуживания заявки в приборе:

$$u_i = w_i + v_i \ . \tag{10}$$

Средние времена ожидания w_{cp} и реакции u_{cp} в узле М1 равны

$$w_{cp} = \sum_{i=1}^{4} \frac{\lambda_i}{\Lambda} w_i = 1 \Big/ \Lambda \sum_{i=1}^{4} l_i \ ; \tag{11}$$

$$u_{cp} = \sum_{i=1}^{4} \frac{\lambda_i}{\Lambda} u_i = 1 \Big/ \Lambda \sum_{i=1}^{4} n_i \ , \tag{12}$$

где $\Lambda = \sum_{i=1}^{4} \lambda_i$ – суммарная интенсивность потоков. Средняя длина очереди l_{cp}:

$$l_{cp} = \sum_{1}^{3} \lambda_i w_i \ . \tag{13}$$

Рассмотрим СМО M_2, в которой все заявки – одного типа и требуют одинакового времени обслуживания, дисперсия которого равна нулю. Это система M/D/1. Для расчета основных характеристик СМО M_2 воспользуемся формулами:

$$w_2 = \frac{\rho_2}{2(1-\rho_2)} * \nu_{m2}, \tag{14}$$

$$u_2 = w_2 + \nu_{m2}, \tag{15}$$

$$l_2 = \lambda_{m2} * w_2, \tag{16}$$

где ρ_2 – коэффициент загрузки СМО M_2, рассчитываемый как

$$\rho_2 = \lambda_{m2} * \nu_{m2} \ . \tag{17}$$

Практическое применение аналитической модели

Рассмотрим возможность применения предложенной аналитической модели для решения следующих практических задач:

1) определение задержки передачи информации, вносимой системой шифрования;
2) определение производительности вычислительной системы и размера оперативной памяти, требуемых для реализации определенного алгоритма шифрования при заданном внешнем воздействии;
3) расчет предельных значений характеристик внешнего воздействия в условиях заданной конфигурации вычислительной системы и определенного алгоритма шифрования.

В качестве исходных данных воспользуемся экспериментально определенными значениями основных параметров моделирования для алгоритмов шифрования DES, CSA, ГОСТ 28147-89 и мГОСТ. Алгоритмы DES и CSA (Common Scrambling Algorithm) [7] в настоящее время используются в большинстве систем условного доступа, однако по ряду параметров существенно уступают алгоритму ГОСТ 28147-89 и модифицированной версии этого алгоритма мГОСТ, основная особенность которого заключается в увеличении скорости работы алгоритма за счет уменьшения числа раундов шифрования. Эксперименты проводились на трех вычислительных платформах:

A) IBM-совместимый персональный компьютер на базе процессора INTEL PENTIUM IV с тактовой частотой 1.8 ГГц под управлением ОСРВ VxWorks.
B) IBM-совместимый персональный компьютер на базе процессора INTEL CELERON с тактовой частотой 1.7 ГГц под управлением ОС Windows 2000.
C) IBM-совместимый персональный компьютер на базе процессора INTEL PENTIUM IV с тактовой частотой 3.2 ГГц под управлением ОС Windows 2000.

Основной целью экспериментов являлось определение средних значений времен обслуживания заявок трех классов (v_1, v_2, v_3), представляющих собой исходные параметры моделирования. Результаты экспериментов для данных вычислительных приведены в табл. 2.

Алгоритм шифрования	**A**			**B**			**C**		
	v_1 мкс	v_2 мкс	v_3 мкс	v_1 мкс	v_2 мкс	v_3 мкс	v_1 мкс	v_2 мкс	v_3 мкс
DES-ECB	9	3	56	14	4	68	4	2	43
CSA	46	3	55	54	4	67	31	2	41
ГОСТ28147	15	3	61	23	4	75	10	2	49
мГОСТ	8	3	61	11	4	75	5	2	49

Таблица 2. Экспериментальные данные

Другим важным исходным параметром является время обслуживания пакета в СМО М2 (v_{M2}). Если в системе шифрования функцию выравнивания скоростей выполняет отдельный специализированный периферийный процессор, то параметр v_{M2} является постоянной величиной, не зависящей ни от алгоритма шифрования, ни от производительности основного процессора. Например, для периферийного процессора DTA-140 фирмы DecTec v_{M2} = 7 мкс, в соответствии с фирменной документацией.

В качестве параметров внешнего воздействия воспользуемся данными, полученными в результате исследования трафика в реальных спутниковых каналах связи стандарта DVB-S (табл. 3).

N	Параметры канала (спутник, частота, поляризация, SR,FEC)	V_{ts}, б/с	λ_1, пакет/с	λ_2, пакет/с	λ_3, пакет/с	λ_{M2}, пакет/с
1	HotBird, 11034, V, 27500, 3/4	38000000	9761	15603	2	25364
2	HotBird, 10723, H, 29900,3/4	41300000	11627	16372	1	27999
3	Eutelsat W4, 12456, L, 27500, 3/4	38000000	23972	1392	2	25364

Таблица 3. Параметры внешнего воздействия (данные на октябрь 2005 года)

На основе (14)–(16) определим основные характеристики СМО М2 (табл. 4).

N канала	P_{M2}	w_{M2} мкс	u_{M2} мкс	l_{M2}	σ_{M2}
1	0,18	0,8	7,8	0,03	0,03
2	0,20	0,9	7,9	0,03	0,03
3	0,18	0,8	7,8	0,03	0,03

Таблица 4. Характеристики СМО М2

Необходимо отметить, что характеристики СМО М2 зависят только от интенсивности поступления транспортных пакетов и не зависят ни от алгоритма шифрования, ни от производительности основной вычислительной платформы. Однако если СМО М2 реализована не в виде отдельного периферийного процессора, как в данном случае, а в виде программного модуля, выполняющегося на основной вычислительной платформе, то, безусловно, производительность вычислительной платформы будет сказываться на основных характеристиках СМО М2.

В отличие от М2, основные характеристики СМО М1 находятся в непосредственной зависимости как от применяемого алгоритма шифрования, так и от производительности вычислительной платформы в соответствии с (11)–(13). Результаты расчетов характеристик СМО М1 и всей подсистемы шифрования для различных каналов данных, параметры которых определены в табл. 2, приведены в табл. 5–7.

ПК	Алгоритм	P_{M1}	w_{M1} мкс	u_{M1} мкс	l_{M1}	σ_{M1}	T_d мкс	R
A	DES-ECB	0,13	0,5	5,9	0,01	0,01	13,6	0,08
	CSA	0,50	20,6	40,2	0,52	0,52	47,9	1,10
	ГОСТ28147	0,19	1,5	9,1	0,04	0,04	16,8	0,12
	мГОСТ	0,13	0,4	5,4	0,01	0,01	13,1	0,07
B	DES-ECB	0,20	1,4	9,2	0,03	0,03	17	0,12
	CSA	0,59	35	58,3	0,89	0,89	66	1,83
	ГОСТ28147	0,29	3,8	15,1	0,10	0,10	22,9	0,24
	мГОСТ	0,17	0,9	7,6	0,02	0,02	15,3	0,09
C	DES-ECB	0,07	0,1	2,9	0,00	0,00	10,6	0,06
	CSA	0,33	7,1	20,3	0,18	0,18	28	0,41
	ГОСТ28147	0,13	0,6	5,7	0,02	0,02	13,4	0,08
	мГОСТ	0,08	0,2	3,3	0,00	0,00	11,1	0,06

Таблица 5. Характеристики алгоритмов для 1 канала

Канал 1 характеризуется минимальной интенсивностью пакетов, требующих шифрования, при достаточно высокой общей интенсивности транспортного потока. Результаты моделирования, приведенные в табл. 1, показывают, что для защиты данного канала можно применять любой из исследуемых алгоритмов шифрования на вычислительных платформах A, B и C.

ПК	Алгоритм	P_{M1}	w_{M1}	u_{M1}	l_{M1}	σ_{M1}	T_d	R
A	DES-ECB	0,15	0,6	6,1	0,02	0,02	13,9	0,09
	CSA	0,58	29,8	50,6	0,83	0,83	58,5	1,72
	ГОСТ28147	0,22	1,8	9,8	0,05	0,05	17,6	0,16
	мГОСТ	0,14	0,5	5,6	0,01	0,01	13,5	0,09
B	DES-ECB	0,23	1,6	9,8	0,05	0,05	17,7	0,15
	CSA	0,69	55,7	80,5	1,56	1,56	88,3	3,18
	ГОСТ28147	0,33	4,8	16,7	0,13	0,13	24,6	0,33
	мГОСТ	0,19	1,1	7,9	0,03	0,03	15,8	0,11
C	DES-ECB	0,08	0,1	2,9	0,00	0,00	10,8	0,06
	CSA	0,39	9,3	23,3	0,26	0,26	31,2	0,57
	ГОСТ28147	0,15	0,7	6	0,02	0,02	13,9	0,10
	мГОСТ	0,09	0,2	3,4	0,01	0,01	11,3	0,07

Таблица 6. Характеристики алгоритмов для 2 канала

Основной особенностью 2 канала является заметное ухудшение характеристик подсистемы шифрования при использовании алгоритма CSA на вычислительных платформах с низкой производительностью.

ПК	Алгоритм	P_{M1}	w_{M1}	u_{M1}	l_{M1}	σ_{M1}	T_d	R
A	DES-ECB	0,22	1,3	9,9	0,03	0,03	17,7	0,11
	CSA	1,11						
	ГОСТ28147	0,36	4,3	18,6	0,11	0,11	26,4	0,27
	мГОСТ	0,20	0,9	8,7	0,02	0,02	16,5	0,10
B	DES-ECB	0,34	3,6	17	0,09	0,09	24,8	0,23
	CSA	1,30						
	ГОСТ28147	0,56	14,4	36,3	0,36	0,36	44,1	0,78
	мГОСТ	0,27	2,1	12,6	0,05	0,05	20,4	0,15
C	DES-ECB	0,10	0,2	4,1	0,01	0,01	11,9	0,06
	CSA	0,75	45,4	74,8	1,15	1,15	82,5	2,35
	ГОСТ28147	0,24	1,6	11,2	0,04	0,04	18,9	0,13
	мГОСТ	0,12	0,3	5,2	0,01	0,01	12,9	0,07

Таблица 7. Характеристики алгоритмов для 3 канала

Канал 3 характеризуется максимальной долей пакетов, требующих шифрования, в общей структуре транспортного потока. Результаты моделирования, приведенные в табл. 7, показали невозможность применения алгоритма CSA на вычислительных платформах с низкой производительностью для защиты данного канала, так как коэффициент загрузки системы больше 1. Остальные алгоритмы могут применяться для защиты 3 канала с разной эффективностью.

Для определения производительности вычислительной системы и размера оперативной памяти, требуемых для реализации определенного алгоритма шифрования при

заданном внешнем воздействии, построим графики зависимостей $T_d(v_1)$ (рис. 2) и $R(v_1)$ (рис. 3) для трех каналов связи.

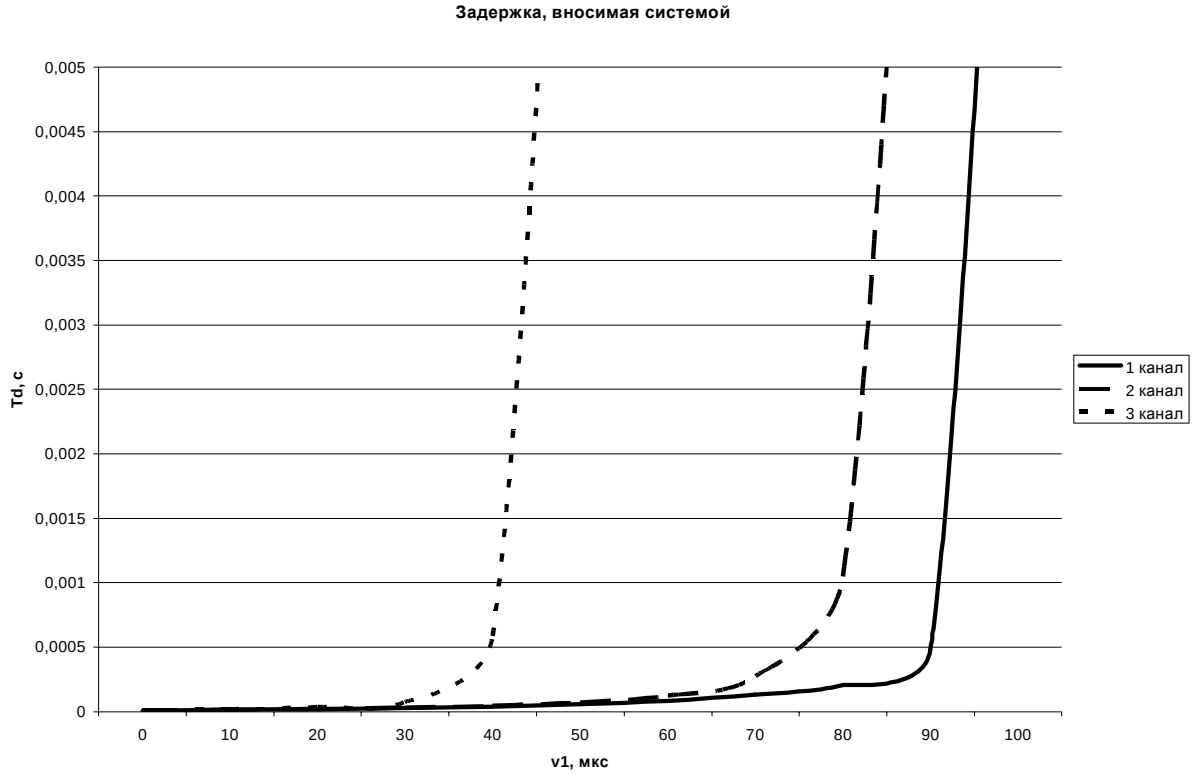


Рис. 2. Зависимость общей задержки от времени обработки пакета, требующего шифрования

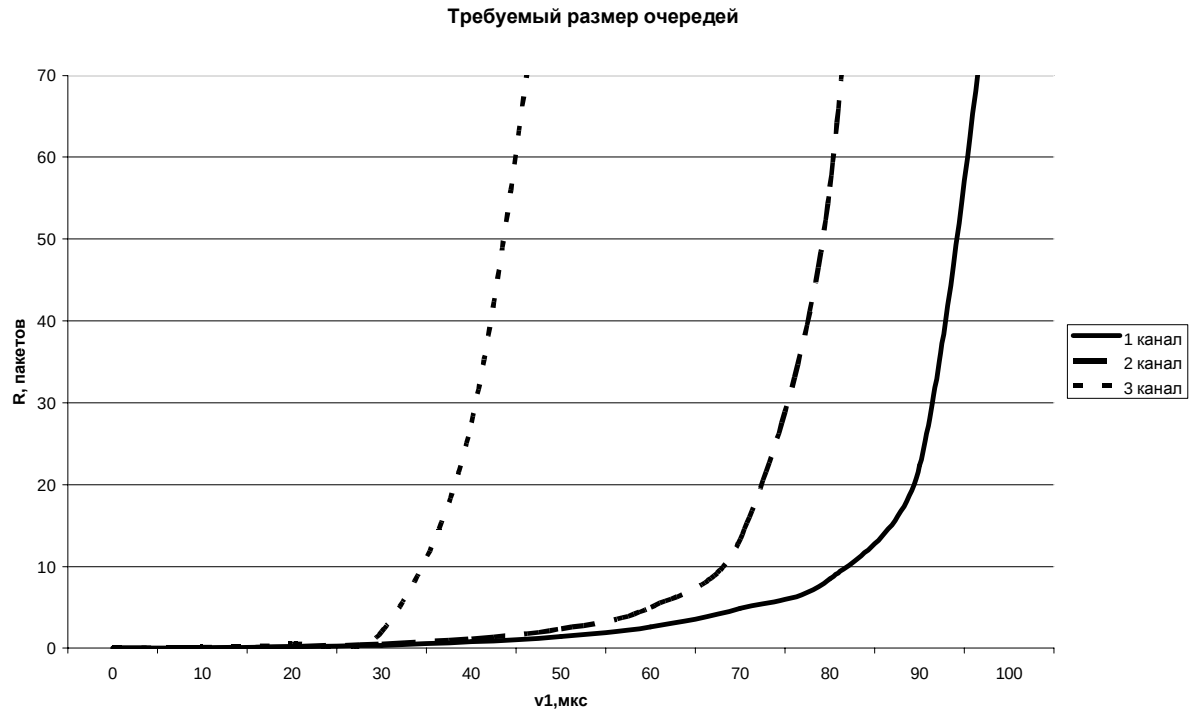


Рис. 3. Зависимость требуемого размера очередей от времени обработки пакета, требующего шифрования

Анализ зависимостей на рис. 2 и рис. 3 показывает, что производительность подсистемы шифрования, определяемая трудоемкостью алгоритма шифрования и быстродействием вычислительной платформы, должна обеспечивать время обработки пакета, требующего шифрования (v_1), не больше 90 мкс для 1 канала связи, не больше 70 мкс для 2 канала связи и не больше 40 мкс для 3 канала связи. В противном случае характеристики системы будут принимать недопустимые значения.

Заключение

В работе предложена аналитическая модель подсистемы шифрования информационных сервисов, которая является важной частью системы условного доступа. Модель позволяет осуществлять расчет и анализ основных характеристик, таких как общая задержка, вносимая подсистемой шифрования, требуемый размер оперативной памяти, пропускная способность подсистемы.

На основе предложенной аналитической модели было проведено детальное исследование и аналитическое сравнение таких алгоритмов шифрования, как DES, CSA, ГОСТ 28147 и мГОСТ. Выявлены основные достоинства и недостатки, присущие этим алгоритмам, определены величины задержек и требуемый размер оперативной памяти. Кроме того, на основе полученных зависимостей приведены рекомендации по требуемой производительности подсистемы шифрования для трех тестовых каналов связи.

Литература

1. Альянх И.Н. Моделирование вычислительных систем. Л.: Машиностроение, 1988.
2. Столлингс В. Современные компьютерные сети. СПб: Питер, 2003.
3. Костин М.В. Системы условного доступа // Телеспутник. 2004. №11. С. 62–64.
4. Костин М.В. Концептуальная модель системы условного доступа // Вестник II межвузовской конференции молодых ученых. СПб: СПбГУ ИТМО, 2005. Т.1. С. 129–137.
5. ETSI EN 300 421 v1.1.2:1997 Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for 11/12 GHz satellite services, ETSI, 1997.
6. ETSI EN 300 468 v1.4.1:2000 Digital Video Broadcasting (DVB); Specification for Services Information (SI) in DVB systems, ETSI, 2000.
7. ETR 289:1996 Digital Video Broadcasting (DVB); Support for use of scrambling and Conditional Access (CA), ETSI, 1996.

АНАЛИТИЧЕСКАЯ МОДЕЛЬ ПОДСИСТЕМЫ ПРИЕМА И ОБРАБОТКИ СЕРВИСНОЙ ИНФОРМАЦИИ

А.В. Костина

Научный руководитель – доктор технических наук, профессор Т.И. Алиев

В работе предлагается аналитическая модель подсистемы приема и обработки сервисной информации (интегрированный приемник/декодер), передаваемой в транспортном потоке стандарта DVB, приведены основные характеристики модели исследуемой системы, приведен анализ и интерпретация результатов.

Введение

Под сервисной информацией будем понимать разнородные данные, инкапсулируемые в реальный транспортный поток и не относящиеся непосредственно к базовым сервисам, таким как видео и аудио. Эти данные могут представлять собой файлы программного обеспечения интегрированных приемников/декодеров, наборы индивидуальных или групповых пользовательских настроек, исполняемый код, базы данных, набор сайтов и другую разнородную информацию.

Основная особенность трафика такого типа заключается в необходимости периодического повторения передачи одной и той же информации, в отличие от базовых сервисов, передаваемых в режиме реального времени. Это отличие обусловлено высокой сложностью, а во многих случаях – невозможностью организации обратного канала связи между принимающей и передающей стороной, предназначенного для установления соединения между приемником и передатчиком с целью обеспечения надежного, гарантированного приема данных. Другой важный аспект, обуславливающий необходимость периодичности передачи данных, заключается в многоадресном характере передачи, т.е. в большинстве случаев передаваемые файлы предназначены для большого числа абонентов (от нескольких десятков до нескольких миллионов). Очевидно, что время начала ожидания данных различными клиентскими терминалами с большой вероятностью будет различаться. Соответственно, чем чаще повторяются данные, тем меньше время ожидания принимающего терминала. Однако увеличение частоты повторов приводит к увеличению доли передаваемого файла в транспортном потоке и, соответственно, к увеличению стоимости передачи.

Учитывая перечисленные особенности спутникового канала связи, возникает задача определения среднего времени приема файла, при инкапсуляции которого в реальный транспортный поток используется карусельный алгоритм, с учетом статического способа разделения цифрового спутникового канала связи между различными сервисами. Применение карусельного алгоритма заключается в разделении всех файлов на секции фиксированного размера и периодической передаче этих секций в определенной последовательности в свободной части транспортного потока. Алгоритм, выполняющийся на принимающей стороне, обеспечивает фильтрацию входящей информации и ассемблирование ее во внутренних буферах с целью последующей обработки.

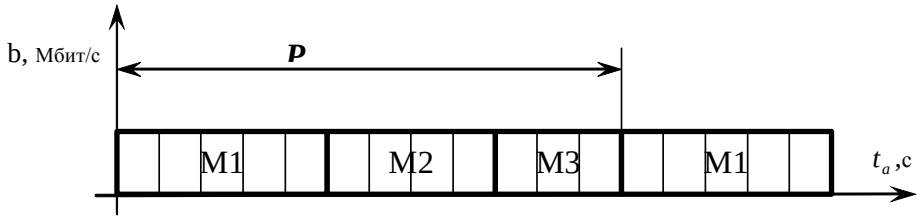


Рис. 1. Пример карусели файлов

На рис. 1 приведен пример карусели, которая состоит из трех файлов (объектов), после последовательной передачи всех файлов цикл карусели начинается снова. Здесь период карусели P (с) – время, за которое совершается передача всех объектов карусе-

ли. Для обеспечения пользователя возможностью быстрого доступа к передаваемому объекту файлы разбиваются на части (секции).

Аналитическая модель системы приема и обработки сервисной информации может быть полезна и необходима как разработчикам приемопередающего оборудования, так и операторам и провайдерам информационных услуг. Можно выделить несколько основных задач, решение которых возможно аналитическими методами. Во-первых, это проблема определения минимальных требований к производительности и объему оперативной памяти принимающей системы при известных параметрах внешнего воздействия. Во-вторых, это задача определения предельных значений параметров внешнего воздействия, при которых будет возможно нормальное функционирование системы с заданными параметрами производительности и объемом оперативной памяти. Кроме того, аналитическая модель может быть полезна для решения задач анализа, синтеза и оптимизации систем приема и обработки сервисной информации.

Разработка аналитической модели

Рассмотрим систему приема и обработки сервисной информации (интегрированный приемник/декодер), на вход которой поступает поток пакетов, имеющих одинаковый размер (в соответствии со стандартом DVB пакет транспортного потока имеет постоянный размер 188 байт). Входящий трафик, генерируемый источником, представляет собой суперпозицию пакетов, принадлежащих разным сервисам.

Исходными данными для модели являются следующие параметры:

1. количество передаваемых файлов дополнительных информационных сервисов в карусели;
2. размеры предаваемых файлов;
3. размер секции файла дополнительного информационного сервиса;
4. матрица вероятностей передач;
5. размер свободной части полосы пропускания спутникового канала связи;
6. интенсивность поступления секций файлов на вход системы.

В качестве выходных характеристик будем рассматривать:

1. время принятия файла фиксированного размера;
2. среднее время ожидания начала передачи файла;
3. вероятность потери секции файла.

Среднее время принятия полного файла связано со средним временем ожидания секции и средним временем обработки секции. Очевидно, что время приема файла (T_r, с) будет складываться из времени ожидания секций файла (T_w, с) и времени их обработки (T_s, с):

$$T_r = (1 + p_{sl}) * \sum_{i=0}^{N-1} (T_w + T_s), \qquad (1)$$

где p_{sl} – вероятность потери секции файла, N – число секций в одном файле. Таким образом, для окончательного расчета T_r необходимо определить аналитические зависимости для p_{sl}, T_w и T_s.

Расчет среднего времени ожидания заданного файла

Метод расчета среднего времени ожидания заданного файла определяется стратегией ожидания начала передачи файла. В общем случае различается две основных стратегии ожидания:

1. ожидание первой секции файла;

2. ожидание любой секции файла.

В первом случае приемник принимает и обрабатывает только последовательно приходящие секции, начиная с первой. Если секция будет пропущена, приемник будет ожидать ее в течение всего периода карусели. Стратегия ожидания любой секции представляет собой более гибкий подход, при котором в случае пропуска секции приемник будет ожидать любую следующую секцию, принадлежащую данному файлу, вне зависимости от ее номера в последовательности.

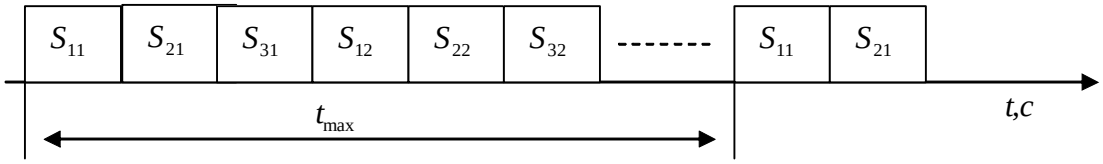


Рис. 2. Стратегия ожидания первой секции

При использовании стратегии ожидания первой секции (рис. 2) значение времени ожидания T_w находится в интервале $t_{\min} \le T_w \le t_{\max}$, где $t_{\min} = 0$, а $t_{\max}$ для данного случая рассчитывается по формуле

$$t_{\max} = \frac{\sum_{i=1}^{N_f} S_{fi}}{b},$$

где S_{j_i} – размер *i*-го файла (байт), N_f – число файлов в карусели, *b* – размер части полосы пропускания спутникового канала связи, выделенный для передачи дополнительных сервисов (байт/с). Тогда среднее время ожидания равно

$$T_w = \frac{t_{\max} - t_{\min}}{2} = \frac{\sum_{i=1}^{N_f} S_{fi}}{2*b}. \qquad (2)$$

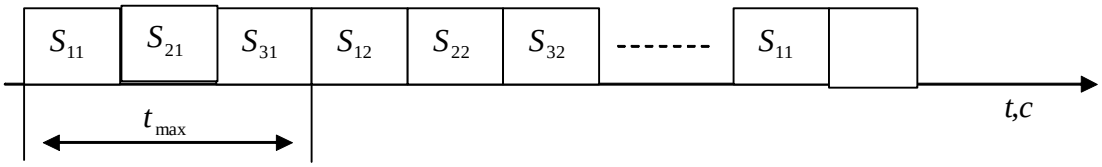


Рис. 3. Стратегия ожидания любой секции

При использовании стратегии ожидания любой секции (рис. 3) имеем:

$$t_{\max} = \frac{\sum_{i=1}^{N_f} S_{si}}{b},$$

где S_{si} – размер секции *i*-го файла (байт), следовательно,

$$T_w = \frac{t_{\max} - t_{\min}}{2} = \frac{\sum_{i=1}^{N_f} S_{si}}{2*b}. \qquad (3)$$

Расчет среднего времени обработки секции файла

Для нахождения среднего времени обработки (T_s) секции файла представим систему приема информации в виде разомкнутой стохастической сети (рис. 4), где S1, S2, S3, S4 – системы массового обслуживания.

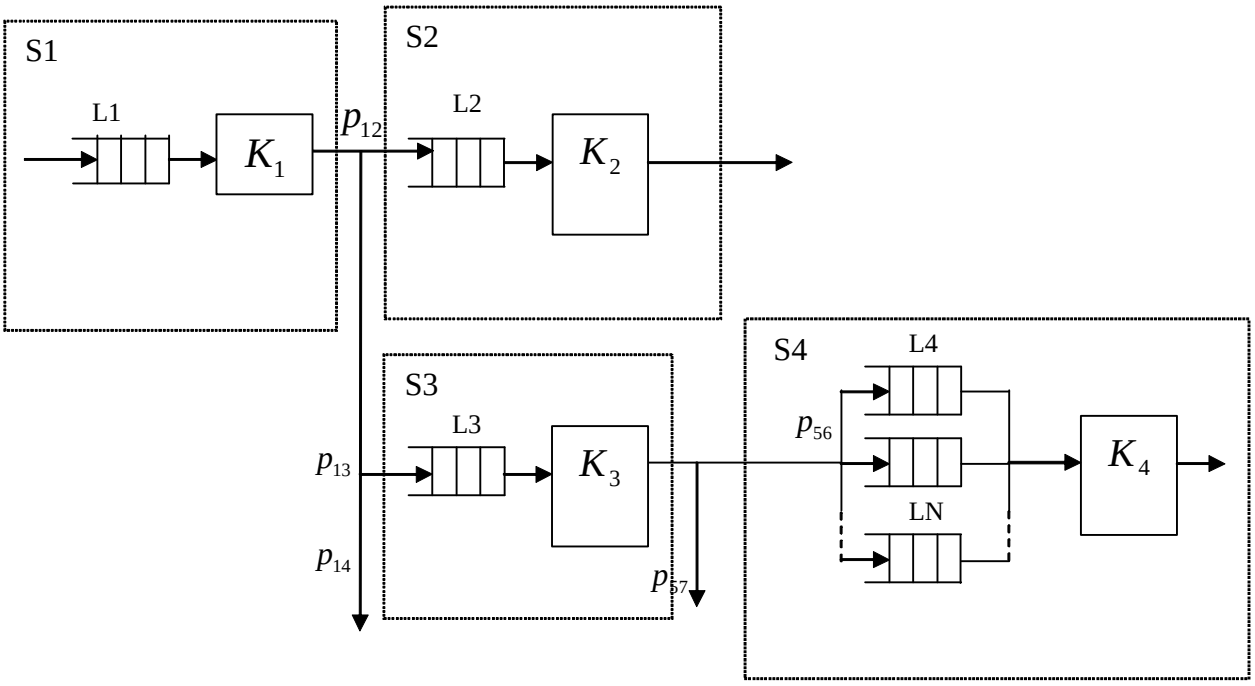


Рис. 4. Система приема и обработки сервисной информации

Перемещение пакетов между узлами данной РСеМО осуществляется в соответствии с маршрутной матрицей P:

$$P = \begin{array}{c|cccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \hline 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & p_{12} & p_{13} & p_{14} & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 & p_{56} & p_{57} \\ 6 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}.$$

Рассмотрим СМО S1, отображающую процесс функционирования демультиплексора. Принятый транспортный поток, представляющий собой суперпозицию элементарных потоков, каждый из которых идентифицируется уникальным PID (Packet Identifier) значением, помещается в буфер L1. Прибор K_1 выполняет фильтрацию по PID значению, т.е. пропускает только пакеты с определенными идентификаторами, остальные пакеты игнорируются. Кроме того, на этом этапе на основе информации, передаваемой в заголовке транспортного пакета, определяется его тип, характеризующий принадлежность пакета к соответствующей группе сервисов – базовым (видео или аудио) или дополнительным (сервисные данные). На основе типа пакета определяется его дальнейший маршрут. Таким образом, транспортные пакеты, относящиеся к базовым сервисам, попадают в аудио- и видеодекодеры для последующего восстановления, т.е. направляются в СМО S2, а транспортные пакеты дополнительных сервисов попадают на обработку в СМО S3.

В данной работе не проводится исследование аудио- и видеопотоков, интерес представляет только вероятность поступления транспортного пакета в СМО S2, так как

эта вероятность оказывает влияние на интенсивность поступления пакетов в СМО S3. К основным задачам СМО S3 относится дополнительная фильтрация данных и формирование секций, представляющих собой минимальные логические части дополнительных информационных сервисов. Каждая секция может принадлежать только одному сервису. После формирования секции, которая может состоять из одного или нескольких транспортных пакетов, из СМО S3 секция направляется в очередь, соответствующую данному информационному сервису.

Из очередей $L_4, L_5, ..., L_N$ секции извлекаются прибором K_4 для окончательной обработки (записи в файл, передачи по каналам связи, отображения на экране терминала и т.д.) в соответствии с определенной дисциплиной обслуживания (ДО), что учитывает разноприоритетный характер информационных сервисов.

Основной задачей предложенной РСеМО является определение среднего времени обработки секции файла в зависимости от исходных параметров. Полное время приема секции файла будет складываться из среднего времени пребывания в СМО S1, СМО S3 и СМО S4:

$$u = (u_1 + u_3 + u_4) * N_{TP},$$

где u_1 – среднее время пребывания заявки в СМО S1, u_3 – среднее время пребывания в СМО S3, u_4 – среднее время пребывания заявки в СМО S4, N_{TP} – число транспортных пакетов в секции. Интенсивность входящего потока λ_0 вычисляется по формуле:

$$\lambda_0 = \frac{c}{k},$$

где с – скорость входящего транспортного потока, k – размер пакета (для DVB стандарта размер транспортного пакета составляет 188 байт).

Рассмотрим более подробно СМО S1, на вход которой поступает поток заявок с интенсивностью λ_0. Так как транспортный поток, поступающий на вход сети, обладает такими свойствами, как стационарность, ординарность, отсутствие последействия, то в качестве входящего потока заявок рассматривается простейший поток. Представим СМО S1 как одноканальную систему массового обслуживания типа M/M /1, характеризующуюся экспоненциальным распределением продолжительности времени обслуживания, и как СМО типа М/Д /1. Это обусловлено тем фактом, что данная подсистема может иметь две реализации: программную и аппаратную.

Входящий поток заявок может быть определен следующими параметрами:

- интенсивность входящего потока λ_0;
- среднее время обслуживания заявки в СМО S1 v_1.

Для определения основных характеристик СМО S1, таких как среднее время пребывания заявки u_1, среднее время ожидания заявки w_1, средняя длина очереди l_1 воспользуемся формулами:

$$w_1 = \frac{\rho_1 * \nu_1}{1 - \rho_1}, \quad (4)$$

где ρ_1 – коэффициент загрузки СМО S1, определяемый как $\rho_1 = \lambda_0 * \nu_1$,

$$l_1 = \frac{\lambda_0 * \nu_1}{1 - \rho_1}, \quad (5)$$

$$u_1 = w_1 + \nu_1 = \frac{\nu_1}{1 - \rho_1}. \quad (6)$$

Рассмотрим СМО S1 как случай аппаратной реализации, т.е. как систему типа M/D/1, Среднее время обслуживания заявки является постоянной величиной и зависит

только от производительности прибора K_1. В этом случае при вычислении основных характеристик применимы формулы:

$$w_1 = \frac{\rho_1 * \nu_1}{2(1-\rho_1)},$$

$$u_1 = w_1 + \nu_1.$$

На вход СМО S3 поступают транспортные пакеты дополнительных информационных сервисов с интенсивностью λ_3. Так как в большинстве случаев СМО S3 имеет программную реализацию, то ее также можно представить в виде модели M/M/1 и использовать формулы (4)–(6) для расчета основных характеристик.

Наибольший интерес представляет СМО S4, на вход которой поступают заявки M типов с интенсивностями λ_1 h λ_M. Пусть $\Lambda = \sum_{i=1}^{M} \lambda_i$ – интенсивность суммарного потока заявок. СМО S4 содержит только один обслуживающий прибор. Пакеты выбираются из очереди в соответствии с дисциплиной обслуживания. Среднее время ожидания пакетов различных типов будет зависеть от используемой дисциплины обслуживания.

Таким образом, каждый тип заявок может быть определен следующей совокупностью параметров:

- интенсивность входящего потока заявок λ_i, $1 \geq i \leq M$;
- среднее время обслуживание заявки v_i, $1 \geq i \leq M$;
- приоритет обслуживания заявок K_i, $1 \geq i \leq M$.

В случае использования бесприоритетной дисциплины обслуживания среднее время ожидания заявок всех типов одинаково и равно

$$w_i = \frac{\sum_{i=1}^{M} \lambda_i \nu_i^{(2)}}{2(1-\sum_{i=1}^{M} \rho_i)},$$

где $\nu_i^{(2)}$ – дисперсия времени обслуживания заявки *i*-го типа. Для экспоненциального распределения $\nu_i^{(2)} = 2\nu_i^2$. Дисциплина обслуживания с относительными приоритетами определяется выражением

$$w_i = \frac{\sum_{i=1}^{M} \lambda_i \nu_i^{(2)}}{2(1-\sum_{i=1}^{M-1} \rho_i)(1-\sum_{i=1}^{M} \rho_i)},$$

а дисциплина обслуживания с абсолютными приоритетами – выражением

$$w_i = \frac{\sum_{i=1}^{M} \lambda_i \nu_i^{(2)}}{2(1-\sum_{i=1}^{M-1} \rho_i)(1-\sum_{i=1}^{M} \rho_i)} + \frac{\sum_{i=1}^{M-1} \rho_i \nu_i}{1-\sum_{i=1}^{M-1} \rho_i}.$$

Среднее время ожидания для заявки *i*-го класса равно

$$u_i = w_i + \nu_i.$$

Среднюю длину очереди заявок *i*-го класса вычислим с помощью формулы Литтла:

$$l_i = \lambda_i \nu_i.$$

Среднее время ожидания и среднее время пребывания заявки из входящего потока в СМО S4 определим по формуле:

$$w_{cp} = \sum_{i=1}^{M} \frac{\lambda_i}{\Lambda} w_i \ , \ u_{cp} = \sum_{i=1}^{M} \frac{\lambda_i}{\Lambda} u_i \ .$$

Практическое применение аналитической модели

Рассмотрим возможность применения аналитической модели для нахождения среднего времени приема файла фиксированного размера. В качестве параметров будем рассматривать следующие исходные данные и ограничения:

- в карусели транслируются 5 файлов, размеры которых приведены в табл. 1, также в этой таблице приведены размеры секций, на которые делятся файлы;

№ файла	Размер файла, байт	Размер секции файла, байт	Количество транспортных пакетов
1	26112	1024	139
2	201242	4096	1070
3	1112040	1024	5915
4	2097152	4096	11155
5	79128	1024	421

Таблица 1. Параметры файлов

- размеры свободной части полосы пропускания и другие параметры тестовых каналов связи, ретранслируемых спутником Hot Bird (13 в.д), приведены в табл. 2.

№ канала	Частота, GHz	SR, сим/с	FEC	V, бит/с	V_f, бит/с	λ_0, 1/с	λ_1, 1/с
1	12713	27500000	3/4	38015700	3601200	25276	2394
2	11131	5632000	3/4	7785400	1785000	5176	1187
3	10957	4340000	3/4	6010000	1600000	5898	1063

Таблица 2. Параметры тестовых каналов

Для определения времени обслуживания в СМО S1, СМО S3 и СМО S4 проведем эксперименты на двух вычислительных платформах:

А) интегрированный приемник/декодер (IRD) FTA 7001 фирмы General Satellite на базе специализированного процессора ST5518 с частотой 81 МГц под управлением ОСРВ OS/20;

В) персональный компьютер на базе процессора Intel Pentium IV с частотой 1,82 ГГц под управлением ОС Windows 2000 с интегрированным периферийным модулем SkyStar2 фирмы TechnoTrend.

Основной особенностью платформы (А) является аппаратная реализация PID-фильтрации, т.е. СМО S1. Это обусловливает относительно малое время обслуживания ν_1, в отличие от времени обработки пакетов в других узлах, в которых обслуживание выполняется программным способом. Платформа (В) характеризуется программной реализацией всех узлов PCeMO, однако время обработки в этих узлах существенно ниже, чем в аналогичных узлах платформы (А), что обусловлено высоким быстродействие универсального процессора. Результаты экспериментов для двух платформ приведены в табл. 3. Предполагается, что все исследуемые файлы имеют одинаковое предназначение (например, должны только размещаться в оперативной памяти с последующей передачей управления), а, следовательно, имеют одинаковое время обработки пакетов в СМО S4.

№	ν_1, мкс	ν_3, мкс	ν_4, мкс
A	25	367	434
B	7	21	37

Таблица 3. Экспериментальные данные

Распределим приоритеты между передаваемыми файлами, причем более высокому приоритету будет соответствовать меньшее число (табл. 4).

№ файла	Приоритет
1	1
2	3
3	2

Таблица 4. Приоритеты файлов

Из (1) вычислим среднее время принятия файлов №1 с применением первого T_{w1} (2) и второго T_{w2} (3) методов расчета времени ожидания секции. В табл. 5–7 приведены соответствующие результаты при использовании в СМО S4 бесприоритетной ДО, с относительными приоритетами и с абсолютными приоритетами, соответственно.

№ канала	№ платформы	T_{w1}, с	T_s, с	T_{r1}, с	T_{w2}, с	T_{r1}, с
1	A	4	0,53	4,53	0,001564	0,53
	B		0,01	4,01		0,01
2	A	7,8	0,17	7,97		0,17
	B		0,01	7,81	0,003155	0,01
3	A	3	0,5	3,5		0,5
	B		0,01	3,01	0,001238	0,01
4	A	8,7	0,16	8,86		0,16
	B		0,01	8,01	0,00352	0,01

Таблица 5. Характеристики системы при бесприоритетной ДО. Файл №1

№ канала	№ платформы	T_{w1}, с	T_s, с	T_{r1}, с	T_{w2}, с	T_{r1}, с
1	A	4	0,5	4,5	0,001564	0,5
	B		0,01	4,01		0,01
2	A	7,8	0,17	7,97		0,17
	B		0,01	7,81	0,003155	0,01
3	A	3	0,5	3,5		0,5
	B		0,01	3,01	0,001238	0,011
4	A	8,7	0,16	8,86		0,16
	B		0,01	8,71	0,00352	0,01

Таблица 6. Характеристики системы при ДО с относительными приоритетами. Файл №1

№ канала	№ платформы	T_{w1}, с	T_s, с	T_{r1}, с	T_{w2}, с	T_{r1}, с
1	A	4	0,49	4,49	0,001564	0,49
	B		0,01	4,01		0,011
2	A	7,8	0,157	7,957		0,16
	B		0,02	7,82	0,003155	0,023
3	A	3	0,49	3,49		0,49
	B		0,01	3,01	0,001238	0,011
4	A	8,7	0,15	8,85		0,15
	B		0,01	8,71	0,00352	0,013

Таблица 7. Характеристики системы при ДО с абсолютными приоритетами. Файл №1

Сравнительный анализ результатов, приведенных в табл. 5–7, показывает, что время приема файла №1 при использовании дисциплины обслуживания с абсолютными приоритетами сокращается. Кроме того, необходимо отметить, что платформа (A) имеет гораздо более худшие характеристики, чем платформа (B), что, в основном, обусловлено разницей в производительности вычислительных платформ. Построим график зависимостей (рис. 5) времени обслуживания в СМО S4 от интенсивности входящей заявок при использовании бесприоритетной дисциплины обслуживания. Анализируя зависимости, можно сделать вывод, что при интенсивности более чем 2300 заявок в секунду характеристики системы будут принимать недопустимые значения.

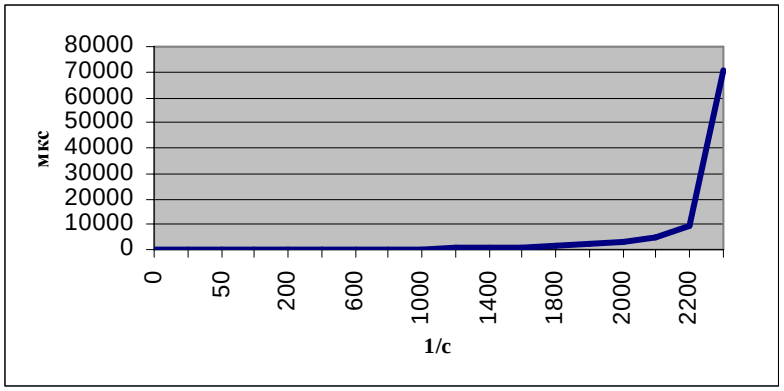


Рис. 5. Зависимость времени обслуживания в СМО S4 от интенсивности заявок

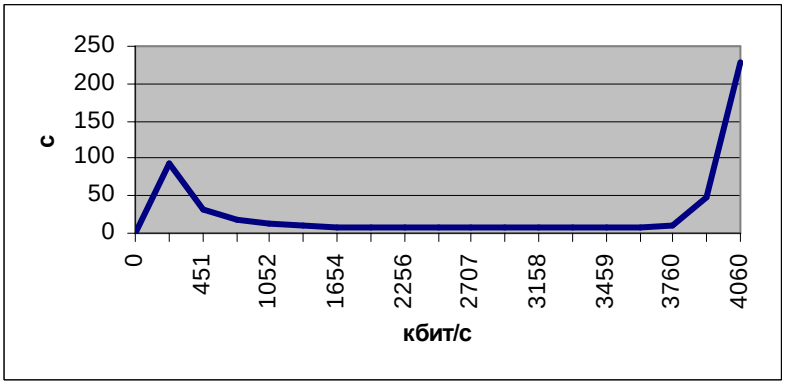


Рис. 6. Зависимость времени приема файла №2 от скорости передачи сервисной информации

Для исследования времени приема файла №2 от скорости передачи сервисной информации построим график (рис. 6), в результате анализа которого можно сделать вывод относительно платформы (А): при входящей скорости более чем 4,1 Мбит/с характеристики системы будут принимать недопустимые значения.

Заключение

В работе предложена аналитическая модель системы приема и обработки сервисной информации (интегрированный приемник/декодер), передаваемой в транспортном потоке стандарта DVB, получены зависимости для расчета основных характеристик модели исследуемой системы, проведен анализ полученных результатов. На основе предложенной аналитической модели определены средние времена приема тестовых файлов в зависимости от исходных условий.

Литература

1. Клейнрок Л. Теория массового обслуживания. М.: Машиностроение, 1979. 432 с.
2. Матвеев В.Ф., Ушаков В.Г. Системы массового обслуживания. М.: Техносфера, 2005. 592 с.
3. Вишневский В.М., Ляхов А.И., Портной С.Л., Шахнович И.В. Широкополосные беспроводные сети передачи информации. М.: Изд-во МГУ, 1984. 240 с.
4. ETSI EN 300 468 v1.4.1:2000 Digital Video Broadcasting (DVB); Specification for Services Information (SI) in DVB systems // ETSI, 2000.
5. 13818-1 Coding of Moving Pictures and Associated Audio – Part 1: Systems, (MPEG –2) // ISO, 1996.

ПОВЫШЕНИЕ ЭФФЕКТИВНОСТИ ПЕРЕДАЧИ РЕЧЕВЫХ СИГНАЛОВ

М.Б. Будько

Научный руководитель – кандидат технических наук Г.П. Жигулин

Цель исследования заключается в выявлении современных подходов к передаче потокового аудио, оптимизации перспективных механизмов и реализации собственной системы передачи акустических сигналов, обладающей способностью динамической настройки в зависимости от состояния сети. В статье рассматривается механизм прерывистой передачи и описываются методы, положенные в основу разработанного алгоритма определения речевой активности.

Введение

Большинство существующих сетей связи имеют достаточно развитую по охвату и содержанию инфраструктуру ввиду доступности для специализирующихся в области телекоммуникаций компаний высокотехнологичных средств. Поэтому для поддержания конкурентоспособности многие уже сейчас строят свою политику на предоставлении дополнительных услуг, что будет главным направлением развития цифровых сетей. К таким услугам относятся: видео и аудио по запросу, приложения электронного бизнеса и др.

Перспективными для передачи аудиоконтента как составляющей указанных услуг являются:

– механизм потоковой передачи;
– механизм прерывистой передачи (кодирование с переменной скоростью).

Потоковая передача аудио позволяет пользователю начать прослушивание без полной загрузки файла, а также выполнять команды управления.

Для передачи речи по сетям существуют два метода кодирования:

– с постоянной скоростью;
– с переменной скоростью.

Постоянная скорость кодирования является традиционным подходом, предполагающим применение одной и той же схемы кодирования ко всем речевым сегментам. Но даже при использовании алгоритмов многоимпульсного и кодового возбуждения, обеспечивающих высокое качество при относительно невысоких скоростях передачи, на скорости 4 кбит/с и меньше качество речи снижается [1].

Одна из возможностей дальнейшего снижения скорости передачи – переход к переменной скорости кодирования, основанной на том, что информация, необходимая для описания сигнала, изменяется во времени.

В речи, особенно при двустороннем разговоре, существуют паузы между звуками, словами и существенные паузы между предложениями. Средняя длительность пауз может составлять до половины общего времени разговора двух абонентов. Ввиду различной информационной избыточности в представлении периодов молчания и звуков активной речи на соответствующих сегментах могут быть применены разные схемы кодирования.

Если одновременно организовано несколько соединений, то экономия полосы пропускания оказывается существенной, что позволяет организовать еще ряд соединений без выделения или прокладки дополнительных каналов связи.

Высокая эффективность использования сетей достигается, когда применяется коллективный доступ с кодовым разделением (CDMA – Code Division Multiple Access) [1]. Системы с CDMA включают модуль контроля переменной скорости (VRCU – Variable Rate Control Unit), созданный для обеспечения оптимального распределения полосы пропускания канала между различными источниками информации. Модуль

анализирует мультиплексируемые каналы для определения скорости передаваемых по ним данных и осуществляет распределение полосы пропускания (рис. 1).

Рис. 1. Схема управления потоками данных с переменными скоростями

Каждый источник информации формирует соответствующий информационный поток $r_i(n)$, который является функцией кадра n и поступает в кодер канала [1]. Затем полученные цифровые потоки (каждый со своей избыточностью, переменной скоростью $R_i(n)$ и динамически выделенной полосой пропускания) поступают в мультиплексор.

Модуль VRCU для обеспечения оптимального распределения ресурсов канала связи анализирует [1]:

- потребности источников информации;
- потребности системы;
- потребности пользователей;
- возможности канала связи.

Таким образом, поскольку современные сети обладают возможностями управления потоками, то актуальной является задача разработки самих алгоритмов кодирования с переменной скоростью, а также модификации существующих для перехода к переменным скоростям передачи.

Основная часть

Существуют следующие способы кодирования речи с переменной скоростью:

- на основе фонетической классификации;
- на основе энергетической классификации;
- обеспечивающие постоянное качество синтезированной речи.

В ходе исследования были проанализированы:

- указанные способы, на примерах конкретных реализаций;

– стандарты ITU-T G729 Annex B и G723.1 Annex A, включающие описания алгоритмов определения периодов молчания;

– RFC 3389, описывающий формат полезной нагрузки комфортного шума.

В результате была построена общая схема кодирования-декодирования с переменной скоростью и выделены характерные для указанной схемы функциональные блоки (рис. 2):

– блок определения речевой активности (VAD – Voice Activity Detector);

– генератор комфортного шума (CNG – Comfort Noise Generator).

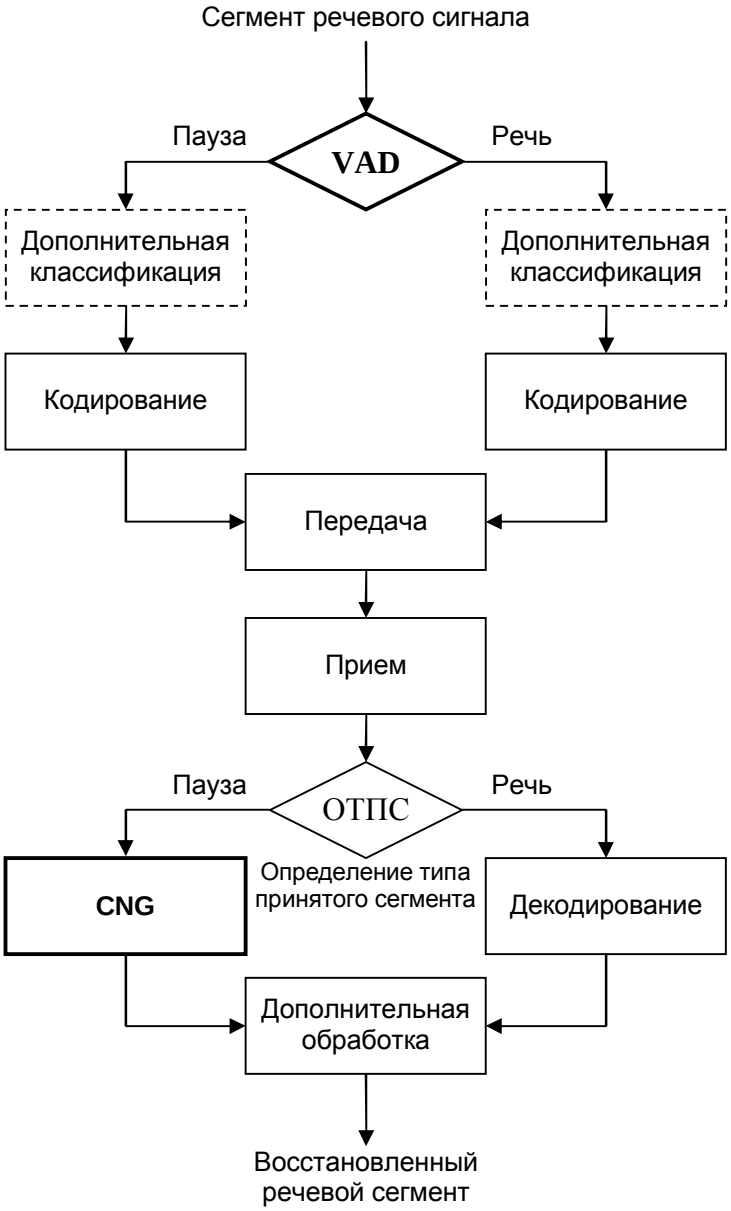


Рис. 2. Общая схема кодирования-декодирования с переменной скоростью

Основные требования, предъявляемые к алгоритмам VAD и CNG:

– низкие вычислительные затраты, поскольку работа ведется на речевом фрагменте «вместе» с кодированием и другими алгоритмами обработки речи в режиме реального времени;

– быстрая адаптация к изменению уровню шума (фона);

– быстрая реакция на начала звуков после паузы;

– восприятие согласных звуков – щелевых (*ф, ш* и т.д.) и аффрикатов (*ц* и *ч*) – или их начальных участков как активной речи, а не фона;

– использование универсального звукового фрагмента для заполнения пауз с возможностью циклической прокрутки или генерация шума, похожего на естественный;

– высокие показатели работы алгоритма VAD при низком показателе SNR (Signal Noise Ratio – отношение сигнал/шум), т.е. в случае, при котором уровень шума может даже превышать уровень сигнала.

В разработанном программном продукте учтены все пункты, кроме последнего. Стабильность работы алгоритма VAD в случае большой зашумленности сигнала требует работы со спектрами или использования других, достаточно трудоемких для вычислительной техники алгоритмов, например, алгоритма на основе подсчета числа переходов через ноль (zero crossing algorithm), алгоритмов предсказания или др. Решение этого вопроса требует дополнительного исследования.

Следует отметить, что время обработки не должно превышать интервал времени между последовательными речевыми сегментами. Другими словами, если каждый сегмент содержит 20 мс речи, то во избежание дополнительных задержек и потерь данных время работы алгоритма на конкретном устройстве не должно превышать 20 мс с учетом кодирования.

Предложенный алгоритм VAD имеет вычислительную сложность менее 300 тыс.оп./с над числами с плавающей точкой, рассчитанную по максимальной ветви программы, что позволяет, наращивая его функциональность, увеличить вычислительную сложность почти на порядок.

CNG практически не требует процессорного времени, поскольку его работа состоит в интерпретации принятого байта как усредненной амплитуды фона и проигрывании (при необходимости циклически) заранее записанного в файл и считанного при инициализации в буфер шумового сэмпла с учетом принятого значения громкости.

Алгоритм VAD основан на работе с адаптивными уровнями фона и активной речи (с фоном). Формулы для расчета соответствующих уровней аналогичны и имеют вид:

$$A_{cur} = (1-\alpha)\cdot AL + \alpha \cdot P_{cur},$$

где AL_{cur}, AL – рассчитанный адаптивный уровень сигнала с учетом текущего и предыдущего отсчетов, соответственно; α ($0 \le \alpha \le 1$) – константа, определяющая скорость реакции на текущий отсчет сигнала; P_{cur} – мощность текущего отсчета сигнала, вычисленная как квадрат его амплитуды. При приходе следующего отсчета AL_{cur} в формулу подставляется как AL.

Таким образом, кратко один цикл работы VAD можно описать следующим образом:

– запоминаем предыдущий AL_1 (адаптивный уровень 1), характеризующий фон;

– считаем AL_1;

– считаем AL_2 (адаптивный уровень 2), характеризующий активность речи;

– сравниваем отношение AL_2 / AL_1 с заданным пороговым значением;

– если идет первая секунда разговора или количество превышений порога больше заданного процента от размера сегмента, то принимается решение о наличии активной речи;

– в другом случае принимается решение о наличии только шума. Тогда в начале каждого периода молчания, а затем через каждую секунду формируется уровень комфортного шума, исходя из значения AL_1.

Для тестирования применялась схема без использования кодирования.

На рис. 3–5 приведены графики изменения исходного сигнала, решения VAD и отношения AL_2 к AL_1 во времени, соответственно.

На графиках отображен один и тот же временной интервал. Подписи к осям *x* и *y* не приводятся, так как в конкретном случае они не несут смысловой нагрузки, а внимание следует уделять изменению графиков.

На рис. 3 приводится пример непрерывной речи. На рис. 4 верхние участки графика являются решениями VAD о наличии периодов молчания. Даже визуально можно определить, насколько механизм VAD позволяет сократить объем передаваемых данных. Рис. 5 демонстрирует принцип, на основании которого VAD принимает решение.

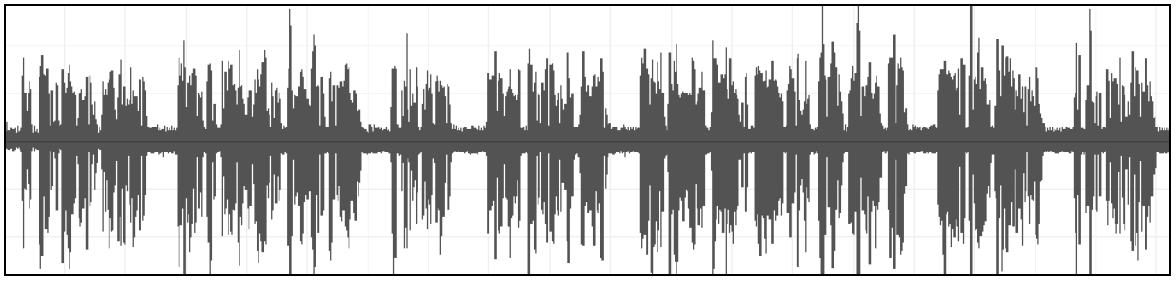


Рис. 3. График изменения исходного сигнала во времени

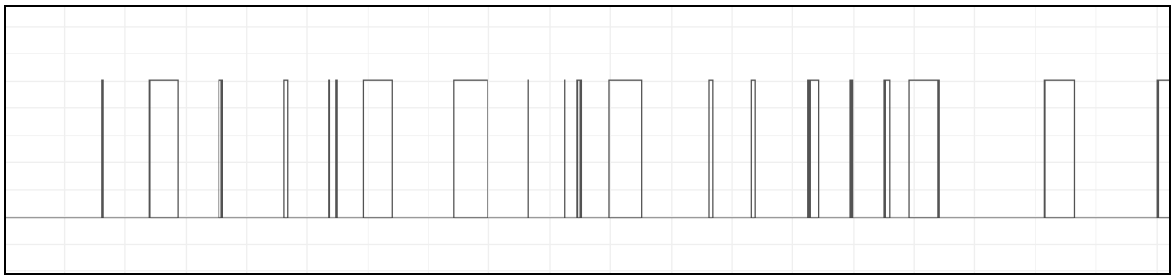


Рис. 4. График изменения решения VAD во времени

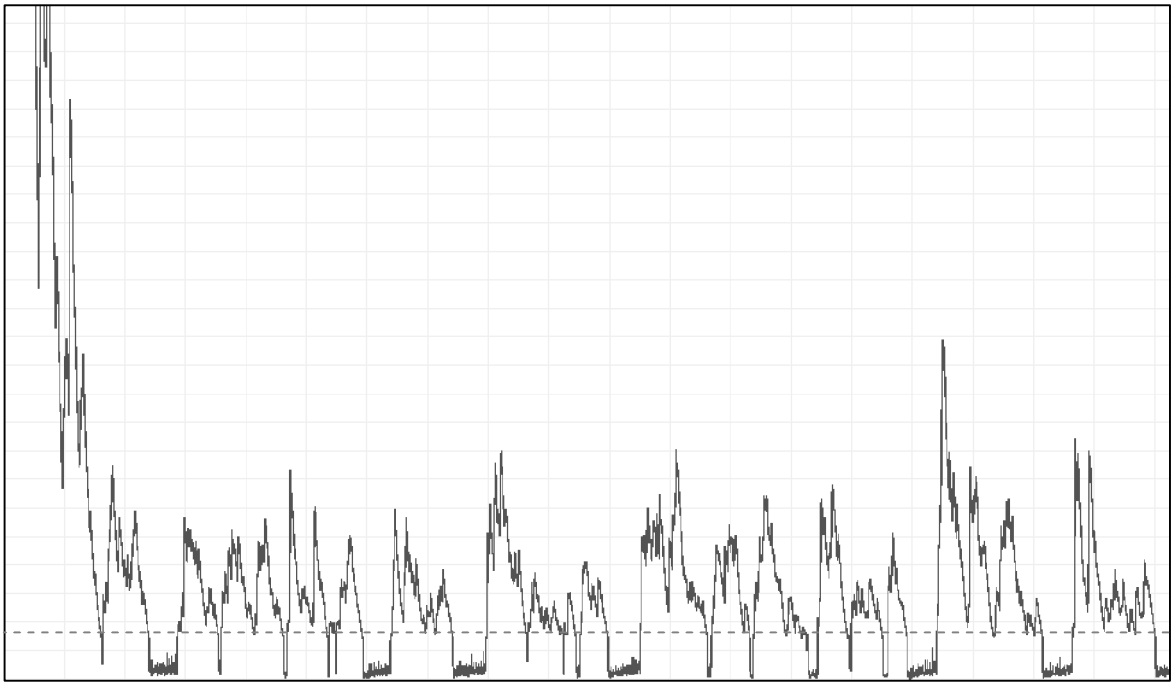


Рис. 5. График изменения отношения AL_2 к AL_1 во времени

В результате выполнены следующие пункты:

– разработка базовых алгоритмов VAD и CNG;

– подбор фонового шума (адаптированный коричневый шум), прокручиваемого циклически (если пауза длится более одной секунды);

– адаптация алгоритма VAD для быстрой речи и согласных звуков, похожих хотя бы начальным участком на шум (с помощью деления блока данных на подблоки и их дифференциальной оценки; особое внимание уделяется последнему подблоку, который может содержать начало звукового сегмента);

– введение дополнительного параметра: допустимого процента превышений порога, при котором блок данных оценивается еще как шум (защита от импульсных помех, увеличение параметра улучшает работу VAD, но может «резать» речь – необходима настройка);

– внесение в программу блоков для обработки тестовых звуковых файлов вне real time, для возможности получения и сравнения графиков:

- адаптивного уровня для шума;
- адаптивного уровня для голоса;
- отношения указанных уровней;
- решения VAD;
- итогового звука;

– интерфейс с возможностью настройки различных параметров шума и выводом результата работы VAD в процентах для real time речи и хранимых файлов (для быстрого сравнения результата работы VAD);

– выбор основного регулируемого параметра для подстройки под различные типы речи (допустимый процент превышений порога, при котором блок данных оценивается еще как шум);

– анализ кода функций VAD и CNG на предмет количества операций с плавающей точкой.

Заключение

В статье рассмотрен механизм прерывистой передачи аудиопотока, описаны методы, положенные в основу разработанного алгоритма определения речевой активности.

Литература

1. Быков С.Ф., Журавлев В.И., Шалимов И.А. Цифровая телефония. // М.: Радио и связь, 2003. С.94–109.

СЖАТИЕ ВИДЕО С ПОМОЩЬЮ ВЫДЕЛЕНИЯ ЛОКАЛЬНЫХ ОБЛАСТЕЙ

И.Н. Курносенков

Научный руководитель – доктор технических наук, профессор А.Ю. Тропченко

В статье излагается суть алгоритма сжатия с помощью выделения локальных областей и рассматривается возможность его применения для сжатия видеопоследовательностей.

Введение

Распространение технологий цифрового вещания и цифрового видео привело к необходимости разработки эффективных методов сжатия видеопоследовательностей. Сжатие видео основано на двух важных принципах: пространственной избыточности, присущей каждому кадру видеоряда, и временной избыточности, т.е. похожести предыдущего кадра на следующий. Таким образом, типичный метод сжатия заключается в кодировании первого кадра с помощью некоторого алгоритма сжатия изображений и последующем кодировании разности первого кадра и последующих [1]. Для кодирования первого кадра предлагается использовать алгоритм сжатия с помощью выделения локальных областей. На последующих кадрах производится поиск областей, выделенных на первом кадре, и вычитание их из изображения. Разностное изображение можно кодировать с помощью алгоритма JPEG либо сохранять среднее значение яркости фона изображения.

Описание алгоритма сжатия

Рассматриваемый алгоритм сжатия относится к фрактальным алгоритмам, которые основываются на повторяемости объектов в реальных изображениях. Фрактальные алгоритмы сжимают изображение путем преобразования доменных областей изображения в ранговые области. Ранговая область представляет собой характерный объект изображения, который повторяется в разных видах в домене. Сжатое изображение представляет собой множество ранговых областей и данные о преобразованиях, с помощью которых из каждой ранговой области можно получить доменные области и собрать из них исходное изображение. Изображение можно рассматривать как набор локальных областей (пятен близкой интенсивности), что может быть записано как:

$$I(X,Y) \approx \sum_{s=1}^{N_1} f_s(x,y), \qquad (1)$$

где $I(X,Y)$ – исходное изображение; $f_s(x,y)$ – функция, аппроксимирующая локальную область D_s с размерами l_{xs}, l_{ys} с центром $\{x_{0s}; y_{0s}\}$; s – порядковый номер области; N_1 – число областей.

Функция f_s, которая задает пятно, должна быть гладкой, чтобы избежать разрывов при аппроксимации больших пятен. Применение аппроксимирующей функции позволяет сохранить информацию о пятне в виде координат центра, амплитуды и размерах фрагмента используемой гармонической функции. К аппроксимирующим функциям предъявляются следующие требования:

- простота аналитического описания аппроксимирующей функции;
- наличие выраженного максимума в центре и монотонность убывания к периферии;
- осесимметричность;
- плавное сопряжение с фоновым изображением;
- простота масштабного преобразования.

К числу аппроксимирующих функций, наиболее полно удовлетворяющих указанным требованиям, могут быть отнесены, в частности, двумерные гармонические функции вида [2]:

$$f_s(x,y)=\begin{cases} C_s[\cos[\omega_s(x-x_{0s})]+1][\cos[\omega_s(y-y_{0s})]+1]; x, y \in D_s \\ 0; x, y \notin D_s \end{cases}. \quad (2)$$

Рассмотрим аппроксимацию выделенных локальных областей фрагментами двумерных гармонических функций вида (1). Такая аппроксимация приводит к погрешности (потерям) при построении изображения из функций, аппроксимирующих пятна, но при этом существенно упрощается задача описания пятен. Для этого нужно определить параметры: C – максимальную амплитуду области; x_0, y_0 – координаты центра области; l_x, l_y – размеры области, однозначно определяющие частоты ω_k.

Если область имеет эллиптическую форму, то необходимо также определить его ориентацию на изображении. В этом случае при аппроксимации объекта необходимо выполнять дополнительное аффинное преобразование аргументов функции f_s, соответствующее ее развороту на угол φ в пространстве изображения.

Для аппроксимации областей используется следующая функция:

$$F=\frac{\cos(\omega x+1)*\cos(\omega y+1)}{4}. \quad (3)$$

После выделения на изображении локальных областей и их аппроксимации значение аппроксимирующей функции вычитается из изображения. Вычитание проводится до тех пор, пока в изображении не останется объектов, амплитуда которых превышает некоторый порог, отделяющий фон от объектов.

В данной работе в качестве объектов сжатия рассматриваются полутоновые цветные изображения. При этом перед самим сжатием изображения преобразуются из цветового пространства RGB в пространство YUV. После преобразования производится прореживание цветовых плоскостей U и V так, как это делается в алгоритме стандарта JPEG (YUV 4:1:1) [6].

Для каждой из цветовых плоскостей общая схема алгоритма для предлагаемого метода будет состоять из следующих блоков.

1. Определение точки максимальной яркости (ТМЯ) на исходном изображении.
2. Проверка условия завершения работы алгоритма: если яркость ТМЯ не превышает уровень яркости фона $I_{\text{background}}$, то завершить процесс сжатия.
3. Определение по алгоритму сегментации всех точек объекта, которому принадлежит ТМЯ.
4. Определение параметров функции f_s для выбранного объекта (C, x_0, y_0, l_x, l_y, φ).
5. Вычитание значения функции f_s из исходного изображения.
6. Переход к п.1.

Сжатие производится для кадров в пространстве YUV. Алгоритм применяется для каждой плоскости, при этом плоскости могут обрабатываться независимо друг от друга.

Таким образом, для каждого кадра исходной последовательности выполняются следующие шаги:

1. Сжатие первого кадра в соответствии с описанным выше алгоритмом.
2. Поиск выделенных областей на следующем кадре с учетом компенсации движения.
3. Вычитание найденных областей из кадра и сохранение их координат.
4. Если найдено менее половины областей, то этот кадр считается началом новой последовательности, переход к п.1.
5. Кодирование полученного разностного изображения.
6. Переход к следующему кадру.

Рассмотрим подробнее стадии алгоритма.

Выделение локальных областей

Для выделения области используется алгоритм пороговой сегментации. Суть алгоритма сводится к тому, что все точки, яркость которых находится в заданных пределах,

считаются принадлежащими одному объекту. В данной работе на выбор точек накладывается еще несколько ограничений, необходимых для того, чтобы выделенные области были выпуклыми. Это требование обусловлено тем, что выпуклые области наиболее точно аппроксимируются выбранными функциями.

На каждой итерации алгоритма сжатия формируется маска области, которая представляет собой двумерный массив отметок, размер которого равен размеру изображения, и выбираются пороги I_{top} и I_{bottom}. Если точка удовлетворяет условиям принадлежности к выделяемой области, то соответствующий элемент массива отметок принимает ненулевое значение.

Сначала производится поиск точки максимальной яркости (ТМЯ) на изображении. Яркость ТМЯ является верхним пределом (I_{top}). Нижний предел определяется из соотношения:

$$I_{bottom} = I_{top} * \xi, \quad (4)$$

где ξ = 0,25 для яркостной компоненты (Y-плоскость) и ξ = 0,125 для цветоразностных компонент (плоскости U и V). Различные значения для разных компонент обусловлены тем, что при выделении областей на плоскостях U и V алгоритм должен обладать большей чувствительностью, чем при выделении областей на плоскости Y. Это связано с тем, что цветоразностные компоненты несут меньше информации, чем Y (что позволяет производить их прореживание без заметного ухудшения качества восстановленного изображения). При этом области на плоскостях U и V не так ярко выражены, как на Y-плоскости (яркость ТМЯ ближе к яркости фона, чем на плоскости Y, т.е. плоскости U и V менее контрастны). Если использовать такое же значение ξ для этих плоскостей, как и для Y, то при выделении областей будет вноситься существенная ошибка.

Затем осуществляется проверка условия завершения алгоритма сжатия. Яркость ТМЯ сравнивается с яркостью фона $I_{background}$. Яркость фона определяется из соотношения:

$$I_{background} = I_{max} * \varepsilon, \quad (5)$$

где I_{max} – максимальное значение яркости в сжимаемом изображении; ε – заданный пользователем коэффициент, изменяющийся в пределах (0,1). Если яркость ТМЯ не превышает $I_{background}$, то процесс сжатия завершается. Коэффициент ε позволяет регулировать точность восстановления сжатого изображения. За счет уменьшения этого коэффициента достигается увеличение точности при уменьшении коэффициента сжатия $K_{сж}$, т.е. при увеличении размера сжатого файла. Определение яркости фона относительно самой яркой точки изображения сообщает алгоритму гибкость при работе с изображениями с разной средней яркостью. ТМЯ также является первой точкой в выделяемой области. Анализ яркости остальных точек изображения происходит в соответствии с алгоритмом заполнения с затравкой. При этом выполняются следующие шаги.

1. Затравочная точка помещается в стек. В данном случае такой точкой будет ТМЯ.
2. Точка извлекается из стека.
3. Проверяются на принадлежность области 8 соседних точек. Точки обходятся по часовой стрелке, как показано на рис. 1.

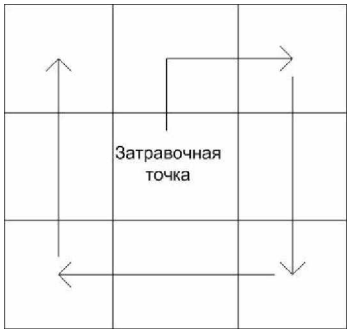


Рис. 1. Схема обхода точек

Если точка принадлежит области, то она помещается в стек. Если точка не удовлетворяет условиям принадлежности к области, то она пропускается.

4. Если стек не пуст, осуществляется переход к п. 2. В противном случае выделение области завершается.

Рассмотрим подробнее условия принадлежности области:

1. Любая из соседних точек должна быть отмечена. Выполнение этого условия гарантируется порядком рассмотрения точек в алгоритме заполнения с затравкой.

2. Яркость точки должна удовлетворять условию $I_{top} \le I \le I_{bottom}$. Если это условие не выполняется, осуществляется переход к следующей точке.

3. Проверяются тройки соседних точек. Если они являются отмеченными, то проверяемая точка отмечается в массиве маски, и осуществляется переход к следующей точке.

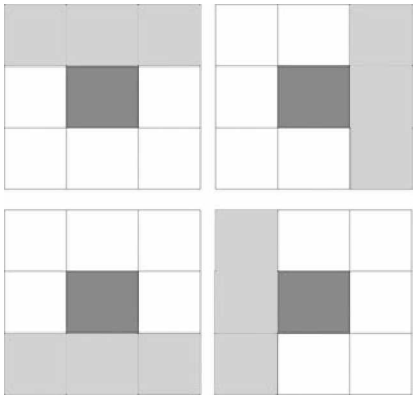


Рис. 2. Проверка троек соседних точек

На рис. 2 темно-серым помечена рассматриваемая точка, а светло-серым – уже отмеченные точки. Если имеет место одна из этих четырех ситуаций, то условие 3 считается выполненным. Данное условие служит для ускорения процесса сегментации, так как позволяет быстро добавлять к области точки, которые не нарушат выпуклость области.

4. Линия, проведенная из любой точки границы области к проверяемой точке, не должна пересекать точки, яркость которых не принадлежит диапазону [I_{bottom}; I_{top}]. При этом данные точки могут быть не отмечены – предполагается, что они будут присоединены к области на последующих итерациях алгоритма выделения области. Это условие служит для обеспечения выпуклости выделяемой области. На рис. 3 белым отмечены точки, которые были отмечены на предыдущих шагах, а черным – точки, которые могут быть отмечены на *k*-ом шаге.

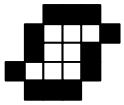


Рис. 3. Добавление новых точек на *k*-ом шаге в выделяемую область

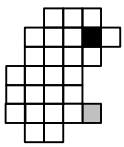


Рис. 4. Исключение точки из области при разметке

На рис. 4 черным помечена точка максимальной яркости, белым – точки, которые были отмечены на предыдущих шагах, и серым – точка, которая не может быть отмечена на текущем (*k*-ом) шаге.

Условия 3 и 4 призваны обеспечить выпуклость выделенной области.

В результате работы алгоритма выделения областей получаем массив, в котором отмечены точки, входящие в выпуклую область. Далее эта область аппроксимируется заданной функцией.

В том случае, если выделенная область состоит всего из одной точки, то она не аппроксимируется функцией. Выделенная точка вычитается из изображения. При восстановлении изображения яркость такой точки полагается равной среднему арифметическому 4 соседних точек.

Определение параметров аппроксимирующей функции

Чтобы аппроксимировать выделенную область с помощью гладкой функции, находятся параметры функции. Определение параметров происходит в следующем порядке.

1. Определяется центр области x_0, y_0.
2. Определяется ориентация области φ.
3. Определяются размеры области l_x, l_y.
4. Определяется амплитуда B.

Рассмотрим подробнее эти шаги.

Для определения центра области x_0, y_0 считаются предварительные размеры (l_{x1},l_{y1}) области без учета ее ориентации и ее границы $x_{\min}$, $y_{\min}$, $x_{\max}$, $y_{\max}$. Координаты центра вычисляются следующим образом:

$$y_0 = y_{\min} + \frac{l_{y1}}{2}, \qquad x_0 = x_{\min} + \frac{l_{x1}}{2}. \tag{6}$$

Затем определяется ориентация области φ.

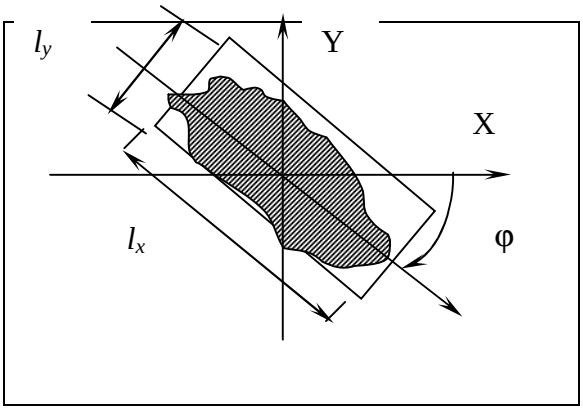


Рис. 5. Определение ориентации локальной области эллиптической формы

Для всех точек области выполняются следующие действия.

Для всех направлений $i = 0,\ldots,7$ выполняется обратное преобразование:

$$x = a_{ci}(x'-x_0) + a_{si}(y'-y_0), \qquad y = -a_{si}(x'-x_0) + a_{ci}(y'-y_0). \tag{7}$$

Для каждого направления для всех точек определяются $x_{\min i}$, $y_{\min i}$, $x_{\max i}$, $y_{\max i}$, т.е. границы прямоугольной области, описывающей объект. Зная количество точек исходного объекта

Cnt, определяем $K_{\varphi i}$ – коэффициент заполнения прямоугольной области для каждого направления:

$$K_{\varphi i} = \frac{Cnt}{(x_{\max i} - x_{\min i} + 1)(y_{\max i} - y_{\min i} + 1)}. \quad (8)$$

Выбирая $max(K_{\varphi i})$, определяем направление (поворот) i, размеры l_x и l_y:

$$\begin{aligned} l_x &= x_{\max i} - x_{\min i} + 1, \\ l_y &= y_{\max i} - y_{\min i} + 1. \end{aligned} \quad (9)$$

Размеры l_x и l_y не должны превышать 255, так как для их хранения используются 2 байта. Если размер выделенной области по любой из координат будет больше 255, то область уменьшается до максимального разрешенного размера.

Новый центр объекта (x_0*, y_0*) для аппроксимирующей функции f_s определяется как

$$\begin{aligned} x_0^* &= \frac{(x_{\min i} - x_{\max i})}{2}, \\ y_0^* &= \frac{(y_{\min i} - y_{\max i})}{2}. \end{aligned} \quad (10)$$

Выполняя прямое аффинное преобразование над (x_0*, y_0*), получаем (x_0, y_0) в пространстве изображения:

$$\begin{aligned} x_0' &= a_{ci} x_0^* - a_{si} y_0^* + x_0, \\ y_0' &= a_{si} x_0^* + a_{ci} y_0^* + y_0. \end{aligned} \quad (11)$$

Значения l_x, l_y, x_0, y_0 округляются до целых. Таким образом, параметры φ, l_x, l_y, x_0, y_0 функции f определены.

Яркость ТМЯ данной области будет иметь амплитуду B.

Определенные таким образом параметры аппроксимирующей функции сохраняются. Значения аппроксимирующей функции вычитаются из исходного изображения.

Поиск выделенных областей на следующем кадре с учетом компенсации движения

Следующий шаг алгоритма заключается в поиске выделенных областей на следующем кадре. Для поиска используются области до их аппроксимации. Поиск происходит в окрестности выделенной области (области поиска). Размеры окрестности определяются следующим образом:

$$\begin{aligned} L_x &= 2 * \Delta x, \\ L_y &= 2 * \Delta y, \end{aligned} \quad (12)$$

где Δx – ширина прямоугольника, в который вписана выделенная область, Δy – высота прямоугольника, в который вписана выделенная область. При этом центром окрестности будет являться точка с координатами (a+Δx/2, b+Δy/2), где a и b – координаты левого верхнего угла прямоугольника, в который вписана область. Для всех возможных положений прямоугольника с выделенной областью внутри области поиска вычисляется разность пикселей, принадлежащих выделенной области, и соответствующих пикселей в области поиска. Положение, при котором средняя разность пикселей (остаточная энергия изображения) минимальна, используется для построения вектора движения области. Пороговое значение остаточной энергии, выше которого области считаются несовпадающими, устанавливается экспериментально.

Вычитание найденных областей из кадра и сохранение их координат

Найденная область вычитается из кадра, при этом сохраняется вектор движения, описывающий перемещение области относительно ее положения на I-кадре, и идентификатор области.

Переход к следующему *I*-кадру

Если найдено менее половины областей, то этот кадр считается началом новой последовательности. Следующий кадр рассматривается как новый *I*-кадр, т.е. сжимается независимо от предыдущих кадров.

Кодирование полученного разностного изображения

Разностное изображение кодируется в соответствии с алгоритмом jpeg и сохраняется наряду с векторами движения и соответствующими идентификаторами областей *I*-кадра.

После этого осуществляется переход к следующему кадру.

Восстановление видеопоследовательности

Для первого *I*-кадра выделенные области хранятся в явном виде. Они используются для восстановления этого кадра и всех последующих. Последующие сжатые кадры представляют собой набор координат и идентификаторов областей *I*-кадра, а также разностное изображение.

Восстановление кадра сводится к вычислению функций $f(x, y)$ на каждой итерации и алгебраическому суммированию. Вычислительные затраты процедуры восстановления минимальны.

После восстановления плоскостей в них производится поиск одиночных точек с нулевой яркостью. Эти точки в подавляющем большинстве случаев являются результатом отбрасывания одноточечных областей на этапе сжатия. Яркость этих точек рассчитывается как среднее арифметическое 4 соседних точек.

Заключение

Для кодирования видео предлагается использовать метод, основанный на выделении локальных областей и параметрическом их описании. Выделение областей производится на первом кадре видеопоследовательности (*I*-кадр), затем производится поиск выделенных областей на последующих кадрах.

Литература

1. Ватолин Д., Ратушняк А., Смирнов М., Юкин В. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео. М.: ДИАЛОГ-МИФИ, 2003. 384 с.
2. Тропченко А.А. Сжатие изображений на основе аппроксимации гладкими функциями. // В сб. Современные технологии. Труды молодых ученых ИТМО. СПб, 2001, с. 110–114.
3. Сэломон Д. Сжатие данных, изображений и звука. М.: Техносфера, 2004. 368 с.
4. Ян Ричардсон. Видеокодирование. H.264 и MPEG-4 – стандарты нового поколения. М.: Техносфера, 2005. 368 с.
5. Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC), 2003.
6. Image and Video Coding – Emerging Standards and Beyond. IEEE Transactions on Circuits and Systems for Video Technology, vol. 8, no. 7, November 1997, pp. 814–837.

ПОСТРОЕНИЕ МОДЕЛИ ДАННЫХ ПРОГРАММЫ ПО ИСХОДНОМУ КОДУ

Г.А. Корнеев

Научный руководитель – доктор технических наук, профессор А.А. Шалыто

В работе предлагается метод построения модели данных программы по ее коду на языке высокого уровня. В результате применения метода все переменные выносятся в модель данных, и строится модификация программы, использующая модель данных и эквивалентная исходной программе.

Введение

В рамках технологии автоматного программирования [1] рассматриваются формальные методы преобразования программ к автоматному виду [2, 3]. При этом для преобразования программы все используемые переменные должны быть вынесены в модель данных, предназначенную для сохранения вычислительного состояния.

В настоящей работе предлагается метод построения модели данных программы по ее исходному коду. При этом входными данными является исходный код программы на языке высокого уровня, а выходными данными – модель данных программы и модифицированный исходный код. В связи с наличием выходных данных двух типов преобразование выполняется в два этапа:

1. *построение модели данных* – по программе строится модель данных, в которую выносятся все используемые в программе переменные;
2. *модификация программы* – программа модифицируется так, чтобы она использовала только переменные модели данных.

В первом разделе приведены требования к преобразуемой программе. Во втором – описывается построение модели данных для итеративных программ, а в третьем – для рекурсивных.

1. Требования к исходной программе

Сформулируем требования к исходной программе.

1. Параметры и возвращаемые значения процедур передаются по значению. Отметим, что это не исключает передачу ссылок или указателей.
2. В идентификаторах не должен использоваться символ подчеркивания (“_”).
3. Используемые выражения не должны иметь побочных эффектов – значения переменных изменяются только оператором присваивания.

Первое ограничение несущественно, так как все современные языки программирования позволяют возвращать из процедур объекты и/или структуры. В языках *Java*, *C* и *C++* все данные передаются по значению. В языке *Pascal* возможен обход этого правила путем применения ключевого слова `var`. Однако такие программы легко преобразуются к требуемому виду за счет применения указателей.

Второе ограничение наложено для того, чтобы символ подчеркивания мог быть использован для образования структурированных идентификаторов. При необходимости, переменные, используемые в программе, можно переименовать произвольным образом.

Третье ограничение исключает применение цепных операторов присваивания, а также операции инкремента (`++`) и декремента (`--`) в языках *Java*, *C* и *C++*. Цепные операторы присваивания легко преобразуются в последовательность присваиваний. В свою очередь, выражения, использующие операции инкремента и декремента, преобразуются в последовательность из нескольких выражений.

2. Построение модели данных по итеративной программе

В соответствии с синтаксисом большинства процедурных языков программирования все переменные, используемые в реализации алгоритма, можно разделить на два класса:

- *глобальные* – определенные на уровне программы и доступные во всех процедурах;
- *локальные* – определенные в рамках процедуры и доступные только в ней;
- *аргументы процедур* – также определенные в рамках одной процедуры и доступные только в ней.

Отметим, что если переменным модели давать те же имена, что и исходным переменным, то возможно дублирование имен переменных модели, что недопустимо. Для решения этой проблемы предлагается использовать правила, изложенные ниже.

Правила выделения глобальных и локальных переменных различны и будут рассматриваться отдельно в разделах 2.1 и 2.2 соответственно.

2.1. Построение модели данных

В случае итеративных программ построение модели разбивается на четыре шага. На каждом из них в модель добавляется набор переменных.

На первом шаге в модель выносятся глобальные переменные. Так как имена всех глобальных переменных различны, то их можно вынести в модель с сохранением имен. Этот шаг может быть формализован следующим образом.

1. Для каждой глобальной переменной в модель добавляется переменная с именем, совпадающим с именем соответствующей глобальной переменной. Тип добавляемой переменной соответствует типу глобальной переменной.

Здесь и далее при формализации действий, выполняемых в рамках шага, им присваиваются номера для последующих ссылок.

На втором шаге выполняется перенос в модель локальных переменных. Работа с локальными переменными более сложна, так как в разных процедурах могут быть определены переменные с одинаковыми именами. Поэтому при построении модели используется следующий прием: если переменная `a` определена в процедуре `calcSum`, то соответствующая ей переменная модели будет иметь имя `calcSum_a`. При этом используется второе требование к исходной программе (раздел 1). Тип добавленной переменной соответствует типу локальной переменной. В общем случае этот шаг может быть формализован следующим образом.

2. Для каждой локальной переменной в модель добавляется переменная с именем вида

   <имя процедуры>_<имя переменной>

Переходя к третьему шагу, отметим, что в случае итеративных программ для аргументов процедур можно использовать ту же схему именования, как и для локальных переменных, что может быть сформулировано следующим образом.

3. Для каждого формального аргумента процедуры в модель добавляется переменная с именем вида

   <имя процедуры>_<имя формального аргумента>

   Тип переменной модели соответствует типу формального аргумента.

В заключение, на четвертом шаге для процедур, возвращающих значения, необходимо добавить в модель переменные для хранения возвращаемых значений.

4. Для каждой процедуры, возвращающей значения, в модель добавляется переменная с именем вида

   <имя процедуры>_

   и типом, соответствующим типу возвращаемого значения.

Предложенная схема образования идентификаторов позволяет не только избежать дублирования имен переменных модели, что требуется синтаксисом языка, но и однозначно восстанавливать по имени переменной модели информацию о том, по какой переменной и из какой процедуры она порождена.

Заметим, что на шагах два и три используется правило, не позволяющее в большинстве языков программирования иметь в одной процедуре локальные переменные и формальные параметры с одинаковыми именами. В языках, разрешающих такие именования, можно имена переменных модели, соответствующих формальным аргументам процедуры, начинать с символа подчеркивания. Таким образом, шаг три примет следующий вид:

3. Для каждого формального аргумента процедуры в модель добавляется переменная с именем вида

   <имя процедуры>_<имя формального аргумента>

   Тип переменной модели соответствует типу формального аргумента.

2.2. Модификация программы

Модифицируем программу так, чтобы она использовала только переменные модели данных, построенной, как указано в предыдущем разделе.

В дальнейшем предполагается, что экземпляр модели доступен всем процедурам как переменная с именем `data`.

Так как программа итеративна, то для каждой локальной переменной, формального параметра процедуры и возвращаемого значения можно выделить статический участок памяти. Фактически это и производится при построении модели данных (шаги 1–4 предыдущего раздела).

Для модификации программы совершим следующие действия.

1. Удалим объявления глобальных переменных.
2. Добавим объявление глобальной переменной `data`.
3. Так как имена глобальных переменных при вынесении в модель не изменились, то для использования глобальных переменных требуется добавить к ним префикс “`data.`” .
4. Если некоторые локальные переменные инициализируются одновременно с их объявлением, то разобьем каждое объявление на две части – объявление и инициализация.
5. Удалим объявления локальных переменных.
6. Ко всем обращениям к локальным переменным добавим префикс вида

   `data.`*<имя процедуры>*`_`

В результате выполнения этих шагов все обращения к локальным переменным будут заменены обращениями к переменным модели.

Последующие шаги преобразуют вызовы процедур.

7. Операторы, вызывающие одну и ту же процедуру более одного раза, разбиваются так, чтобы ни одна процедура не вызывалась дважды в одном операторе.
8. Вызов процедуры в выражении разбивается на две части:
   - оператор вызова процедуры (с сохранением возвращаемого значения в соответствующей переменной модели);
   - исходное выражение, с заменой вызова процедуры на обращение к переменной вида `data.`*<имя процедуры>*`_`.
9. Вызовы процедур преобразуются следующим образом. Пусть вызывается процедура с *N* формальными аргументами. Обозначим имена этих аргументов следующим образом:

   аргумент1, *аргумент2*, ... *аргументN*.

Реальные аргументы обозначим как

выражение1, *выражение2*, ..., *выражениеN*

соответственно. Тогда оператор вызова процедуры заменяется на N операторов вида

`data.`*<имя процедуры>*`_`*<аргументM>* `=` *<выражениеM>*`;`

где M пробегает значения от 1 до N и вызов процедуры без аргументов.

10. Обращения к формальным аргументам процедуры заменим обращением к соответствующим переменным модели.
11. Заменим каждый оператор возврата значения

    `return` *выражение*`;`

    на присваивание значения выражения переменной

    `data.`*<имя процедуры>*`_`

    и простой оператор возврата (без возвращаемого значения).
12. Заголовки всех процедур изменяются так, чтобы процедуры не имели параметров и возвращаемых значений.

Отметим, что программа является работоспособной не только после выполнения всех шагов, но и после выполнения первых двух, трех, четырех, шести или семи шагов, что может быть использовано для проверки правильности производимых преобразований.

2.3. Упрощенная запись (@-нотация)

Для ссылок на переменные, вынесенные в модель данных, удобно применять *@-нотацию*: запись вида `@a` будет означать обращение к переменный a модели данных.

Если переменная a является глобальной, то запись `@a` эквивалентна записи

```
data.a
```

Если же a – локальная переменная, то `@a` будет означать ссылку на соответствующую локальную переменную, вынесенную в модель данных. Например, если в процедуре `calcSum` была определена переменная a, то в ней запись `@a` будет эквивалентна записи

```
data.calcSum_a
```

Применение @-нотации позволяет приблизить запись программ с вынесенной моделью данных к исходной записи.

В дальнейшем во всех примерах для обращения к переменным модели применяется @-нотация.

2.4. Пример построения модели данных и модификации программы

Рассмотрим изложенный метод на примере программы, вычисляющей максимальный из локальных минимумов в массиве натуральных целых чисел (для примеров здесь и в дальнейшем используется язык *Java*).

Исходная программа имеет следующий вид:

```
int[] a; // Массив, в котором осуществляется поиск

/** Подсчитывает указанный максимум */
void calc() {
  int m = 0;
  for (int i = 1; i < a.length - 1; i++) {
    if (isMin(a[i-1], a[i], a[i+1]) && a[i] > m) m = a[i];
  }
}

/** Возвращает, является ли b минимумом аргументов */
boolean isMin(int a, int b, int c) {
  int m = a > b ? a : b;
  return b == (m > c ? m : c);
}
```

Построим модель данных, как указано в разделе 2.1:

```
/** Модель данных */
Class Model {
  int[] a; // Глобальная переменная
  int calc_m, calc_i, isMin_m; // Локальные переменные
  int isMin_a, isMin_b, isMin_c; // Аргументы isMin
  boolean isMin_; // Возвращаемое значение isMin
}
```

Перейдем к модификации программы. После выполнения первых трех шагов из раздела 2.2 процедура `calc` приобретает следующий вид:

```
Model d; // Объявление переменной модели

void calc() {
  int m = 0;
  for (int i = 1; i < a.length - 1; i++) {
    if (isMin(@a[i-1], @a[i], @a[i+1]) && @a[i] > m)
        m = @a[i];
  }
}
```

А процедура `isMin` осталась без изменений.

Выполнение шагов 4–6 изменяет обе процедуры:

```
void calc() {
  @m = 0;
  for (@i = 1; @i < @a.length-1; @i++) {
    if (isMin(@a[@i-1], @a[@i], @a[@i+1]) && @a[@i] > @m) @m =
        @a[@i];
  }
}

boolean isMin(int a, int b, int c) {
  @m = a > b ? a : b;
  return b == (@m > c ? @m : c);
}
```

Так как в рассматриваемой программе осуществляется единственный вызов процедуры, то на седьмом шаге программа не модифицируется.

После выполнения шагов восемь и девять имеем:

```
void calc() {
  @m = 0;
  for (@i = 1; @i < @a.length-1; @i++) {
    @isMin_a = @a[@i - 1];
    @isMin_b = @a[@i];
    @isMin_c = @a[@i + 1];
    isMin();
    if (@isMin_ && @a[@i] > @m)
        @m = @a[@i];
  }
}

boolean isMin(int a, int b, int c) {
  @m = a > b ? a : b;
  return b == (@m > c ? @m : c);
}
```

Модификация завершается применением шагов 10–12 к процедуре `isMin`. Итоговая программа выглядит следующим образом:

```
class Model {
  int[] a; // Глобальная переменная
  int calc_m, calc_i, isMin_m; // Локальные переменные
  int isMin_a, isMin_b, isMin_c; // Аргументы isMin
  boolean isMin_; // Возвращаемое значение isMin
}
```

```
Model d; // Объявление переменной модели

void calc() {
  @m = 0;
  for (@i = 1; @i < @a.length-1; @i++) {
    @isMin_a = @a[@i - 1];
    @isMin_b = @a[@i];
    @isMin_c = @a[@i + 1];
    isMin();
    if (@isMin_ && @a[@i] > @m)
         @m = @a[@i];
  }
}

void isMin() {
  @m = @a > @b ? @a : @b;
  @isMin_ = @b == (@m > @c ? @m : @c);
}
```

Заметим, что при выделении модели объем исходного кода несколько увеличился, но при автоматизированной обработке это несущественно.

3. Построение модели данных по рекурсивной программе

В отличие от итеративных программ, при рекурсивном вызове процедур для каждой локальной переменной выделяется новая область памяти, в то время как в модели ей соответствует только одна переменная. Таким образом, возникает проблема, связанная с тем, что при рекурсивном вызове могут изменяться локальные переменные другого экземпляра рекурсивной процедуры. Для учета этой особенности нужно несколько изменить правила, изложенные в разделе 2.1.

3.1. Построение модели данных

Если в программе используется только косвенная рекурсия, то можно непосредственно применять правила именования, изложенные в разделе 2.1.

При использовании непосредственной рекурсии требуется ввести дополнительные переменные модели:

5. для каждого формального аргумента процедуры в модель дополнительно к переменным, добавленным на шаге 3, добавляется переменная с именем вида

   <имя процедуры>_<имя формального аргумента>_

   Тип добавленной переменной соответствует типу формального аргумента.

Добавленные таким образом переменные будут использоваться как временные хранилища значений реальных аргументов вызываемой процедуры.

3.2. Модификация программы

Для хранения локальных переменных и реальных аргументов процедур требуется не только модель, но и дополнительная память. Для сохранения данных будет использован стек, как и при обычном выполнении.

Для этого введем оператор *квази-присваивания* (в программах он будет обозначаться “@=”) который не только изменяет значение своего левого аргумента, но и сохраняет замещенное значение в стеке. Нам также понадобится оператор извлечения значений, сохраненных оператором *квази-присваивания*, обозначаемый “?=”. Эквиваленты этих операторов легко реализуются на любом языке программирования.

При модификации программ, использующих рекурсию, шаги 1–8 и 10–12 (раздел 2.2) остаются без изменений, а шаг 9 заменяется в зависимости от типа рекурсивного вызова.

Если вызов не является непосредственной рекурсией, то девятый шаг выглядит следующим образом (для сокращения записи вместо *<имя процедуры>_<аргументM>* будем писать *имяM*):

- Вызовы процедур преобразуются следующим образом. Пусть вызывается процедура с *N* формальными аргументами. Обозначим имена этих аргументов следующим образом:

  аргумент1, *аргумент2*, ... *аргументN*.

  Реальные аргументы обозначим как

  выражение1, *выражение2*, ..., *выражениеN*

  соответственно. Тогда оператор вызова процедуры заменяется на:

  - *N* операторов квази-присваивания:

    `data.`*<имя процедуры>_<аргументM>* `@=` *<выражениеM>*`;`

    где *M* пробегает значения от 1 до *N*;
  - вызов процедуры без аргументов.
  - *N* операторов восстановления значений, сохраненных квази-присваиваниями:

    `data.`*<имяM>* `?=;`

    где M пробегает значения от 1 до *N* в обратном порядке.

Если же вызов является непосредственный рекурсией, то шаг 9 еще более усложняется:

9. Вызовы процедур преобразуются следующим образом. Пусть вызывается процедура с *N* формальными аргументами. Обозначим имена этих аргументов следующим образом:

   аргумент1, *аргумент2*, ... *аргументN*.

   Реальные аргументы обозначим как

   выражение1, *выражение2*, ..., *выражениеN*

   соответственно. Тогда оператор вызова процедуры заменяется на:

   - *N* операторов присваивания вида

     `data.`*<имяM>*`_ =` *<выражениеM>*`;`

     где *M* пробегает значения от 1 до *N*.
   - *N* операторов квази-присваивания, копирующие значения реальных аргументов:

     `data.`*<имяM>* `@= data.`*<имяM>*`_;`

     где *M* пробегает значения от 1 до *N*.
   - Вызов процедуры без аргументов.
   - *N* операторов восстановления значений сохраненных квази-присваиваниями:

     `data.`*<имяM>* `?=;`

     где *M* пробегает значения от 1 до *N* в обратном порядке.

Указанные модификации позволяют формально выделять модель и преобразовывать программы использующие рекурсию.

3.3. Пример выделения модели и модификации программы

Рассмотрим изложенные модификации на примере программы, вычисляющей факториалы первых *n* натуральных чисел:

```
/** Количество чисел, для которых требуется вычислить
    факториал. */
int n;
```

```
/** Основная процедура. */
void calc() {
  for (int i = 1; i <= n; i++) {
    // Вывод значения факториала
    System.out.println(fact(i));
  }
}

/** Вычисляет факториал числа a. */
int fact(int a) {
  if (a == 0) {
    return 1;
  } else {
    return a * fact(a - 1);
  }
}
```

После выполнения шагов 1–8 имеем:

```
/** Модель данных. */
class Model {
  int n; // Глобальная переменная
  int calc_i; // Локальная переменная процедуры calc
  int fact_; // Возвращаемое значение процедуры calc
  // Переменная для формального параметра a процедуры fact
  int fact_a, fact_a_;
}
Model d;

void calc() {
  for (@i = 1; @i <= @n; @i++) {
    fact(@i);
    System.out.println(@fact_);
  }
}

int fact(int a) {
  if (@a == 0) {
    @fact_ = 1;
  } else {
    fact(@a - 1);
    @fact_ = @a * @fact_;
  }
}
```

После выполнения остальных шагов процедуры изменяется описание и вызовы процедуры `fact`:

```
void calc() {
  for (@i = 1; @i <= @n; @i++) {
    @fact_a @= @i;
    fact();
    @fact_a ?=;
    System.out.println(@fact_);
  }
}

void fact() {
  if (@a == 0) {
    @fact_ = 1;
  } else {
    @a_ = @a - 1;
    @a @= @a_;
    fact();
    @a ?=;
```

```
            @fact_ = @a * @fact_;
        }
    }
```

Заметим, что вызов процедуры `fact` из `calc` не является прямой рекурсией, а вызова `fact` из себя самой ею является. Поэтому код для первого из этих вызовов не использует переменную модели `@a_`.

Заключение

Предложенный метод позволяет производить выделение модели данных путем формального выполнения описанных шагов, что позволяет строить на его основе инструменты, осуществляющие выделение данных программы по ее исходному коду и модификацию исходной программы с целью использования построенной модели данных.

Литература

1. Шалыто А.А. SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб: Наука, 1998.
2. Туккель Н.И., Шалыто А.А., Шамгунов Н.Н. Реализация рекурсивных алгоритмов на основе автоматного подхода. // Телекоммуникации и информатизация образования. 2002. № 5.
3. Корнеев Г.А., Шалыто А.А. Преобразование программ в систему взаимодействующих конечных автоматов. // Труды Второй всероссийской научной конференции «Методы и средства обработки информации». М.: МГУ.

2 ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ

ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ СУБЪЕКТОВ РОССИЙСКОЙ ФЕДЕРАЦИИ КАК СОСТАВНАЯ ЧАСТЬ НАЦИОНАЛЬНОЙ БЕЗОПАСНОСТИ РОССИИ

Е.А. Проценко
Научный руководитель – доктор технических наук, профессор Л.Г. Осовецкий

В статье предпринята попытка описать состояние информационной безопасности субъектов Российской Федерации.

Так как в России основным субъектом обеспечения безопасности является государство, которое формирует механизмы реализации государственной политики в сфере информации, информатизации и защиты информации через органы законодательной, исполнительной и судебной власти, то задача правового обеспечения информационной безопасности (ИБ) в форме методов правового, организационно-технического и социально-экономического регулирования, а также политической деятельности государства, является приоритетной. Юридической базой для проведения работ в области обеспечения безопасности информации, в том числе и безопасности информации субъектов Российской Федерации, являются нижеследующие документы.

1. Концепция национальной безопасности [1] *(далее – Концепция)* определяет необходимость сохранности информационных ресурсов, а также отмечает процесс нарастания угроз национальной безопасности РФ в информационной сфере путем получения несанкционированного доступа *(далее – НСД)* к информационным ресурсам и нарушения нормального функционирования информационных и телекоммуникационных систем *(далее – ИТКС)*. Решение проблемы сохранности информационных ресурсов, а также снижение процесса нарастания угроз национальной безопасности РФ в информационной сфере при развитии многоуровневой системы обеспечения национальной безопасности РФ *(далее – СОНБ РФ)* в соответствии с законодательством, с привлечением уполномоченных в этой области органов, позволит повысить информационную безопасность РФ *(далее – ИБ РФ)*.

2. Доктрина информационной безопасности [2] *(далее – Доктрина)*, развивая Концепцию применительно к информационной сфере путем формирования официальных взглядов на государственную политику в области обеспечения ИБ РФ, в число приоритетных ставит задачу обеспечения ИБ РФ путем:

- совершенствования системы обеспечения ИБ РФ в определенных сферах общественной жизни на федеральном, межрегиональном, региональном и объектовом уровнях;
- информационного обеспечения государственной политики РФ;
- обеспечения безопасности ИТКС и систем, как уже развернутых, так и создаваемых на территории России;
- законодательного разграничения полномочий в области обеспечения ИБ РФ между федеральными органами государственной власти *(далее – ОГВ)* и органами государственной власти субъектов РФ *(далее – ОГВ субъектов РФ)*;
- совершенствования нормативной правовой базы обеспечения ИБ РФ;
- координации деятельности федеральных ОГВ, ОГВ субъектов РФ, предприятий, учреждений и организаций в области обеспечения ИБ РФ независимо от их формы собственности;

- развития и совершенствования государственной системы защиты информации и системы защиты государственной тайны, в рамках правовых методов обеспечения ИБ РФ;
- внесения изменений и дополнений в законодательство РФ, регулирующее отношения в области обеспечения ИБ, в целях создания и совершенствования многоуровневой системы обеспечения ИБ РФ, устранения внутренних противоречий в федеральном законодательстве, противоречий между федеральными законодательными актами и законодательными актами субъектов РФ;
- создания правовой базы для формирования в РФ региональных структур обеспечения ИБ в рамках организационно-технических методов обеспечения ИБ РФ;
- усиления правоприменительной деятельности федеральных органов исполнительной власти *(далее – ФО ИВ РФ)*, органов исполнительной власти субъектов РФ *(далее – ОИВ субъектов РФ)*;
- разработки критериев и методов оценки эффективности (достаточности) систем и средств обеспечения ИБ РФ;
- координации деятельности федеральных ОГВ, ОГВ субъектов РФ по формированию и реализации единой государственной политики в области обеспечения ИБ РФ;
- разработки нормативной правовой базы, регулирующей отношения в информационной сфере.

3. Приоритетные проблемы научных исследований в области ИБ РФ [3] и Основы государственной политики в области обеспечения ИБ субъектов РФ [4], базируясь на Концепции и Доктрине, отмечают следующее.

3.1. Интересы общества, реализуемые в информационной сфере субъекта РФ, заключаются в следующем:

- государственная политика в области обеспечения ИБ субъектов РФ является частью государственной политики в области обеспечения ИБ РФ. Она разрабатывается и выполняется, прежде всего, ОГВ субъектов РФ во взаимодействии и в координации с деятельностью ФО ИВ РФ в области обеспечения ИБ страны;
- выполнение ОГВ субъекта РФ возложенных на них функций в области обеспечения ИБ;
- обеспечение безопасности информационных ресурсов ограниченного доступа, использование и распоряжение которыми делегировано федеральными ОГВ субъекту РФ, а также созданных этим субъектом.

3.2. Основным направлением государственной политики в области обеспечения ИБ субъектов РФ является совершенствование законодательства РФ, регулирующего отношения в области взаимодействия ФО ИВ РФ и ОИВ субъектов РФ в целях обеспечения их ИБ, а также в области формирования и использования федеральных информационных ресурсов, информационных систем, информационных ресурсов и информационных систем, находящихся в совместном ведении, и информационных ресурсов субъектов РФ путем уточнения правовых норм, регулирующих отношения:

- по согласованию деятельности органов исполнительной власти субъектов РФ и ФО ИВ РФ по созданию, обеспечению устойчивости функционирования и безопасности ИТКС, сетей связи и информации, включая информационные ресурсы;
- по использованию федеральных информационных систем и ресурсов, информационных систем и ресурсов совместного ведения ОГВ субъектов РФ, а также информационных систем и ресурсов субъектов РФ федеральными ОГВ.

3.3. Первоочередными мероприятиями по реализации государственной политики в области обеспечения ИБ субъектов РФ являются:

- разработка проектов нормативных правовых актов, закрепляющих механизмы правового регулирования отношений в области взаимодействия и координации деятельности ФО ИВ и органов исполнительной власти субъектов РФ по обеспечению

ИБ субъектов РФ и ИБ РФ в целом, а также в области формирования и использования федеральных информационных ресурсов и информационных систем, информационных ресурсов и информационных систем, находящихся в совместном ведении, информационных ресурсов и информационных систем субъектов РФ;

- создание методического обеспечения оценки ИБ субъектов РФ, и ее влияния на информационную безопасность РФ в целом.

На основе положений рассмотренных выше документов функции координации и контроля в области защиты информации на уровне Северо-Западного федерального округа *(далее – СЗФО)* возложены на субъектов, которые представлены на рис. 1.

Рис. 1. Структура координации и контроля на федеральном уровне

Создание в 2001 г. Координационного совета при полномочном представителе Президента РФ в СЗФО, правопреемником которого стал Межведомственный совет по защите информации при полномочном представителе Президента РФ в СЗФО *(далее – Совет)*, послужило основой формирования многоуровневой системы защиты информации в Северо-Западном регионе на базе одобренного этим же Советом ряда рамочных документов в области ЗИ, совокупность которых позволяет обеспечить требуемый уровень безопасности в информационной сфере и адекватно реагировать на внутренние и внешние угрозы информации. На общесистемном уровне – это:

- Закон субъекта РФ «Об информационных ресурсах» или «О защите информации»;
- концепция ЗИ в субъекте РФ;
- Положение о системе ЗИ в субъекте РФ;
- Положение о порядке организации и проведения работ по ЗИ конфиденциальной информации;
- Положение о Комиссии по ЗИ в субъекте РФ;
- План работы Комиссии по ЗИ;
- Перечень сведений конфиденциального характера;
- Положение о порядке обращения со служебной информацией в органах исполнительной власти субъекта РФ;
- Положение по контролю состояния ЗИ в субъекте РФ.

На региональном уровне – это:

- документы по ЗИ муниципальных образований в субъектах РФ;
- документы по ЗИ предприятий (учреждений, организаций), входящих в состав РСЗИ;
- документы по ЗИ на конкретных объектах защиты, входящих в состав предприятий (учреждений, организаций).

Отсутствие данных документов не позволит своевременно определить политику создания, развития и эффективного функционирования системы ЗИ субъекта РФ.

Основными факторами создания Совета послужили:

1. недостаточность и в большинстве случаев – отсутствие нормативной правовой базы по формированию многоуровневой системы ЗИ в субъектах РФ;

2. возрастание зависимости результатов деятельности ОГВ и местного самоуправления *(далее – ОМСУ)*, предприятий, учреждений и организаций от достоверности используемой ими информации, своевременности ее получения, надежности принятых мер по сохранению в тайне информации ограниченного доступа;

3. необходимость учета и защиты информационных ресурсов от НСД при их создании, приобретении, накапливании субъектами РФ, предприятиями, организациями, учреждениями и гражданами;

4. разнородность классификации одной и той же информации при ее обработке в различных государственных и негосударственных учреждениях;

5. использование органами власти и управления региональных и глобальных информационных систем, в которых накапливаются, обрабатываются и передаются огромные массивы информации, подлежащей защите, что создает предпосылки для НСД к ней;

6. использование третьими лицами средств НСД к информации; постоянный рост числа преступлений в сфере компьютерной информации и в связи с этим – рост расходов на обеспечение безопасности информации, а также невозможность своевременно и достоверно прогнозировать и выявлять эти угрозы, оценивать их опасность, принимать адекватные меры по устранению;

7. отсутствие качественно подготовленных специалистов, а также нормативной методической литературы и постоянно возрастающей стоимости средств и услуг по ЗИ, что делает невозможным обеспечения эффективной ЗИ в большинстве предприятий, организаций и учреждений;

8. невозможность достоверно предсказать отрицательные экономические, социальные и экологические последствия в случае возникновения критических ситуаций, связанных с нарушениями безопасности информации в кредитно-финансовых учреждениях, системах управления, связи и транспорта, а также на экологически опасных производствах и в сфере энергетики.

Совет стал высшим органом в СЗФО по вопросам защиты информации, в структуру которого входят уже упомянутые выше органы обеспечения, координации и контроля в области ЗИ. Основной задачей Совета является реализация конституционной обязанности государства по обеспечению национальной безопасности РФ, а также создание благоприятных условий для выработки и реализации комплекса мер по формированию и развитию эффективной системы защиты информации как составляющей системы информационной безопасности, охватывающей все уровни и ветви органов государственной власти СЗФО.

В 2003 г. Президент РФ определил, что 2004 г. должен стать годом приоритетов в формировании законодательной базы субъектов РФ, соответствующей федеральному законодательству. Данная позиция Президента небезосновательна, так как необходимость принятия субъектами РФ, в пределах своей компетенции, нормативных правовых актов, регулирующих отношения в области ЗИ обусловлена положениями:

- Конституции РФ;
- Закона РФ «О государственной тайне»;
- Федеральных законов «Об информации, информатизации и защите информации», «Об общих принципах организации местного самоуправления в Российской Федерации» и др.

На начало 2006 г. в СЗФО можно отметить положительную динамику в разработке и принятии исполнительными органами власти субъектов РФ нормативных

правовых актов по вопросам регулирования в области ЗИ: действуют 58 нормативных правовых актов (в Республики Карелия – 6, в Республики Коми – 4, в Архангельской области – 12, в Вологодской области – 2, в Калининградской области – 4, в Ленинградской области – 1, в Мурманской области – 3, в Новгородской области – 3, в Псковской области – 17, в Санкт-Петербурге – 6), но следует отметить, что органы власти попрежнему являются слабым звеном в системе защиты информации СЗФО. Основной причиной такого положения дел является формальное отношение руководства субъектов РФ к особой важности проблемы обеспечения ИБ.

Решение проблемы формирования нормативной правовой базы субъекта РФ в области ЗИ позволит определить:

- правовой статус работ по ЗИ с учетом требований федерального законодательства;
- порядок управления и распоряжения информационными ресурсами как собственностью субъекта РФ;
- организационно-правовые, организационно-технические и экономические основы, регулирующие взаимоотношения субъектов права в области ЗИ субъекта РФ;
- порядок финансирования работ по формированию, использованию и защите информационных ресурсов;
- ответственность должностных лиц и порядок их взаимодействия при совершенствовании многоуровневой системы ЗИ, в том числе в процессе реформирования органов местного самоуправления.

Кроме того, это позволит повысить эффективность защиты информации, создать систему, соответствующую задачам обеспечения национальной безопасности, безопасности и устойчивого развития регионов, адекватно реагирующую на угрозы безопасности информации в социально-экономической, административно-управленческой, правоохранительной и других сферах деятельности субъектов РФ, находящихся в пределах СЗФО.

Использование органами власти (в том числе и субъектов РФ) в своей повседневной деятельности различных средств информатизации приводит к наращиванию массивов информации, а также оказывает существенное влияние на технологические процессы обработки и защиты данной информации. Применение этих средств значительно опережает процессы совершенствования и развития соответствующей нормативной и методической базы, поэтому задача модифицирования федерального законодательства и создание четко выстроенной и согласованной правовой базы субъектов РФ в области ЗИ с учетом складывающегося уровня информатизации является более чем актуальной.

Литература

1. Концепция национальной безопасности Российской Федерации (утв. Указом Президента РФ от 17 декабря 1997 г. № 1300) (в ред. Указа Президента РФ от 10 января 2000 г. № 24).
2. Доктрина информационной безопасности Российской Федерации от 9 сентября 2000 г. № ПР-1895.
3. Приоритетные проблемы научных исследований в области информационной безопасности Российской Федерации от 07.01.2003. Одобрено секцией по информационной безопасности научного совета при Совете Безопасности Российской Федерации (протокол от 28 марта 2001 г. № 1).
4. Основы государственной политики в области обеспечения информационной безопасности субъектов Российской Федерации. Одобрено на заседании Межведомственной комиссии Совета Безопасности Российской Федерации по информационной безопасности, протокол от 27 марта 2003 г. № 1.3.

ИСПЫТАНИЕ ПРОГРАММНОГО ПРОДУКТА НА СООТВЕТСТВИЕ ОЦЕНОЧНОМУ УРОВНЮ ДОВЕРИЯ

Д.А. Пикулькин, Ю.В. Проскурин, М.А. Фокин
Научный руководитель – доктор технических наук, профессор Л.Г. Осовецкий

Статья посвящена рассмотрению программы для оценки программного продукта на соответствие оценочному уровню доверия по системе Руководящего документа по безопасности информационных технологий.

Введение

В наше время наблюдается стремительное развитие информационных технологий. Все большее количество ценной информации хранится и передается в цифровом формате, и все большее число злоумышленников переквалифицируются на кражу информации с электронных носителей. Для этого было придумано огромнейшее количество способов, начиная с использования ошибок в программном обеспечении и заканчивая программными закладками. Много скандалов уже было по поводу универсальных ключей в средствах шифрования, которые позволяли получить доступ к любой зашифрованной информации, или по поводу клавиатурных шпионов, встроенных в одну из версий популярной операционной системы. Все это обусловливает необходимость проверки программного обеспечения перед тем, как устанавливать его туда, где идет обработка конфиденциальной информации.

Основная часть

Государственная техническая комиссия России разработала Руководящий документ по безопасности информационных технологий (РД БИТ) и критерии оценки безопасности информационных технологий. РД БИТ содержит систематизированный каталог требований к безопасности информационных технологий (ИТ), порядок и методические рекомендации по его использованию при задании требований, разработке, оценке и сертификации продуктов и систем ИТ по требованию безопасности информации.

Документ предназначен также для органов сертификации и испытательных лабораторий, аккредитованных в системе сертификации средств защиты информации (СЗИ) по требованиям безопасности информации. Основной целью РД является повышение доверия к безопасности продуктов и систем информационных технологий. Положения РД направлены на создание продуктов и систем информационных технологий с уровнем безопасности, адекватным имеющимся по отношению к ним угрозам и проводимой политике безопасности с учетом условий применения, что должно обеспечить оптимизацию продуктов и систем ИТ по критерию «эффективность – стоимость».

Сертификация программного обеспечения происходит следующим образом. Эксперт в соответствии с РД БИТ проводит проверку программного продукта на соответствие определенным параметрам. Основные принципы общих критериев (ОК) состоят в том, что следует четко сформулировать угрозы безопасности и положения политики безопасности организации, а достаточность предложенных мер безопасности должна быть продемонстрирована. Более того, следует предпринять меры по уменьшению вероятности наличия уязвимостей, возможности их проявления (т.е. преднамеренного использования или непреднамеренной активизации), а также степени ущерба, который может явиться следствием проявления уязвимостей. Дополнительно следует предпринять меры для облегчения последующей идентификации уязвимостей, а также по их устранению, ослаблению и/или оповещению об их использовании или активизации. В зависимости от возможности детализации данной проверки выставляется оценка, т.е. определяется уровень доверия к программному продукту.

Предлагаемая программа призвана упростить процесс оценки программного продукта. Она позволяет осуществлять быструю навигацию по классам РД БИТ, быстрое заполнение таблицы для расчета оценочного уровня доверия (ОУД) и получение количественной оценки рейтинга уязвимости всей системы. Оценка уровней доверия происходит по 26 критериям, которые отражают:

- области охвата, т.е. увеличение рассматриваемой части продукта или системы ИТ;
- глубину, т.е. детализацию рассматриваемых проектных материалов и реализации;
- строгость, т.е. применение более структурированного и формального подхода.

Предлагаемая программа упрощает действия оценщика. Она позволяет на основе таблицы (обзор оценочных уровней доверия) присвоить программному продукту один из семи уровней доверия (см. табл.).

Класс доверия	**Семейство доверия**	**Компоненты доверия из оценочного уровня доверия**						
		ОУД1	**ОУД2**	**ОУД3**	**ОУД4**	**ОУД5**	**ОУД6**	**ОУД7**
Управление конфигурацией	ACM_AUT				**1**	1	2	2
	ACM_CAP	**1**	**2**	**3**	**4**	4	**5**	5
	ACM_SCP			**1**	**2**	**3**	3	3
Поставка и эксплуатация	ADO_DEL		1	1	**2**	2	2	3
	ADO_IGS	**1**	1	1	1	1	1	1
Разработка	ADV_FSP	**1**	1	1	**2**	**3**	3	**4**
	ADV_HLD		**1**	**2**	2	**3**	**4**	**5**
	ADV_IMP				**1**	**2**	**3**	3
	ADV_INT					**1**	**2**	**3**
	ADV_LLD				**1**	1	**2**	2
	ADV_RCR	**1**	1	1	1	**2**	2	**3**
	ADV_SPM				**1**	**3**	3	3
Руководства	AGD_ADM	**1**	1	1	1	1	1	1
	AGD_USR	**1**	1	1	1	1	1	1
Поддержка жизненного цикла	ALC_DVS			**1**	1	1	**2**	2
	ALC_FLR							
	ALC_LCD				**1**	**2**	2	**3**
	ALC_TAT				**1**	**2**	**3**	3
Тестирование	ATE_COV		**1**	**2**	2	2	**3**	3
	ATE_DPT			**1**	1	**2**	2	**3**
	ATE_FUN		**1**	1	1	1	**2**	2
	ATE_IND	1	**2**	2	2	2	2	**3**
Оценка уязвимостей	AVA_CCA					**1**	**2**	2
	AVA_MSU			**1**	**2**	2	**3**	3
	AVA_SOF		**1**	1	1	1	1	1
	AVA_VLA		**1**	1	**2**	**3**	**4**	4

Таблица. Обзор оценочных уровней доверия

Оценку программного продукта производим по 22 критериям. Исходные предпосылки для составления программы изложены ниже.

1) Требования к руководству администратора построены так, что ограничения среды будут поняты администраторами и операторами объекта оценки (ОО). Руководство администратора – основное средство, имеющееся в распоряжении разработчика для предоставления администраторам ОО детальной и точной информации о том, как

осуществлять администрирование ОО безопасным способом и эффективно использовать привилегии функций безопасности ОО (ФБО) и функции защиты.

2) Семейство «Установка, генерация и запуск» предусматривает, чтобы копия ОО была конфигурирована и активизирована администратором так, чтобы показать те же самые свойства защиты, что и у оригинала ОО. Процедуры установки, генерации и запуска дают уверенность в том, что администратор будет осведомлен о параметрах конфигурации ОО и о том, как они способны повлиять на ФБО.
3) Требования к руководству пользователя направлены на то, чтобы пользователи могли эксплуатировать ОО безопасным способом (например, ограничения использования, предусмотренные профилем защиты (ПЗ) или заданием по безопасности (ЗБ), необходимо четко объяснить и проиллюстрировать). Руководство – основное средство, имеющееся в распоряжении разработчика, для предоставления пользователям ОО необходимой общей и специфической информации о том, как правильно использовать функции защиты ОО. В руководстве необходимо осветить два аспекта. Во-первых, требуется объяснить, что делают доступные пользователю функции безопасности и как они будут использоваться, чтобы пользователи имели возможность последовательно и действенно защищать свою информацию. Во-вторых, требуется разъяснить роль пользователя в поддержании безопасности ОО.
4) Семейство «Устранение недостатков» направлено на то, чтобы недостатки, обнаруженные потребителями ОО, отслеживались и исправлялись, пока ОО сопровождается разработчиком.

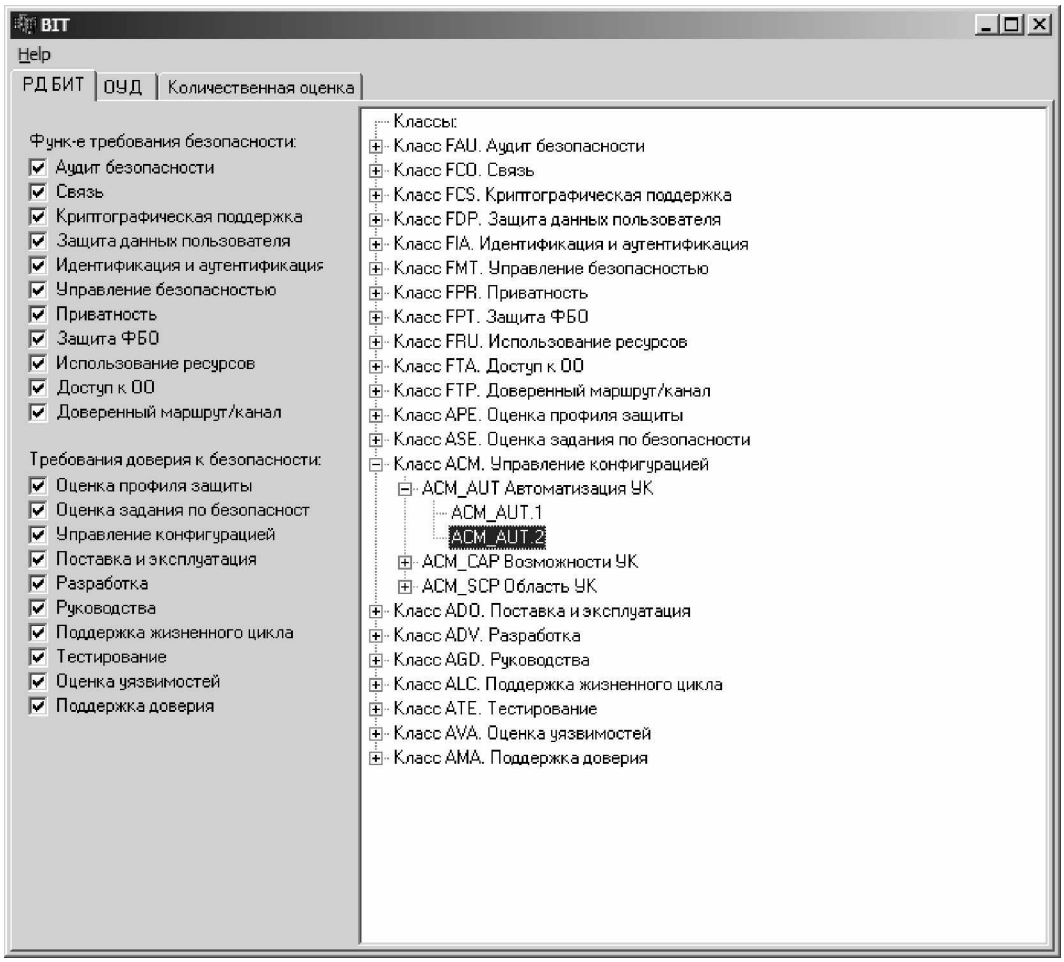
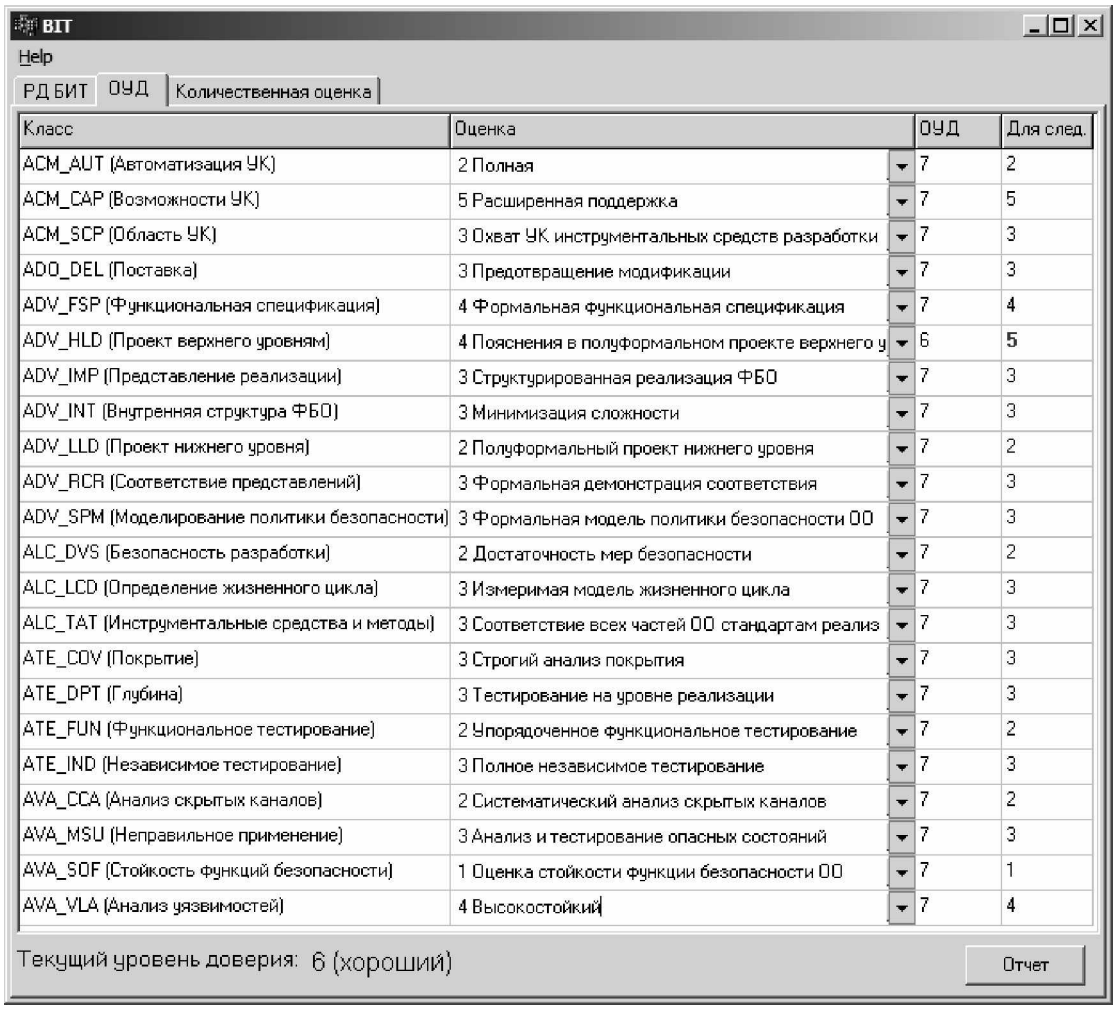


Рис. 1. Навигатор по РД БИТ

Рис. 2. Дерево классов



Класс	Оценка	ОУД	Для след.
ACM_AUT (Автоматизация УК)	2 Полная	7	2
ACM_CAP (Возможности УК)	5 Расширенная поддержка	7	5
ACM_SCP (Область УК)	3 Охват УК инструментальных средств разработки	7	3
ADO_DEL (Поставка)	3 Предотвращение модификации	7	3
ADV_FSP (Функциональная спецификация)	4 Формальная функциональная спецификация	7	4
ADV_HLD (Проект верхнего уровням)	4 Пояснения в полуформальном проекте верхнего у	6	5
ADV_IMP (Представление реализации)	3 Структурированная реализация ФБО	7	3
ADV_INT (Внутренняя структура ФБО)	3 Минимизация сложности	7	3
ADV_LLD (Проект нижнего уровня)	2 Полуформальный проект нижнего уровня	7	2
ADV_RCR (Соответствие представлений)	3 Формальная демонстрация соответствия	7	3
ADV_SPM (Моделирование политики безопасности)	3 Формальная модель политики безопасности ОО	7	3
ALC_DVS (Безопасность разработки)	2 Достаточность мер безопасности	7	2
ALC_LCD (Определение жизненного цикла)	3 Измеримая модель жизненного цикла	7	3
ALC_TAT (Инструментальные средства и методы)	3 Соответствие всех частей ОО стандартам реализ	7	3
ATE_COV (Покрытие)	3 Строгий анализ покрытия	7	3
ATE_DPT (Глубина)	3 Тестирование на уровне реализации	7	3
ATE_FUN (Функциональное тестирование)	2 Упорядоченное функциональное тестирование	7	2
ATE_IND (Независимое тестирование)	3 Полное независимое тестирование	7	3
AVA_CCA (Анализ скрытых каналов)	2 Систематический анализ скрытых каналов	7	2
AVA_MSU (Неправильное применение)	3 Анализ и тестирование опасных состояний	7	3
AVA_SOF (Стойкость функций безопасности)	1 Оценка стойкости функции безопасности ОО	7	1
AVA_VLA (Анализ уязвимостей)	4 Высокостойкий	7	4

Рис. 3. Получение оценочного уровня доверия

Первая часть (рис. 1) программы представляет собой упрощенный навигатор по РД БИТ. Наглядно предоставлены классы из второй и третьей части РД, которые

описывают функциональные требования к безопасности и требования доверия к безопасности. Классы представлены в виде дерева классов, к каждому элементу которого, вне зависимости от уровня, есть описание (см. рис. 2).

Вторая часть программы предназначена для получения оценочного уровня доверия. Там представлены все классы проверки, необходимые для получения ОУД. В зависимости от проведенных проверок текущий ОУД динамически меняется. Красным цветом выделяются те классы, предоставление которых необходимо для повышения ОУД (см. рис. 3). Когда оценка закончена, генерируется отчет, содержащий в себе текущий уровень ОУД и требования для его увеличения. Пример отчета приведен на рис. 4.

Рис. 4. Пример отчета

На рис. 5 представлена последняя, третья часть, которая позволяет узнать рейтинг уязвимости и возможности системы по противостоянию злоумышленнику.

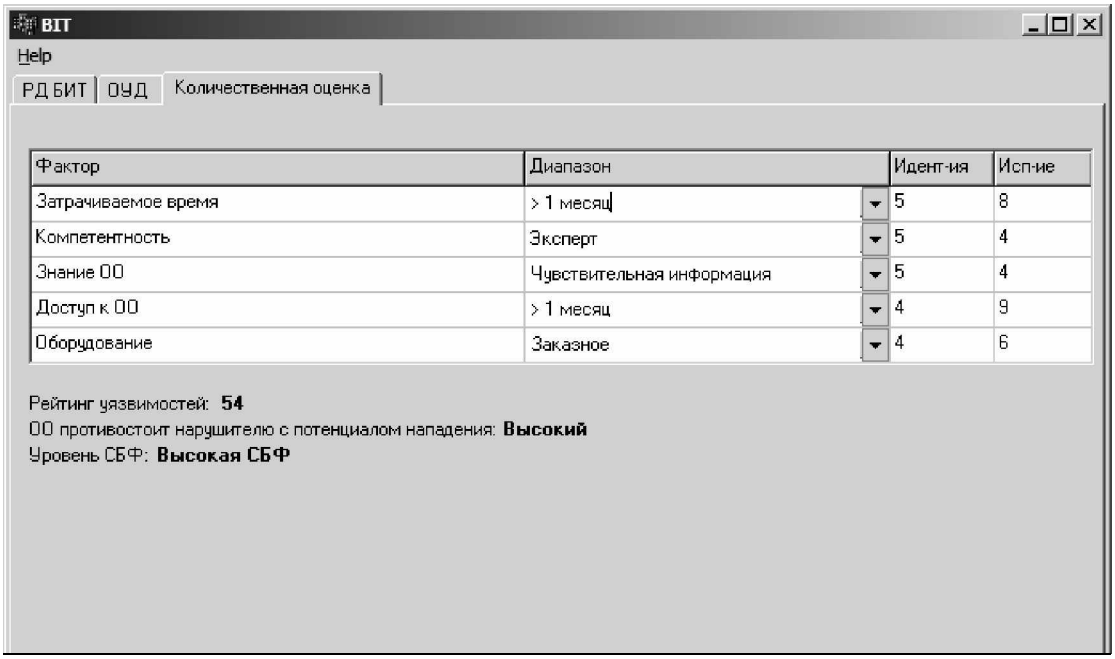


Рис. 5. Оценка рейтинга уязвимости

Оценка зависит от таких параметров, как время доступа к объекту, наличие информации об объекте, используемое на объекте оборудование и затрачиваемое время. На выходе выдается рейтинг уязвимости (численная характеристика), уровень СБФ и, соответственно, потенциал нарушителя, которому система может противостоять.

Заключение

Применение разработанной программы позволяет значительно уменьшить трудозатраты при оценке уровня доверия, что ведет к повышению качества оценки.

МОДЕЛЬ СОЗДАНИЯ ПРОФИЛЕЙ ЗАЩИТЫ ДЛЯ СЕТЕЙ СВЯЗИ И СИСТЕМ КОММУТАЦИИ

Н.А. Минакова

Научный руководитель – доктор технических наук, профессор Л.Г. Осовецкий

В статье рассматривается подход к созданию профилей защиты для сетей связи и систем коммутации. Основу выбранного решения составляет метод, определяющий критичность сегментов (МОУКС). На основании определенного уровня критичности и предполагаемых угроз для сегмента к нему предъявляются функциональные требования безопасности (ФТБ) и оценочные уровни доверия (ОУД) из ISO 15408 определенной иерархии, что позволит создать обоснованный профиль защиты для сети связи и системы коммутации.

Введение

С 2007 г. отменяются руководящие документы Федеральной службы по техническому и экспортному контролю Российской Федерации (ФСТЭК), позволяющие проводить аттестацию и сертификацию объектов информатизации по требованиям безопасности, и вступает в силу новый международный стандарт ISO 15408 [1]. На сегодня ФСТЭК имеет в проекте порядка 20 профилей защиты автоматизированных систем и средств вычислительной техники, и не один из них не описывает подхода для определения требований безопасности к сетям связи и узлам коммутации, что обусловливает актуальность выбранной тематики [2].

Работа находится на стыке двух отраслей – телефонии и информационной безопасности. Первая отрасль успешно развивается с 1876 г., когда Александром Беллом был изобретен первый телефонный аппарат. Вторая – более молодая и менее изученная, требующая разработки новых методов и моделей – возникла на базе информационных технологий, активно внедряемых в область телефонии. В свою очередь, развитие и интеграция IT-решений приводит к появлению новых видов угроз и уязвимостей, требующих внедрения новых решений по защите или комплексного применения уже известных и проверенных.

Разработка таких решений ведется рядом телекоммуникационных компаний и центров по защите информации. Согласно законодательству Российской Федерации, системы защиты информации должны иметь сертификат ФСТЭК на определенный класс защищенности или соответствовать требованиям профиля защиты или задания по безопасности. Для обоснования этих требований и служит метод определения уровня критичности сегмента, что должно акцентировать внимание оценщика на более уязвимых частях сетей определенного класса. Обоснование конкретных требований по защите происходит на основании описания угроз, приводимого в МОУКС, а уровень этих требований (отражается иерархией элемента в компоненте ФТБ из ISO 15408) обосновывается уровнем критичности сегмента по МОУКС.

Модель создания профилей защиты для сетей связи и систем коммутации

Основные компоненты модели представлены на рисунке:

(1) МОУКС, включающий сегментацию и классификацию сетей связи и систем коммутации, а также показатели критичности для каждого из сегментов с точки зрения информационной безопасности (в том числе ГОСТ Р 51275-99) [3];

(2) РД «Безопасные информационные технологии. Критерии оценки защищенности» (ISO 15408) [1].

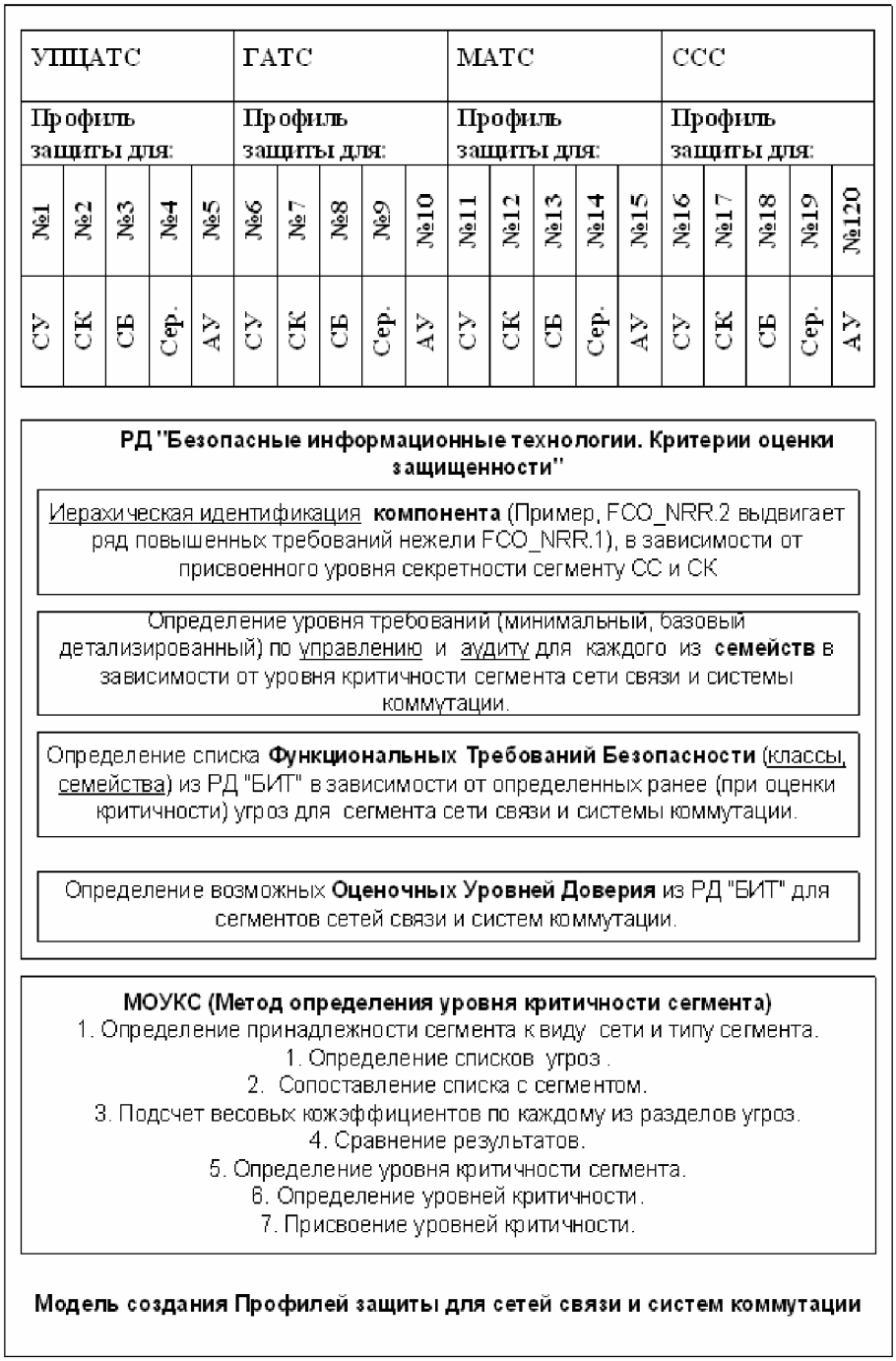


Рис. Модель создания профилей защиты для сетей связи и систем коммутации

Классификация и сегментация сетей связи

Сети связи и системы коммутации с точки зрения информационных технологий можно разделить на два сегмента – систему управления и систему коммутации [4].

С точки зрения угроз информационной безопасности целесообразно выделять 5 сегментов: система управления (СУ), система коммутации (СК), система биллинга (СБ), абонентские устройства (АУ) и сервисы.

Современные сети связи также делятся на две категории: телефонные сети общего пользования (ТфОП), отличающиеся наземными каналами связи, и сети сотовых операторов, отличающиеся мобильностью абонента и предоставлением большого

спектра сервисов (мы не рассматриваем правительственные сети связи, а также сети связи спецслужб, так как на них существуют закрытые стандарты).

Телефонные сети общего пользования с точки зрения угроз для сегментов корректно было бы разделить на ведомственные, городские и междугородние [5]. Подобного деления у сотовых операторов не происходит. На практике возможна детализация по тарифам (корпоративные и остальные), но все угрозы в этом случае будут сведены к сервисам, которые, в свою очередь, уже представлены в виде сегмента.

Таким образом, предлагается следующая классификация сетей связи с точки зрения информационной безопасности:

(1) ведомственные;
(2) городские;
(3) междугородние;
(4) сети сотовых операторов.

Каждый вид сети состоит из пяти сегментов, описанных выше, кроме междугородней сети, в которой отсутствует сегмент абонентских устройств.

Метод определения уровня критичности сегмента (МОУКС)

Как видно из рисунка, МОУКС работает по алгоритму, представленному ниже.

1. Задание критериев, которые необходимо учитывать при определении критичности сети связи и систем коммутации:
   - основных угроз;
   - факторов, воздействующих на защищаемую информацию;
   - специфических угроз [6];
   - сценариев атак.

   Результат – развернутый список (ФВЗИ, специфичные угрозы для СС и СК, стандартные угрозы для ИБ ОИ, сценариев атак), обоснование выбранных критериев.
2. Анализ каждого из сегментов на наличие в нем специфических угроз, основных угроз, факторов воздействующих на защищаемую информацию и сценарии атак.

   Результат – 20 списков по каждому из 20 видов сегментов, отражающих ФВЗИ, специфичные угрозы для СС и СК, стандартные угрозы для ИБ ОИ, сценарии атак.
3. Подсчет общего суммарного значения для каждого из 20 видов сегментов по каждому из критериев.

   Результат – числовые коэффициенты, отражающие наличие в сегменте определенного ряда угроз из четырех списков критериев.
4. Сравнение результатов.

   4.1. Сравнение результатов для сегментов одного типа в разных сетях.

   Результат – критичность сегмента по сравнению с другими сетями.

   4.2. Сравнение результата для сегментов разного типа одной сети.

   Результат – критичность сегмента сети по сравнению с другими сегментами этой сети.

   4.3. Сравнение результатов при анализе сценариев атак.

   Результат – список сегментов, уязвимых через рассматриваемый сегмент, и список сегментов, через которые рассматриваемый сегмент становится уязвимым.
5. Определение уровня критичности сегмента на основе:
   - суммарного веса ФВЗИ,
   - суммарного веса специфических угроз,
   - суммарного веса стандартных угроз,
   - коэффициента уязвимости основанного на сценарии анализа атак.
6. Определение 5 уровней критичности при анализе результатов по суммарному весу коэффициентов (коэффициент критичности сегмента).
7. Присвоение уровня критичности каждому из 5 сегментов 4-х различных видов сети.

Результат при оценке сценариев атак на сегменты сети связи и системы коммутации

НСД к одному из сегментов может привести к воздействию на другие сегменты, НСД к ним, к прямому выводу из строя, модификации данных сегмента или реализации какой-либо другой угрозы, приводящей к одному из видов ущерба.

Как видно из табл. 1, наиболее уязвимыми сегментами являются система биллинга (СБ) и сегмент абонентских устройств (АУ). Воздействие на них возможно со стороны всех 5 остальных сегментов, определенных в модели.

СУ (3)	← СУ ← СК ← Ї СБ ← Ї С ← АУ	СК (3)	← СУ ← СК ← Ї СБ ← Ї С ← АУ	СБ (5)	← СУ ← СК ← СБ ← С ← АУ	Сервисы (2)	← СУ ← Ї СК ← Ї СБ ← С ← Ї АУ	АУ (5)	← СУ ← СК ← СБ ← С ← АУ

Таблица 1. Анализ сценариев атак для сегмента

Сегмент	Коэффициент критичности	Коэф. уязвимости	Суммарный вес
СУ	5	3	8
СК	4	3	7
СБ	2	5	7
Сер	3	2	5
АУ	4	5	9

Таблица 2. Косвенные коэффициенты сегментов

Косвенно критичными являются те системы, которые более других способны привести к потерям, если к ним будет произведен НСД. На первом месте по критичности стоит СУ, при НСД к ней возможно нанесение ущерба всем 5 сегментам сетей связи и систем коммутации: 1– СУ, 2 – СК, АУ, 3 – Сер., 4 – СБ (коэффициенты критичности приведены в табл. 2). Соответственно, проводятся оценки показателей по другим критериям (ГОСТ, специфическим угрозам и возможности модификации, уничтожения и хищения информационных ресурсов (ИР) сегмента). В частности, оценка возможности хищения, модификации, уничтожения ИР сегмента сети связи в системе коммутации приведена в табл. 3.

Под информационными ресурсами подразумевается оборудование сегмента и информация, обладающая ценностью. Ценность информации зависит от возможного нанесения ущерба оператору, абоненту или третьему лицу в случае хищения, модификации или уничтожения.

Представляется невозможным провести оценку вероятности угроз по причине отсутствия или недоступности статистики возникновения угрозы, поэтому предлагается оценить возможность возникновения или отсутствие угрозы как таковой для сегмента сети связи или системы коммутации.

Таким образом, одной из наиболее уязвимых сетей связи является сеть связи сотовых операторов, что подтверждает аксиому «При увеличении функциональных возможностей увеличивается и количество угроз», характерную и для СС, и для СК.

Угрозы по сегментам	Вид сети связи (системы коммутации)			
Угроза СУ	**УПАТС**	**ГАТС**	**АМТС**	**ССС**
Хищение Модификация Уничтожение	- + -	- + -	- + -	- + -
Угроза СК	УПАТС	ГАТС	АМТС	ССС
Хищение Модификация Уничтожение	+ + +	+ + +	+ + -	- + -
Угроза СБ данных	УПАТС	ГАТС	АМТС	ССС
Хищение Модификация Уничтожение	- + -	- + -	+ + -	+ + +
Угроза Сервисы	УПАТС	ГАТС	АМТС	ССС
Хищение Модификация Уничтожение	+ + -	+ + +	+ + +	+ + +
Угроза АУ	УПАТС	ГАТС	АМТС	ССС
Хищение Модификация Уничтожение	- + -	- + -	- - -	+ + -
Количество угроз для сети	**8**	**8**	**8**	**10**

Таблица 3. Оценка возможности хищения, модификации, уничтожения ИР сегмента сети связи в системе коммутации

Профиль защиты для системы управления УПЦАТС

Согласно модели, профиль защиты создается для вида сети связи: ведомственной, городской, междугородней или сотовой. В профиль защиты из модели включаются:

(1) ОУД,

(2) пПакеты ФТБ,

(3) сСписок угроз.

Каждый из сегментов представлен пакетом ФТБ.

Функциональный пакет – предназначенная для многократного использования совокупность функциональных компонентов, объединенных для удовлетворения совокупности определенных целей безопасности [1].

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. А вот для УПЦАТС, напротив, оценка сегмента АУ будет стоять на 2-ом месте после системы управления, параллельно системе коммутации, о чем можно судить уже по результатам произведенного анализа угроз.

По результатам работы модели будет создан профиль защиты для системы управления ведомственной сетью связи и системой коммутации.

Заключение

Модель не претендует на уникальность выбранного способа оценки (получения количественных показателей на основе качественных), но позволяет производить оценку сетей связи и систем коммутации с точки зрения информационной безопасности, учитывая законодательный аспект, специфику объекта, опыт оценки стандартных объектов информатизации.

Модель создания профилей защиты для сетей связи позволит выдвинуть ряд обоснованных требований к защите сетей связи и систем коммутации и сформировать их в профиль или задание по безопасности [7].

Разработка профилей защиты для сетей связи и систем коммутации позволит аттестовать сети связи и сертифицировать системы коммутации согласно международному стандарту ISO 15408 (РД «Безопасные информационные технологии. Критерии оценки защищенности». 1, 2, 3 часть).

Литература

1. Руководящий документ ФСТЭК «Безопасные информационные технологии. Критерии оценки защищенности». Федеральная служба по техническому и экспортному контролю, 2002. 1, 2, 3 часть.
2. Минакова Н.А. Нормативно правовое обеспечение методики аттестации УПЦАТС по требованиям безопасности // Теория и технология программирования и защиты информации. Применение вычислительной техники. 2005. С. 89–95.
3. ГОСТ Р 51275-99 «Защита информации. Объект информатизации. Факторы, воздействующие на защищаемую информацию».
4. Гончарок М.Х., Крюков Ю.С. Построение СЗИ в цифровых АТС и выбор класса защищенности // Защита информации. Конфидент. 2004. № 2.
5. Ярмухаметов А. Информационная безопасность корпоративных (ведомственных) сетей связи // Информост – радиоэлектронника и телекоммуникации. 2002. № 3.
6. Минакова Н.А. Аспекты информационной безопасности УПЦАТС // Теория и технология программирования и защиты информации. Применение вычислительной техники. 2005. С. 45–49.
7. Проект «Концепция информационной безопасности сети связи общего пользования взаимоувязанной сети связи Российской Федерации» Министерство связи и информатизации, 2002.

МЕТОДЫ DATA MINING В АВТОМАТИЗИРОВАННОМ ПОСТРОЕНИИ ПРОФИЛЯ ПОЛЬЗОВАТЕЛЯ ЗАЩИЩАЕМОЙ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ

М.Л. Гарусев (Российский государственный гуманитарный университет)
Научный руководитель – доктор технических наук, профессор Е.Е. Тимонина (Российский государственный гуманитарный университет)

Предложенный автором подход по применению технологии Data Mining для эффективного накопления и обработки данных аудита автоматизированных систем может помочь в решении ряда проблем в предметной области по минимизации привлечения ресурсов людей-экспертов при функционировании системы защиты информации. Data Mining – подвид прикладного направления в отрасли анализа данных – Knowledge Discovery.

Введение

До недавнего времени главной целью служб информационной безопасности являлось обнаружение злоупотреблениями системных или сетевых ресурсов (несанкционированный доступ к ним). Чтобы разрешить эту проблему, предприятия осуществили закупку и инсталляцию огромного количества инструментальных средств защиты, включая межсетевые экраны, системы сканирования защищенности, системы обнаружения вторжений и т.д. В то время как каждая из этих программ обеспечивает необходимую функцию защиты, проблема состоит в большом объеме регистрационных данных, который они накапливают. В сети каждого предприятия работает большое количество компьютеров и сетевых устройств, которые генерируют и накапливают файлы (или базы данных) событий, с которыми они сталкиваются. Эти события обычно достаточно специфичны для каждого вида операционной системы, приложения или сетевого оборудования, и для каждого типа программно-аппаратного комплекса набор событий представляет собой конечное множество. Когда в сети реализуется какая-либо атака на информационные ресурсы (несанкционированный доступ [1, 2]), существует высокая вероятность того, что атакующие злоумышленники оставили в файлах регистрации какие-либо признаки нападения. Отметим также риск бесполезного «шума сообщений» из-за ошибочных срабатывания сенсоров обнаружения вторжений.

Традиционно важной составляющей комплексной системы защиты информации является подсистема защиты от несанкционированного доступа [3], контролирующая доступ пользователей распределенной информационной системы к ее ресурсам. Вся информация о доступе субъектов к объектам сохраняется системой аудита распределенной информационной системы [3]. Проблемой является разнородный формат данных аудита, вызванный гетерогенной средой происхождения информации по безопасности. Различные элементы комплексной системы защиты информации, будучи разработанными различными производителями с использованием различных стандартов, не в состоянии обеспечить унифицированное представление данных аудита для его дальнейшей интеллектуальной обработки.

Решением данной проблемы занимаются так называемые SEM-системы (Security Event Management [4], другие часто встречающиеся названия – Security Management System (SMS) [5], Active Security Management (ASM) [6], Security Information Management (SIM) [7]). В основе построения таких систем лежит принцип консолидированного сбора информации по безопасности в единой точке распределенной информационной системы и ее совместной обработки. Этот подход получил название корреляции событий (event correlation) [8–10]. Базой технологии является возможность оперирования шаблонами событий, находящимися в исходных данных аудита.

При этом на настоящий момент все существующие SEM-системы предлагают подход выработки шаблонов вручную, с помощью людей-экспертов. Все разработчики, как правило, имеют собственные исследовательские команды, задачей которых является анализ форматов данных аудита производителей средств информационной безопасности и их включение в единую SEM-систему.

Решением задачи нахождения шаблонов в первичных данных может служить новейшая технология Knowledge Data Discovery и одна из ее составляющих – технология Data Mining (дословно «разработка данных») [11]. Data Mining – это исследование и обнаружение «машиной» (алгоритмами, средствами искусственного интеллекта) в первичных данных скрытых структур или зависимостей, которые ранее не были известны.

Службы ИБ нуждаются в средствах интеллектуального анализа данных аудита в любой информационной системе, с помощью которых было бы возможно увеличить качество и скорость принятия решений, сократив объем обрабатываемых данных. Под качеством решений понимается то, что система при принятии любого решения пытается учесть как можно больше факторов, влияющих на текущую ситуацию с режимом безопасности. Скорость принятия решений в идеальном случае должна позволять системе являться системой реального времени.

Системы реального времени являются одним из самых важных классов программного обеспечения. Согласно [12], они определяются как системы, имеющие жесткие временные ограничения. При этом выделяются системы *on-line*, к которым относятся, в частности, многие бизнес-приложения (например, система заказа авиабилетов, которая имеет много рабочих мест и должна быстро обрабатывать большое число запросов), и ПО для встроенных систем управления каким-либо оборудованием (embedded systems). В [13] говорится, что «системы реального времени – это такие системы, в которых правильность работы зависит не только от логических результатов вычислений, но и от времени, затраченного на получение этих результатов».

Таким образом, актуальность выбранной темы для исследования определяется необходимостью разработки языков, обеспечивающих анализ данных аудита в гетерогенных системах при возможных временных ограничениях. На базе этих языков служба безопасности предприятия в режиме реального времени может обнаруживать атаки, что, в свою очередь, требует разработки метода быстрого формирования профилей по различным направлениям функционирования распределенной информационной системы (частный случай – профиль пользователя такой информационной системы).

Новизна темы состоит в переносе концепций распределенных вычислений, развиваемых в рамках сетевых технологий, в сферу защиты информации, т.е. данные концепции переносятся на задачу построения профилей пользователей защищаемой информационной системы предприятия. Базой для решения таких задач является понятие события – сообщения некоторого формата, независимого от аппаратной платформы, в которой произошел инцидент, о причине возникновения данного сообщения.

Использования методов Data Mining для создания языка анализа данных аудита

События в распределенной системе находятся в различных отношениях друг к другу. Самое простое отношение – время возникновения события A перед событием B относительно системных часов. Обычно синхронизация представлена временными метками в событиях. Причинность – другой вид отношений между событиями.

Существует теоретическая предпосылка: событие в системе корреляции событий является причиной другого события в определенных ситуациях; иначе любые два события независимы. Далее мы покажем, как важны причинно-следственные связи для реализации распределенной системы корреляции.

Когда работает система корреляции событий, происходит моделирование какого-то кусочка реального мира, где присутствуют события, составляющие суть функционирования системы, отношения между ними, история их происхождения, а также возникновение сторонних, независимо возникающих событий (например, внешних по отношению к системе). Для удобства дальнейшего изложения назовем некоторую последовательную цепочку событий частично упорядоченным набором событий, потому что различные отношения (т.е. причина и время) между событиями, которые явно представлены в такой истории, являются частично упорядоченными событиями.

Здесь не требуется давать дополнительных пояснений, так как для человека очень естественно оценивать мир в причинно-временных понятиях. Мы рождаемся, чтобы изучать законы окружающего мира, и во многом действуем автоматически, подчиняясь шаблонам поведения, реактивным зависимостям на некоторые раздражители и биологическим циклам.

Если говорить формально о причинности, то можно отметить, что это явление во многом является определяющим для корреляции событий или тревог. Абстрактно, входной поток для системы корреляции представляет собой последовательность сообщений E1 = (T1, S1, D1, M1), E2 = (T2, S2, D2, M2), E3 = (T3, S3, D3, M3), …., где E – само событие, сообщение, тревога; T – время возникновения события; S – источник события, отправитель сообщения; D – получатель сообщения; M – сама информативная часть события, сообщения.

Для такой последовательности должно выполняться условие $Ti \leq Ti+1$. Корреляционный шаблон (correlation pattern) описывает ситуацию, когда можно распознать в потоке определенную последовательность событий внутри некоторого временного окна определенной длины.

Здесь мы первый раз сталкиваемся с шаблонами, которые на самом деле являются вторым столпом теории корреляции. Человек всю свою жизнь наблюдает мир вокруг себя, изучает его структуру, определяя причины каких-либо действий и явлений и связанные с ними эффекты; собирает и документирует повторяющиеся решения, полученные при различных условиях. Эти эмпирические правила, представляющие регулярность поведения некоторого объекта или субъекта, называются «образами», «образцами» или «шаблонами». Визуальные образы, которые мы запоминаем – самое простое выражение концепции шаблонов. Очень много шаблонов сама природа заложила в наше сознание в виде инстинктов: мы наследуем от предков действия и реакции, которые гарантируют нам выживание при определенном наборе внешних воздействий. Другие шаблоны должны быть изучены при развитии человека, в детском саду, школе, университете и т.п. и формируют уже искусственные расширения человеческого сознания. Способность изучать и выявлять шаблоны предоставляет нам человеческие преимущества – приспосабливаться к внешней среде и, приспособившись – уметь ее изменить. Недаром ученые отмечают, что основной навык детей – это подражание. Разумеется, каждое событие обрастает миллионом частных мелочей, зависящих от конкретной ситуации, и эта поверхностная сложность автоматически удаляется нашим сознанием, чтобы добраться до сути события (например, обжечься можно на костре, плите, от раскаленного металла, однако автоматическая реакция на суть явления – ожог – будет одна – человек отдернет руку).

Теория шаблонов получила свое развитие в архитектуре, а первое определение шаблона, которое мы привели выше, было введено К. Александром в работе «The

Theory» в 1977 г. [13]. Именно архитекторы увидели первыми, что из архитектурных элементов можно создавать библиотеки решений и заниматься верификацией сооружений на соответствие определенным типам. А в начале 90-х эта теория пришла в информатику, где получила широчайшее развитие во многих прикладных областях информатики.

Очень интересное применение шаблоны получили в прикладном направлении Data Mining. Особенно активное развитие методы Data Mining получили в области анализа данных, находящихся под управлением СУБД [14]. И это естественно, так как данные в СУБД уже находятся в структурированной форме – разбиты на поля и записи. Сразу же отметим, что все современные средства защиты информации умеют работать с СУБД, а решений, предоставляющих результаты своей работы в неструктурированном виде, практически не осталось на рынке – из-за повсеместного требования унификации и стандартизации различного вида ресурсов. Та модель, которая будет описана далее в статье, также сводит слабо структурированные данные из текстовых файлов аудита к форматированным данным.

Построение модели автоматизированного построения профилей пользователей

Для применения технологии Data Mining нужно создать модель подсистемы защиты информации одной из реальных компьютерных систем. Наша модель будет использовать два метода Data Mining – метод поиска эпизодических или частотных правил (episode rules) [15,16] и метод поиска ассоциативных правил (associate rules) [17]. Это следует из того, что работа пользователя в информационной системе определяется, во-первых, временными зависимостями, а во-вторых – зависимостями в окружающей среде (т.е. отношениями между объектами и субъектами в системе).

Метод поиска эпизодических правил (или поиск эпизодов) поможет нам найти все рекуррентные последовательности в базе данных аудита некоторой информационной системы, причем с учетом временных зависимостей, а ассоциативные правила дадут возможность оценить параметры каких-либо явлений в этой же базе данных.

Основной вид правила эпизода – следующий:

IF определенная комбинация событий наблюдается в период времени T, THEN другой набор событий произойдет внутри этого периода времени T с определенной вероятностью.

Такое правило в реальной системе может принять вид:

IF системный администратор вошел с консоли и в течение 5 секунд запустил команду «mount /dev/hdb5», THEN он в течение 30 секунд запустит команду резервного копирования «tar» с вероятностью 60%.

При этом в реальном системном журнале, например, некоторой Linux-системы, это может выглядеть следующим образом:

```
Jan 5 12:00:47 localhost login(pam_unix)[1279]: session opened for user root by (uid=0)
Jan 5 12:00:47 localhost -- root[1279]: ROOT LOGIN ON vc/1
Jan 5 12:03:04 localhost execute "mount" (uid=root) with option (mount /dev/hdb5 /backup)
Jan 5 12:03:27 localhost execute "tar" (uid=root) with option (tar acz /backup)
```

Здесь обратим внимание на то, что эти связанные события находятся в разных строках системного журнала, что затрудняет их обработку, могут различаться временные интервалы между ними, а между строками могут находиться другие записи систе-

мы аудита информационной системы, не имеющие никакого отношения к действиям, совершаемым пользователем root.

Ассоциативные правила имеют следующую форму:

IF событие имеет определенную комбинацию параметров, THEN это событие имеет еще некоторые другие параметры с определенной очевидностью.

В реальной системе это может означать:

IF системный администратор вошел с консоли tty1 в бизнес-часы в рабочий день THEN процесс LOGIN завершился успешно с очевидностью 98%.

Обычно этот метод используют для нахождения так называемого «скрытого» знания, т.е. ищут неочевидные зависимости, с малой очевидностью. Но для целей наблюдения за информацией, накапливаемой подсистемой аудита, мы будем применять этот метод с точностью до наоборот – нас будут интересовать наиболее тривиальные зависимости. Именно из них мы и будем формировать профиль пользователя информационной системы (т.е. новые поступающие данные будут оцениваться на соответствие с правилами с высокой очевидностью (их еще называют тривиальными)).

Для поиска правил эпизода был задействован алгоритм ECLAT [18] в программной реализации одного из исследователей рабочей группы нейронных сетей и систем нечеткой логики Департамента обработки знаний и инжиниринга языков университета Магдебурга К. Боргельта [19]. Это не самая последняя модификация поискового алгоритма, но в своем роде реализации Боргельта являются самыми простыми и оригинальными, поэтому могут служить своего рода отправной точкой для исследователей, создающих свои версии алгоритмов поиска ассоциативных правил и правил эпизодов.

Поиск осуществлялся над системным файлом /var/log/messages операционной системы Linux. Отметим сразу серьезную проблему, которая встала перед автором при исследовании – все алгоритмы, подобные ECLAT, осуществляют перебор в множестве целых чисел (это связано с тем, что все множества в алгоритмах отражаются на бинарные матрицы). Т.е. алгоритм требовал подачи на свой входной интерфейс строк вида:

```
0 1 9 1 7 1 8 1 0 1 8 12
1 8 1 8 1 9 1 10 11 10 2 0 1 9 13 9 1 0 1 8 1 0 3 16
9 1 8 12 8 13 8 1 7 1 7 1 8 13 6
0 1 7 1 8
1 9 1 9 1 0
0 1 7 1 8
8 12 8 1 9 13 9 1 7 1 8 1 0 1 0 3 16
16 18 19
.....
```

В данном случае каждая строка является набором неких событий в пределах одного временного «плавающего» окна. В примере явно видно, что событие, закодированное номером 9, всегда следует за событием, закодированным числом 1; частным вариантом является 8, следующая за 9, и т.п. Чтобы решить проблему с нумерацией, было принято следующее решение: во всех строках файла /var/log/messages удалялись численные значения, которые не являлись повторяющимися (т.е. номера процессов в системе, даты, время и т.п.). После этого строки хешировались с помощью алгоритма MD5 к строкам одного вида и одной же длины, и затем эти хеш-значения последовательно нумеровались.

```
1) Nov 12 23:58:30 login(pam_unix)[2072] authentication failure; logname= uid=0 euid=0 tty=vc/3 ruser= rhost= user=maka
```

```
    2) login(pam_unix)[] authentication failure; logname= uid=
euid= tty=vc/ ruser= rhost= user=maka
    3) 063284d13530d3167b90e5793cf958a6

    1) Nov 12 23:58:33 login[2072] FAILED LOGIN 1 FROM (null)
FOR maka, Authentication failure
    2)  login[]  FAILED  LOGIN  FROM  (null)  FOR  maka,
Authentication failure
    3) 371c4189f3705063ed5b52a9029cd665

    1) Nov 12 23:59:00 -- maka[2072] =  LOGIN ON vc/4 BY maka
    2) -- maka[] =  LOGIN ON vc/4 BY maka
    3) eaa10f778958129e990255eaa4a5429a
```

Такой подход с помощью использования СУБД позволил крайне быстро осуществить уникальную нумерацию строк в файле /var/log/messages и подготовить для алгоритма необходимый формат данных (как известно, СУБД очень быстро обрабатывают строки определенного строгого формата и длины). Результаты работы алгоритма представляются в следующем виде:

```
12 (13, 76%)
7 12 (13, 76%)
9 12 (13, 76%)
7 9 12 (11, 62%)
1 8 12 (8, 45%)
1 9 12 (8, 45%)
1 8 9 12 (5, 30%)
1 7 12 (5, 30%)
1 7 8 12 (5, 30%)
1 7 9 12 (3, 14%)
….
```

В результате гипотеза была подтверждена – с помощью алгоритма ECLAT действительно были выявлены повторяющиеся закономерности, наличествующие в системе, среди которых, естественно, было много тривиальных – так, например, тривиально очевидными являлись последовательности строк при загрузке операционной системы – они всегда повторяются, за исключением числовых кодов (даты, времени и т.п.). Наличие таких тривиальных последовательностей сразу решило дополнительную задачу контроля неизменности операционной среды, хотя задачей применения ECLAT было только оперативное создание профиля пользователя (без необходимости чистки «мусора» в журналах аудита). Это происходит ввиду того, что изменения конфигурации системы будут также выявлены при анализе последовательностей строк загрузки.

Для опытов было взято окно длительностью в 3 минуты – это значение было подобрано интуитивно, исходя из предположения, что практически все осмысленные группы действий пользователей будут укладываться в этот интервал. При дальнейших исследованиях весьма интересно изменять данный интервал, например в логарифмическом порядке (5 с, 10 с, 30 с, 1 минута, 5 минут, 10 минут), и посмотреть, как ведет себя алгоритм в таких условиях – повышается или понижается точность работы, а также опытным путем установить оптимальные значения для наблюдения за действиями пользователей.

Итак, мы показали возможность построения временного профиля работы пользователя в системе с помощью метода поиска частотных правил. Теперь вооружимся ассоциативными правилами и проверим гипотезу об их применимости к анализу собственно действий пользователя, отражаемых в рамках одного сообщения. Для тестовой

модели были взяты только сообщения о входе пользователя в систему – для наглядности решения. Ниже приведены исходные строки в Linux-системе.

```
Nov 5 13:40:47 localhost -- root[1279]: ROOT LOGIN ON vc/1
Nov 6 10:40:47 localhost -- root[1279]: maka LOGIN ON vc/4
Nov 7 03:37:31 localhost login[1160]: FAILED LOGIN 1 FROM
(null) FOR root,Authentication failure
```

Путем несложного скрипта трансформируем их к следующему виду (т.е. сформируем наборы значений в каждой транзакции):

```
root Monday BusinessHours LOGIN-ON vc/1
maka Tuesday BusinessHours LOGIN-ON vc/3
root Thursday NonBusinessHours LOGIN-FAILED
```

Это необходимо, так как. все алгоритмы для поиска ассоциативных правил, доступные для использования, работают в формате «Значение1-пробел-Значение2-пробел-…», при этом естественно, что во всех строках (или транзакциях, как их еще называют в теории, Значение1 всегда принимает значения из множества пользователей данной вычислительной системы, Значение2 принимает значения из множества дней недели, и т.п.).

Полученный файл далее был передан на обработку алгоритму APRIORI [20], также в программной реализации К. Боргельта [19]. На текущий момент уже придуманы более совершенные и быстрые реализации алгоритма APRIORI, работающие в битовых матрицах, но реализация Боргельта удобна своими входными и выходными форматами. Результатом работы алгоритма стал файл со строками следующего вида:

```
NonBusinessHours <- Thursday  (33.3%, 100.0%)
Thursday <- NonBusinessHours  (33.3%, 100.0%)
LOGIN-FAILED <- Thursday  (33.3%, 100.0%)
Thursday <- LOGIN-FAILED  (33.3%, 100.0%)
root <- Thursday  (33.3%, 100.0%)
LOGIN-FAILED <- NonBusinessHours  (33.3%, 100.0%)
NonBusinessHours <- LOGIN-FAILED  (33.3%, 100.0%)
root <- NonBusinessHours  (33.3%, 100.0%)
root <- LOGIN-FAILED  (33.3%, 100.0%)
...
```

Для учебной базы с размерностью в 7000 строк (4 пользователя, 7 дней недели, 2 типа времени суток, 8 входных терминалов), алгоритм выработал чуть более 700 правил за несколько секунд.

Заключение

Фактически в данной работе мы построили модель системы автоматического формирования профиля пользователя, причем для каждого правила в профиле оценена его очевидность. Для проверки модели позже был создан программный код, который в режиме реального времени считывал строки из системного журнала и при появлении строк входа в систему осуществлял сравнение с полученными правилами. Проверка показала, что итоговый набор правил вполне адекватно позволил распознать нетипичное время или дату входа пользователя, нестандартный терминал, а также ошибки, допускаемые при наборе пароля.

Таким образом, показано, что DM-методы и применение корреляции событий вполне способны решать задачи по эффективному снижению нагрузки на компьютеры и людей-экспертов в задаче обработке данных аудита автоматизированных систем.

Литература

1. ГОСТ Р 51241-98. Средства и системы контроля и управления доступом. Классификация. Общие технические требования. Методы испытаний.
2. Гостехкомиссия России, Руководящий документ. Защита от несанкционированного доступа к информации. Термины и определения. М., 1992.
3. Гостехкомиссия России, Руководящий документ. Автоматизированные системы. Защита от несанкционированного доступа к информации. Классификация автоматизированных систем и требования по защите информации. М., 1992.
4. Mehdizadeh Y. IT Security Event Management, CISSP, GSEC, June 2004. // www.infosecwriters.com/text_resources/pdf/SEM.pdf
5. Ricoh Group, Information Security, Techical Report, 2004. // http://www.ricoh.com/about/csr/report/pdf2004/15-16.pdf
6. Seung-Hee Oh, Taekyong Nam, Sungwon Sohn. Active Security Management (ASM). Sensor Management Mechanism for Sensor Integrity in Active Security Technology, Electronics and Telecommunications Research Institute, 2002.
7. OpenService Inc., Security Threat Manager. 2005. // http://www.open.com/products/security-threat-manager.jsp
8. Nygate Y. A., Event correlation using rule and object based techniques, Integrated Network Management IV, Chapman and Hall, London, pp. 278–289, 1995.
9. Bouloutas A.T., Calo S.B., Finkel A., and Katzela I. Distributed fault identification in telecommunication networks. // Journal of Network and Systems Managemen. 1995. V. 3. №.3. P. 295–312.
10. Jakobson G. and Weissman M.D. Alarm correlation. // IEEE Network. 1993. V. 7. № 6. P. 52–59.
11. Fayyad U., Piatetsky-Shapiro G. and Smyth P. From Data Mining to Knowledge Discovery in Databases. // AI MAGAZINE. 1996. FALL. P. 37–54.
12. Braek F., Haugen Th. Engineering Real Time Systems. Prentice Hall International (UK) Ltd. p. 398, 1993.
13. Alexander C. Ishikawa et al. The Theory, Oxford Press, London, 1977.
14. Shapiro A. and Frawley W.J. Knowledge Discovery in Databases. // AAAI Press, Menlo Park, California. 1991. P. 325–345.
15. Agrawal R. and Srikant R. Mining sequential patterns. // Proceedings of the 11th International Conference on Data Engineering (ICDE' 95), Taipei, Taiwan. March 1995. pp. 3–14.
16. Mannila H., Toivonen H. and Verkamo A.I. Discovery of frequent episodes in event sequences, *Data Mining and Knowledge Discovery*, Vol. 1, No. 3, pp. 259–289, 1997.
17. Agrawal R., Imielinski T. and Swami A. Mining association rules between sets of items in large databases. // P. Buneman and S. Jajodia (eds.), Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD' 93), ACM, Washington, D.C. May 1993. P. 207–216.
18. Zaki M.J., Parthasarathy S. and Li W. A localized algorithm for parallel association mining. // ACM Symposium Parallel Algorithms and Architectures, Jun 1997.
19. http://fuzzy.cs.uni-magdeburg.de/~borgelt/index.html
20. Borgelt C. and Kruse R. Induction of Association Rules: Apriori Implementation. // 14th Conference on Computational Statistics, Berlin, Germany, Compstat. 2002.

КОМПЛЕКС ЛОГИЧЕСКОГО КОНТРОЛЯ ИСПОЛЬЗОВАНИЯ ОБЩИХ КРИТЕРИЕВ

Е.О. Калашник

Научный руководитель – доктор технических наук, профессор Л.Г. Осовецкий

Статья посвящается разработанному комплексу логического контроля использования общих критериев для поддержки специалистов при работе с ОК.

Введение

Не вызывает сомнений, что с введением ГОСТ Р ИСО/МЭК 15408 «Общие критерии» (далее ОК) потребовалось разработать единый подход к написанию профилей защиты (заданий по безопасности), которые являются одними из направлений ОК в области безопасности информационных технологий [1]. Представление ОК в формализованной графической форме позволило бы специалисту наиболее объективно и эффективно спроектировать и разработать профиль защиты (задание по безопасности).

На сегодняшний день в России нет отечественных программных продуктов, позволяющих реализовать проверку правильности использования ОК при написании профиля защиты.

Учитывая большое количество соотношений между требованиями, которые должны быть, как правило, комплексно удовлетворены, автором был разработан комплекс логического контроля использования ОК, который позволяет осуществлять проверку достаточности логического использования функциональных компонентов [2] и компонентов доверия [3] для формального написания профиля защиты.

Методика моделирования

Моделирование проводится с точки зрения разработчика профиля защиты (задания по безопасности). Модель содержит структурированное графическое представление классов требований безопасности. Основная цель моделирования – это представление для специалиста, разрабатывающего профиль защиты, ключевой системы классов требований безопасности стандарта ОК, а также контроль их логического использования.

На данный момент модель содержит 21 класс требований безопасности по ОК. Возможно добавление новых функциональных компонентов (см. рис. 1) и компонентов доверия (см. рис. 2), если того требует разработчик.

Отметив те семейства, которые разработчик выбрал для своего профиля защиты, автор, как это продемонстрировано на рис. 3, может определить, правильно ли и достаточно ли он обозначил свой набор требований безопасности, который является наименьшим выбираемым набором требований безопасности.

Заключение

На сегодняшний день модель является прототипом, первым шагом к написанию единого инструментального программного комплекса для разработки профилей защиты, который облегчил бы задачу разработчикам.

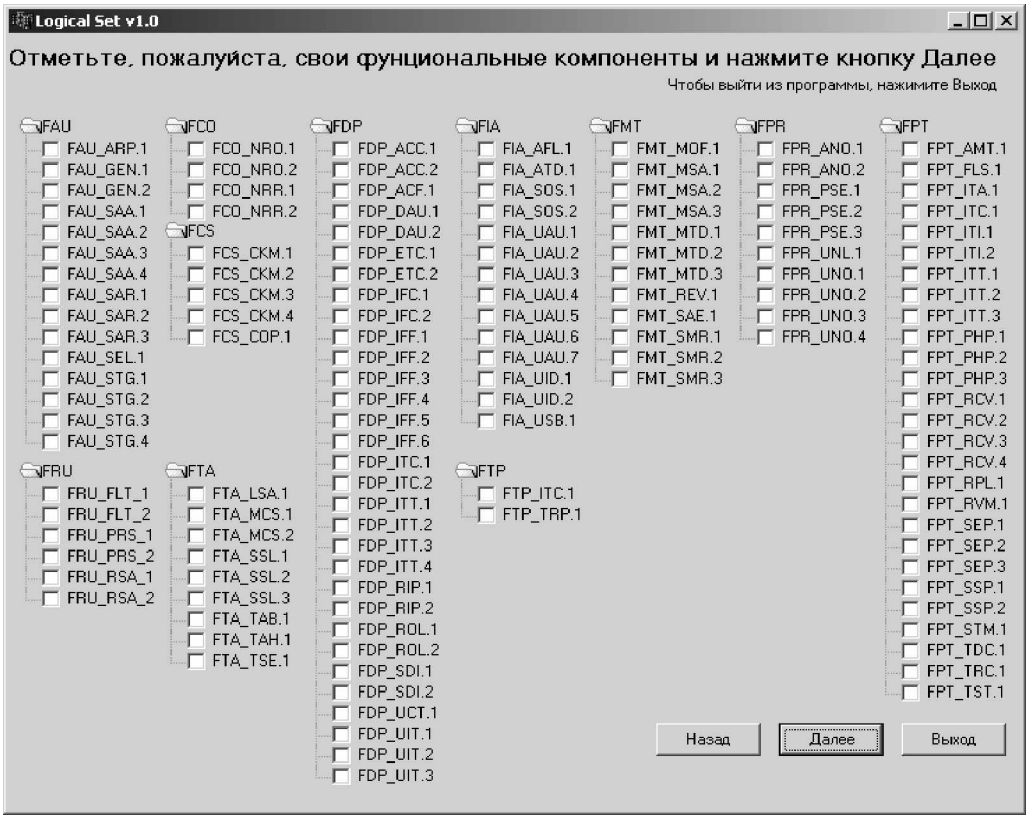


Рис. 1. Добавление новых функциональных компонентов

Рис. 2. Добавление компонентов доверия

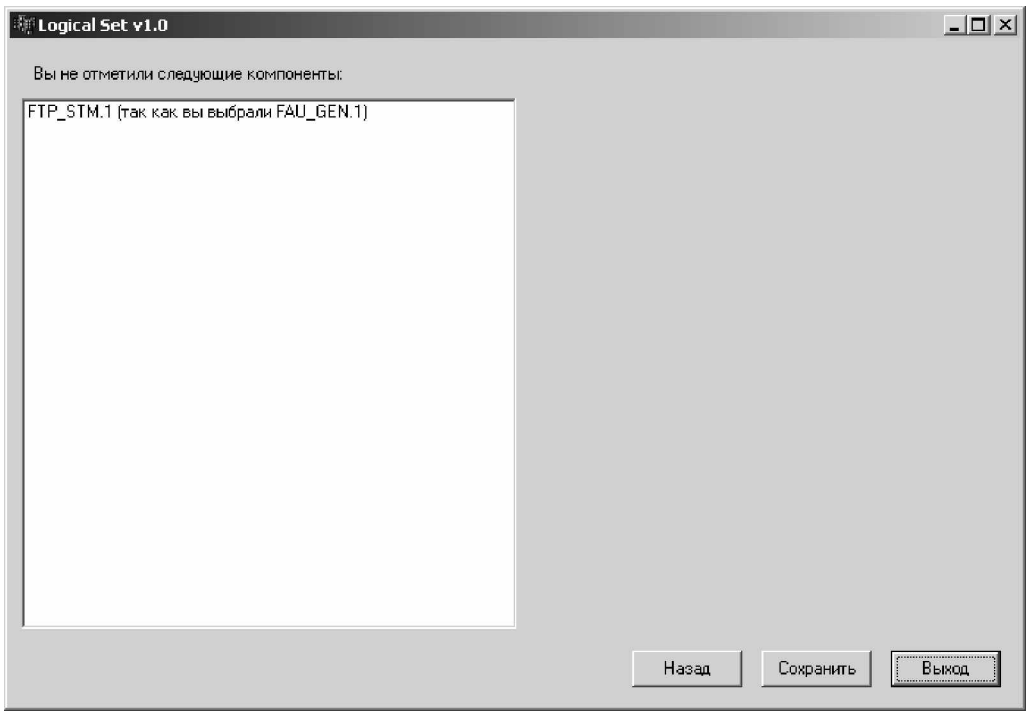


Рис. 3. Определение набора требований безопасности

Литература

1. ГОСТ Р ИСО/МЭК 15408-1. Информационная технология. Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий. Часть 1. Введение и общая модель. Госстандарт России, М., 2002.
2. ГОСТ Р ИСО/МЭК 15408-2 Информационная технология. Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий. Часть 2. Функциональные требования безопасности. Госстандарт России, М., 2002.
3. ГОСТ Р ИСО/МЭК 15408-3 Информационная технология. Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий. Часть 3. Требования доверия к безопасности. Госстандарт России, М., 2002.

ИСПОЛЬЗОВАНИЕ 36 КИТАЙСКИХ СТРАТАГЕМ В СФЕРЕ ИНФОРМАЦИОННОГО ВЗАИМОДЕЙСТВИЯ

А.А. Вяхирев

Научный руководитель – доктор технических наук, профессор Л.Г. Осовецкий

Статья посвящается использованию 36 китайских стратагем в сфере информационного взаимодействия [2]. Разработано формализованное описание данной проблемы в виде компьютерного языка.

Введение

Использование 36 китайских стратагем [1] в сфере информационного взаимодействия позволяет применить данные принципы победы в компьютерных технологиях.

Разработанный язык представляет собой формализованное описание 36 китайских стратагем [3]. Язык представляет собой действие каждого из участников в какой-либо промежуток времени, т.е. описывает пошаговые действия противников [4]. Цель данного языка – описать взаимодействие между противниками при применении стратагем друг к другу. Язык описывает многосубъектовую модель взаимодействия, что и позволяет применять эти принципы в сфере информационного взаимодействия. Рассмотрены картины видимости для каждого из участников. Введено такое понятие, как ресурс участника. При взаимодействии каждый из участников получает определенное количество ресурсов. Составлена таблица побед и поражений при определенном взаимодействии. Общий подсчет ресурсов производится после окончания взаимодействия. Ведется разработка программного комплекса, работающего на данных принципах.

Основная часть

В качестве первого примера рассмотрим стратагему 3: «Воспользовавшись чужим ножом, убить человека» [3].

Как показано в табл. 1, в данной стратагеме субъект s1 втягивает во взаимодействие субъекта s3. s3 взаимодействует с субъектом s2, но при этом s2 не знает, что s3 был втянут во взаимодействия субъектом s1. Тем самым s1 косвенно воздействует на s2, не участвуя в непосредственных взаимодействиях.

3	s1	s2	s3	s3Єs1	s3->s2		S1	S2
s1	+	+	+	+	+		5	3
s2	+	+	+	–	–			

Таблица 1. Стратагема 3

Как видно из таблицы, субъект s1 видит себя, s2 также знает о том, что s3 принадлежит ему и видит то, что s3 взаимодействует с s2 так, как хотелось бы s1. s2 видит s1, себя, s3, но не знает, что s3 принадлежит s1 и что s3 действует по причине, зависящей от s1.

В качестве второго примера рассмотрим стратагему 36: «Бегство – лучший прием» [5].

Здесь s1 знает, что он слабее s2, поэтому уходит от взаимодействия (убегает). В данной стратагеме субъект s1 выходит из игры, тем самым s2 не может применить ответную стратагему относительно s1.

Табл. 2 отражает результаты взаимодействия субъектов, применивших стратагемы.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0	0	0	–1	0	0	0	–1	–1	–1	0	–1	0	0
2	0	0	0	–1	0	0	0	–1	–1	–1	0	–1	0	0
3	0	0	0	–1	0	0	0	–1	–1	–1	0	–1	0	0
4	1	1	1	0	1	1	1	0	0	0	1	–1	1	1
5	0	0	0	–1	0	0	0	–1	–1	–1	0	–1	0	0
6	0	0	0	–1	0	0	0	–1	–1	–1	0	–1	0	0
7	0	0	0	–1	0	0	0	–1	–1	–1	0	–1	0	0
8	1	1	1	0	1	1	1	0	0	0	1	–1	1	1
9	1	1	1	0	1	1	1	0	0	0	1	–1	1	1
10	1	1	1	0	1	1	1	0	0	0	1	–1	1	1
11	0	0	0	–1	0	0	0	–1	–1	–1	0	–1	0	0
12	1	1	1	1	1	1	1	1	1	1	1	0	–1	–1
13	0	0	0	–1	0	0	0	–1	–1	–1	0	1	0	0
14	0	0	0	–1	0	0	0	–1	–1	–1	0	1	0	0
15	–1	–1	–1	1	–1	–1	–1	1	1	1	–1	1	–1	–1

Таблица 2. Общая таблица результатов

Как видно из приведенного фрагмента, табл. 2 дает 3 результата: –1; 0; 1, со следующими значениями:

–1 – субъект s1 проигрывает в результате применения стратагемы;

0 – ни один из субъектов не выигрывает, ничья;

1 – выигрывает первый субъект, s1.

Субъекты действуют пошагово, т.е. сначала субъект s1 применяет свою стратагему относительно субъекта s2, результатом взаимодействия является баллы, набираемые по критерию видимости картины взаимодействия. Затем второй субъект (s2) применяет свою стратагему относительно первого субъекта (s1).

В качестве третьего примера рассмотрим совместное применение двух стратагем: 10 – «В улыбке прятать нож» и 14 – «Занять труп, чтобы вернуть себе жизнь» [5].

В табл. 3 произведен анализ стратагемы 10 «В улыбке прятать нож».

10	s1	s2	obm
s1	+	+	+
s2	+	+	-

Таблица 3. Анализ стратагемы 10

Субъект s2 не видит истинных намерений s1. В данном случае obm является функцией входа в доверие, чтобы ослабить бдительность соперника.

В табл. 4 произведен анализ стратагемы 14 «Занять труп, чтобы вернуть себе жизнь».

14	s1	s2	s3	s3Єs1
s1	+	+	+	+
s2	-	+	+	-

Таблица 4. Анализ стратагемы 14

Субъект s1 вводит нового субъекта s3, но s2 не знает, что s1 u s3 – одно и то же лицо. Таким образом, s1 взаимодействует с s2 через s3.

Таблицы построены на основе видимых картин.

Субъект s1 применяет стратагему №10. В результате ему начисляется 3 балла, а s2 – 2 балла. Субъект s2 применяет свою стратагему, №14. В результате ему начисляется 4 балла, а субъекту s1 – 2. Следовательно,

$S= \Sigma s1 – \Sigma s2 = –1$.

Заключение

Реализация защиты информации связана с планированием некоторой системы мероприятий, а значит, и с расходованием ресурсов. В зависимости от того, насколько хорошо или плохо спланированы эти мероприятия, настолько же малы или велики будут и затраты ресурсов. На практике не менее широко распространена и другая постановка задачи, когда при заданном объеме ресурсов требуется так спланировать систему мероприятий по его реализации, чтобы обеспечить наибольшую степень достижения цели.

Трудности возрастают, если реализация планируемой системы мероприятий встречает организованное противодействие некоторой другой стороны, которая преследует цели, не совпадающие, а часто и прямо противоположные целям первой. Планирующей стороне при распределении своих ресурсов в этих условиях потребуется учитывать возможное противодействие разумного противника и степень его информированности. Разработанный компьютерный язык позволяет смоделировать огромное количество конфликтных ситуаций многосубъектового взаимодействия, учитывая множество параметров, необходимых для достижения полнейшей картины взаимодействия.

Литература

1. Харро фон Зенгер. Стратагемы. М.: ЭКСМО, 2004. В 2-х т.
2. Расторгуев С.П. Введение в формальную теорию информационной войны. М.: Вузовская книга, 2002.
3. Воеводин А.И. Стратагемы – стратегии войны, манипуляции, обмана. М.: ИД Эт Сетера Паблишинг, 2004.
4. Лефевр В.А. Конфликтующие структуры. М.: Советское радио, 1973.
5. Малявин В.И. Тридцать шесть стратагем. Китайские секреты успеха. М.: Белые Альвы, 2000.

ОЦЕНКА ЗАЩИЩЕННОСТИ КОМПЛЕКСА УПРАВЛЕНИЯ ПОДВИЖНЫМИ ОБЪЕКТАМИ НА БАЗЕ ИГРОВОЙ МОДЕЛИ

С.В. Егоров

Научный руководитель – доктор технических наук, профессор Ю.А. Гатчин

В работе представлен расчет информационной защищенности комплекса управления подвижными объектами. Этот расчет основывается на игровой модели процессов защиты и показателях прочности отдельных средств защиты возможных каналов несанкционированного доступа к информации.

Введение

При повсеместном внедрении новых информационных технологий и увеличении их влияния на жизнь общества особо важную роль приобретает проблема информационной безопасности.

Для построения надежной информационной защиты необходимо выявить возможные угрозы информационной безопасности и определить меры и средства защиты, но все эти средства могут крайне неэффективно функционировать, если не проводить соответствующую оценку прочности информационной защиты [1, 2]. Острота и актуальность проблемы оценки информационной защищенности обусловлена тем, что сегодня не существует универсальной методики оценки эффективности системы защиты информации (СЗИ) [1]. Формирование научно-методологической базы для решения проблемы защиты сопряжено с большими трудностями, связанными с необходимостью учитывать подверженность процессов защиты влиянию трудно предсказуемых злоумышленных действий людей [3]. Поэтому множество предлагаемых формальных моделей СЗИ оказалось неприемлемым для адекватного отображения процессов защиты. Применение теории вероятностей и теории множеств дает субъективный результат, а модели на основе теории марковских цепей связаны с существенными ограничениями (каждая попытка вскрытия защиты не связана с предыдущими), что исключает их практическое использование [1].

В данной работе ставится задача, используя основы теории игр, составить алгоритм оценки информационной защищенности комплекса управления подвижными объектами (КУПО) и вывести оценку обобщенного показателя защищенности КУПО от несанкционированного доступа (НСД). В работе использовались данные эффективности средств защиты для каналов несанкционированного доступа в КУПО, полученные в работах [4, 5].

1. Объект исследования

В качестве объекта исследования выступает комплекс управления подвижными объектами. Этот комплекс представляет собой автоматизированную систему управления с распределенной обработкой данных и имеющий открытый радиоканал управления подвижным объектом. Схема комплекса изображена на рисунке.

2. Игровая модель защиты информации

Процесс защиты связан с конфликтом между стороной, которая обеспечивает безопасность информации, и стороной, желающей незаконно получить ее. Очевидно, конфликт носит антагонистический характер, т.е. цели одной из сторон строго противоположны целям другой. Для анализа таких конфликтов лучше всего подходит теория игр, так как она позволяет моделировать действия обеих сторон [1]. Теория игр –

это теория математических моделей принятия оптимальных решений в условиях конфликтов. Она дает возможность получить стратегию рационального поведения для достижения максимального выигрыша или максимальной вероятности выигрыша [6, 7].

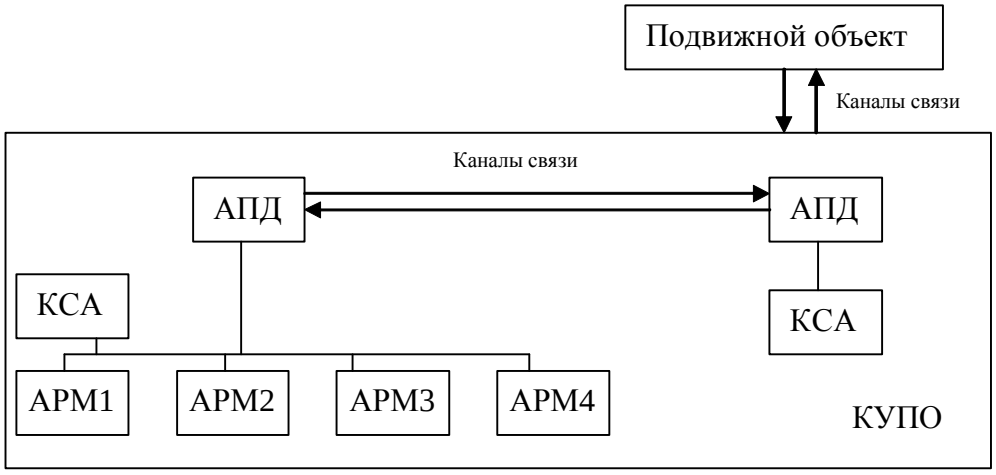


Рис. Схема комплекса управления подвижными объектами: АПД – аппаратура передачи данных; КСА – комплекс средств автоматизации; АРМ – автоматизированное рабочее место

В зависимости от ценности информации к системе защиты могут предъявляться различные требования. При этом возможны две принципиально различные ситуации, обусловливающие необходимость решения соответственно первой или второй задачи. Первая ситуация предполагает, что информация является конфиденциальной или составляет коммерческую тайну. В этом случае следствием потери информации для ее владельца будет экономический ущерб, который можно оценить количественно. Тогда назначение СЗИ состоит в том, чтобы обеспечить уровень защиты, который можно преодолеть только при затратах со стороны злоумышленника, сравнимых со стоимостью самой СЗИ и размером ущерба от потери информации. Задача оптимизации состоит в том, чтобы при заданном размере затрат на СЗИ обеспечить максимальный уровень защиты.

Вторая ситуация возникает, когда информация является государственной или военной тайной, и невозможно оценить стоимость ущерба от ее потери. Система защиты при этом должна обеспечить требуемый уровень безопасности информации (как правило, очень высокий, близкий к 1), а оптимизация состоит в минимизации затрат ресурсов для обеспечения заданного уровня защиты. В данной работе рассматривается КУПО, который относится к автоматизированным системам военного назначения (АС ВН).

Естественно предположить, что для организации защиты информации пользователь заранее проанализировал все возможные в его ситуации угрозы информации и выбрал защитные средства и организационные мероприятия таким образом, чтобы они перекрывали все угрозы. Тогда можно определить конечное число угроз и реакций на них системы защиты. Такие угрозы и средства защиты информации в КУПО выявлены и перечислены в работах [4, 5].

Главное достоинство теории игр при решении проблемы защиты – в том, что она определяет стратегию рационального поведения как стороны, обеспечивающей безопасность информации, так и злоумышленника. Анализ оптимальной стратегии последнего дает возможность вскрыть недостатки системы защиты и улучшить ее. Такая модель относится к классу игр с неполной информацией [1].

3. Оцениваемые показатели и расчетные соотношения

В результате анализа технического задания, программной и эксплуатационной документации на конкретный вариант исполнения экспертами определяются следующие характеристики: l – количество объектов КУПО; m – количество защитных механизмов, используемых в тестируемой КУПО; n – количество угроз НСД к объектам КУПО; $\mu_{ik} \in [0;1]$, i =1,…n – коэффициенты опасности угроз НСД для каждого k объекта КУПО; $k = 1,....,l$; $\gamma_{ijk} \in]0;1]$, j =1,...,m – коэффициент эффективности использования механизма защиты j от i-й угрозы НСД к объектам КУПО; a_{ik} – затраты на реализацию угроз на объектах КУПО; b_{jk} – стоимость средств защиты на объектах КУПО; C_k – информационная ценность объектов КУПО.

Используя материалы [1, 4, 5], можно построить платежную матрицу, описывающую возможные информационные воздействия на КУПО при использовании различных вариантов применяемых в тестируемой автоматизированной системы мер и средств защиты информации.

$$\varphi_{i\theta}^{k} = \frac{C_k \mu_{ik}}{a_{ik}}[1-(1-\gamma_{i\theta k})\frac{B}{lb_{\theta k}}], \qquad (1)$$

$$i = 1,....,n, \qquad \theta = 1,....,w, \; w = 2^m - 1,$$

где $\varphi_{i\theta}^{k}$ – элемент платежной матрицы для объекта k ; $\gamma_{i\theta k} = \prod_{j=1}^{m} \gamma_{ijk} h_{j\theta} \quad \forall h_{j\theta} = 1$; $h_{i\theta}$ – коэффициент использования механизма защиты j в варианте СЗИ θ ; $b_{\theta k} = \sum_{j=1}^{m} b_{jk} h_{j\theta}$; B – суммарная стоимость СЗИ тестируемого КУПО.

Тогда обобщенный показатель защищенности объекта КУПО от НСД Q_{θ}^{k} определяется выражением

$$Q_{\theta}^{k} = \frac{V_{эт}^{k}}{V_{\theta}^{k}}, \qquad (2)$$

где $V_{эт}^{\kappa}$ – значение цены игры, задаваемой формулой (1), в смешанных стратегиях; V_{θ}^{k} – значение цены игры для используемого в КУПО варианта СЗИ θ .

Показатель защищенности КУПО от НСД $Q_{КУПО}$ рассчитывается по формуле

$$Q_{КУПО} = \frac{\sum_{k=1}^{l} Q_{\theta}^{k}}{l}. \qquad (3)$$

Значения показателей μ_{ik} , γ_{ijk} , a_{ik} , b_{jk} , C_k предлагается определять экспертно на основе метода прямого установления предпочтений путем попарных сравнений [1].

Затем в соответствии с табл. определяется класс защищенности представленной на испытания АС ВН.

Решение о соответствии КУПО требованиям по защищенности от НСД принимается на основе сравнения результатов рассчитанного класса защищенности с требованиями, определенными в программе испытаний.

Данная модель расчета информационной защищенности была применена для расчета обобщенного показателя защищенности КУПО, используя поле угроз, характерных для информационной модели КУПО [4, 5]. Получено значение 0,74, что соответствует классу защищенности 1Б.

Класс защищенности КУПО	Значение показателя защищенности КУПО
1А	0.84–1.0
1Б	0.68–0.84
1В	0.52–0.68
1Г	0.36–0.52
1Д	0.20–0.36

Таблица. Требования по защищенности КУПО от НСД

Очевидно, что со временем у злоумышленника будут появляться все более совершенные средства для взлома СЗИ, и постепенно он будет приобретать какой-то опыт по вскрытию защиты, как бы она ни изменялась и ни модифицировалась. Поэтому важнейшей характеристикой, не отслеживаемой известными формальными моделями, является изменение эффективности защиты с течением времени. В теории игр эту проблему можно решить, моделируя игры с учетом обесценивания информации. При этом вводится коэффициент $R[0,1]$, являющийся функцией зависимости ценности информации от времени. При оценке эффективности защиты в разные моменты времени достаточно умножить элементы матрицы на значение R, соответствующее этому моменту времени, и снова решить задачу. Такой метод выгодно отличается наглядностью и объективностью.

Заключение

В работе представлен алгоритм оценки системы защиты информации для комплекса управления подвижными объектами на основе игровой модели системы защиты информации.

По результатам экспертной оценки прочности защиты каждого средства защиты информации для соответствующих угроз безопасности информации проведен расчет и получен обобщенный показатель защищенности всей системы защиты. Полученный результат соответствует классу защищенности 1Б.

Работа позволяет оценить информационную защищенность комплекса управления подвижными объектами и может послужить основанием для внесения изменений в перечень применяемых средств защиты с целью повышения информационной защиты охраняемого объекта. Достоинством метода является возможность внесения изменений в перечень применяемых средств защиты на основании, как требуемого показателя защищенности, так и стоимостного ограничения.

Литература

1. Воробьев А.А. Анализ моделей процессов защиты информации от несанкционированного доступа в автоматизированных системах. // Информатика – машиностроение. 1999. №2. С. 32–34.
2. Мельников В.В. Безопасность информации в автоматизированных системах. М.: Финансы и статистика, 2003. 368 с.

3. Воробьев А.А. К вопросу оценивания качества системы защиты информации. // Проблемные вопросы сбора, обработки и передачи информации в сложных радиотехнических системах. Тез. докл. третьей межведомственной научно-технической конференции. СПб.: Пушкин, ВУРЭ ПВО, 1997.
4. Егоров С.В. Каналы несанкционированного доступа и средства их защиты в комплексах управления подвижными объектами. // Научно-технический вестник СПбГУ ИТМО. 2005. №20. С. 181–185.
5. Егоров С.В. Расчет информационной прочности в комплексах управления подвижными объектами. // Сборник научных трудов II конференции молодых ученых. СПб.: СПбГУ ИТМО, 2005.
6. Оуэн Г. Теория игр./ Пер. с англ. М.: Мир, 1971.
7. Шишкин Е.В. От игр к играм. Математическое введение. Изд. 2-е, исправл. М.: Едиториал УРСС, 2003. 112 с.

К РАЗРАБОТКЕ АЛГОРИТМА АКТИВНОГО ТЕСТИРОВАНИЯ КОРПОРАТИВНОЙ СЕТИ

Д.В. Звонов, Ф.Г. Нестерук
Научный руководитель – доктор технических наук, профессор Л.Г. Осовецкий

Предложен алгоритм активного тестирования информационной сети с использованием адаптивных средств, базирующийся на основных свойствах систем нечеткой логики, нейронных сетей и экспертных систем, связанных с адаптацией и возможностью представления опыта специалистов информационной безопасности в виде системы правил логического вывода.

Введение

Организация опыта квалифицированных специалистов информационной безопасности в виде базы знаний экспертной системы, представляемой системой правил нечеткого логического вывода, и их последующее отображение в виде нейро-нечетких сетей (ННС), входящих в состав классификаторов системы активного тестирования (САТ) позволяет, с одной стороны, объяснить результаты работы системы активного тестирования путем анализ информационных полей ННС, с другой – передавать опыт, накопленный в процессе эксплуатации корпоративные сети аналогичного назначения, путем наследования – перезаписи информационных полей нейро-нечетких сетей [1].

Адаптация к изменению множества уязвимостей корпоративной сети и поля угроз информационной безопасности рассматривается как одно из наиболее важных свойств интеллектуальных систем, которое позволяет корректировать средства активного тестирования при изменении входной информации и внешнего окружения. Обучающими факторами являются избыточность входной информации и скрытые в данных закономерности, которые видоизменяют информационное поле нейро-нечетких сетей в процессе обучения адаптивных классификаторов САТ [1, 2]. Необходимо автоматизировать процесс активного тестирования и коррекции структуры системы защиты информации (СЗИ).

Автоматизация обеспечения информационной безопасности

Для уменьшения вероятности несанкционированного доступа в корпоративную сеть акционерного коммерческого банка (АКБ) в последнее время используют средства автоматизации управления системами защиты информации, к которым относят корреляторы событий, системы обновлений, средства ЗА (аутентификации, авторизации и администрирования) и системы управления рисками [3, 4].

Корреляторы событий *по результатам анализа* системных журналов выделяют признаки атак, системы обновления автоматизируют процедуру оперативного устранения выявленных ошибок и поиска скрытых уязвимостей программного обеспечения (ПО). Средства ЗА позволяют управлять идентификационной информацией и допуском пользователей к информационным ресурсам, а системы управления рисками – моделировать и оценивать ожидаемый ущерб атаки на корпоративную сеть.

Перечисленные задачи с успехом решаются с использованием интеллектуальных средств нейронных сетей, нечеткой логики и генетических алгоритмов.

Общей чертой существующих систем защиты информации является наличие отдельных средств идентификации (задача классификации) угроз, выявления и устранения уязвимостей и оперативной реакции на несанкционированные проникновения в корпоративную сеть, а общим недостатком – отсутствие в СЗИ средств, обеспечивающих накопление и обобщение опыта взаимодействия корпоративной сети с внешней средой и нейтрализации внутрисистемных угроз.

Для автоматизации обеспечения информационной безопасности корпоративной сети необходим комплексный подход на основе иерархически организованной СЗИ с применением интеллектуальных средств автоматической идентификации угроз и выявления уязвимостей, а также накопления опыта нейтрализации атак и устранения уязвимостей корпоративной сети АКБ.

Для автоматизации выявления и устранения уязвимостей необходима оценка защищенности корпоративной информационной сети, используемая в качестве целевой функции процесса оптимизации системы активного тестирования. Подобной оценкой защищенности корпоративной информационной сети может являться рейтинг защищенности, который учитывает структуру системы защиты информации [5, 6], качественные показатели работы СЗИ [7] и оценки предотвращенного ущерба [8].

В случае изменения поля угроз в процессе эксплуатации защищаемой корпоративной информационной сети может проявляться ряд новых уязвимостей, не отраженных в исходной модели, и, соответственно, потенциальная возможность реализации новых угроз безопасности информационным ресурсам АКБ. В этой связи целесообразно рассматривать модель системы информационной безопасности в динамике, начиная с начального этапа жизненного цикла системы.

Для оптимизации структуры системы активного тестирования разработан алгоритм тестирования информационной сети АКБ.

Алгоритм активного тестирования информационной сети АКБ

Алгоритм активного тестирования информационной сети АКБ содержит следующую последовательность этапов:

1. решение задачи классификации угроз информационным ресурсам АКБ по вектору известных уязвимостей корпоративной сети; производится соотнесение посылок (вектора известных уязвимостей) с классификационными заключениями (угрозами информационным ресурсам);
2. решение задачи кластеризации угроз информационным ресурсам АКБ по вектору известных уязвимостей корпоративной сети при расширении множества известных уязвимостей корпоративной сети; производится разбиение входных векторов на группы (векторов уязвимостей корпоративной сети) и отнесение вновь поступающего входного вектора к одной из групп либо формирование новой группы (группы уязвимостей корпоративной сети);
3. формирование экспертных оценок для определения степени соответствия угроз информационным ресурсам АКБ векторам известных уязвимостей корпоративной сети;
4. представление результатов решения задач п. 1 и 3 в виде систем правил нечеткого логического вывода;
5. реализация систем правил нечеткого логического вывода в виде нейро-нечетких классификаторов (классификаторов «уязвимости – угрозы»);
6. реализация результатов решения задачи п.2 в виде кластеризаторов на основе самообучающейся НС (кластеризаторов «уязвимости – угрозы»);
7. наследование или передача опыта нейтрализации уязвимостей, приобретенного в процессе эксплуатации САТ корпоративной сети, в проектируемую систему активного тестирования путем перенесения информационных полей четких и нейронечетких сетей;
8. обучение классификаторов по п. 5, 6 на обучающей выборке – подмножестве входных векторов (векторов известных уязвимостей) с целью формирования информационных полей четких и нейро-нечетких сетей;

9. адаптация информационных полей четких и нейро-нечетких сетей (классификаторов и кластеризаторов «уязвимости – угрозы») в процессе эксплуатации информационной сети АКБ;
10. коррекция адаптируемых экспертных оценок (п. 3) и систем правил нечеткого логического вывода (п. 4) по результатам адаптации информационных полей нейро-нечетких сетей;
11. формулирование новых правил нечеткого логического вывода в случае расширения классификации (кластеризации) по результатам выполнения п. 9 и 2;
12. формирование оценок защищенности информационной сети АКБ, исходя из результатов выполнения п.10 и распределения механизмов защиты по иерархии системы активного тестирования корпоративной сети;
13. анализ структуры связей нейро-нечетких классификаторов, системы правил нечеткого логического вывода и комплекса оценок защищенности информационной сети АКБ по п. 12 для определения интенсивности использования МЗ в структуре САТ;
14. формирование спецификации на разработку отсутствующих механизмов защиты;
15. коррекция системы активного тестирования корпоративной сети за счет расширения перечня используемых механизмов защиты и их размещения в адаптивной САТ;
16. повторение этапов п. 9–15 в процессе эксплуатации САТ для регулярного обновления базы знаний системы активного тестирования корпоративной сети и накопления опыта по устранению уязвимостей информационной сети АКБ.

Методика построения политики безопасности информационной сети АКБ

В ходе проведения оптимизации корпоративной сети АКБ в соответствии с алгоритмом активного тестирования корпоративной сети происходит выявление ранее неизвестных уязвимостей. Необходимо предпринимать оперативные меры и проводить мероприятия по модернизации СЗИ, направленные на своевременное устранение выявленных уязвимостей.

Критерием необходимости модернизации СЗИ, с одной стороны, являются расчетные показатели качества обеспечения информационной безопасности КИС (рейтинг защищенности корпоративной сети АКБ) [6], а, с другой – такие показатели экономической целесообразности модификации СЗИ, как чистая приведенная стоимость, индекс рентабельности и срок окупаемости инвестиций [9].

Рисунок иллюстрирует основные этапы методики построения политики безопасности, которая позволяет сформулировать практические рекомендации по внесению изменений в действующую политику безопасности АКБ.

Заключение

Предложенный алгоритм за счет регулярного тестирования корпоративной сети АКБ производит оптимизацию размещения механизмов защиты в структуре системы активного тестирования по критерию максимизации уровня защищенности. Наличие адаптивных средств позволяет циклическим выполнением процедур обучения и последующего анализа информационных полей нейро-нечетких сетей обеспечить постоянный мониторинг уязвимостей корпоративной сети АКБ.

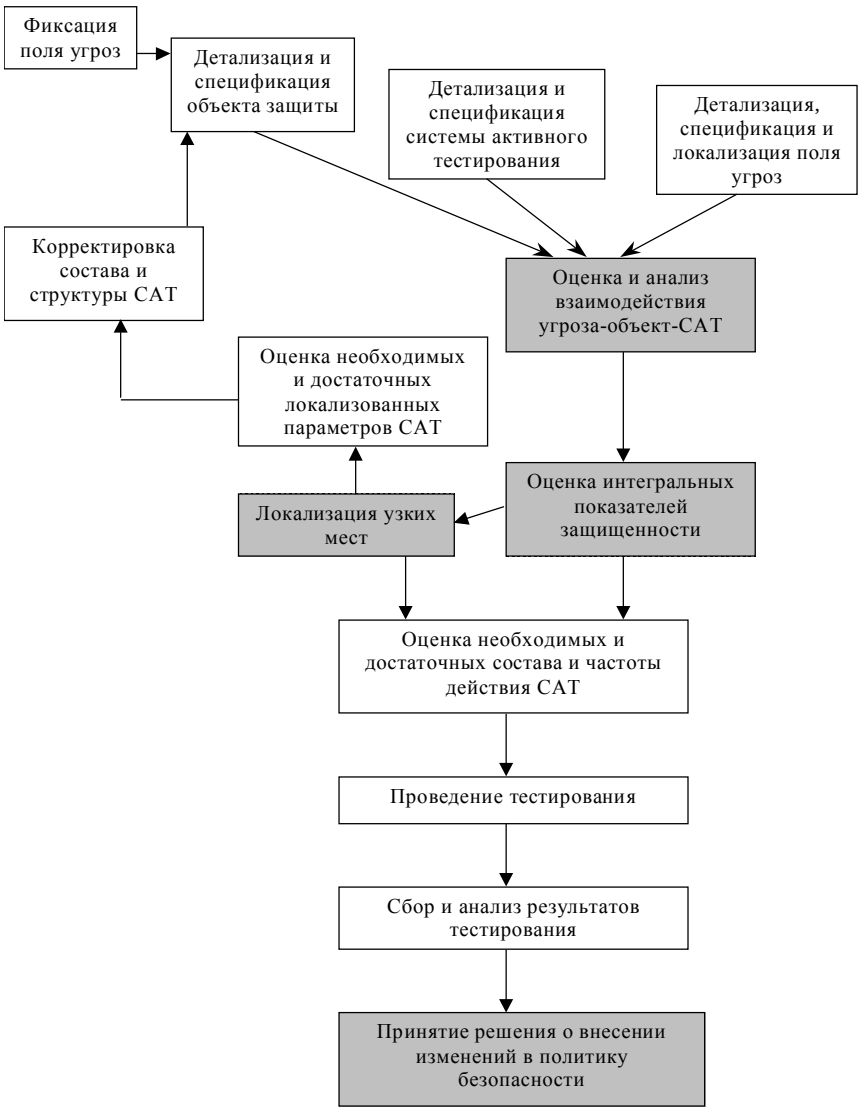


Рис. Основные этапы методики построения политики безопасности

Литература

1. Нестерук Г.Ф., Осовецкий Л.Г., Нестерук Ф.Г. О применении нейро-нечетких сетей в адаптивных системах информационной защиты. // Нейроинформатика-2005: Материалы VII всероссийской научно-технической конференции. М.: МИФИ (ТУ), 2005. Ч.1. С. 163–171.
2. Negnevitsky M. Artificial intelligence: a guide to intelligent systems. // Addison-Wesley, 2002. 394 p.
3. Коржов В. Моделирование угроз. // Computerword Россия. 2005, № 18. С. 34.
4. Звонов Д.В. Системы активного тестирования при построении комплексных систем обеспечения безопасности корпоративных сетевых технологий. // Материалы 2-го международного научно-практического семинара «Безопасные информационные технологии – XXI». СПб, 2000.
5. Осовецкий Л., Шевченко В. Оценка защищенности сетей и систем. // Экспресс-электроника. 2002. № 2–3. С. 20–24.

6. Нестерук Ф.Г., Осовецкий Л.Г., Штрик А.А., Жигулин Г.П. К оценке информационных ресурсов и безопасности глобальных компьютерных систем. // Перспективные информационные технологии и интеллектуальные системы. 2004. № 4.
7. Карпычев В.Ю., Минаев В.А. Цена информационной безопасности. // Системы безопасности. 2003. № 5. С. 128–130.
8. Жижелев А.В., Панфилов А.П., Язов Ю.К., Батищев Р.В. К оценке эффективности защиты информации в телекоммуникационных системах посредством нечетких множеств // Изв. вузов. Приборостроение. 2003. Т. 46. № 7. С. 22–29.
9. Инвестиции: Учебник / Под ред. В.В. Ковалева, В.В. Иванова, В.А. Лялина // М.: ТК Велби, 2003. 440 с.

О МОДЕЛИРОВАНИИ АДАПТИВНЫХ СРЕДСТВ АКТИВНОГО ТЕСТИРОВАНИЯ КОРПОРАТИВНОЙ СЕТИ

В.О. Доскач, Д.В. Звонов, И.М. Инюшин, Р.В. Леонтьев
Научный руководитель – кандидат технических наук, доцент Г.Ф. Нестерук

В работе рассматривается диалоговая программная среда и результаты компьютерного моделирования процессов обучения нейросетевых классификаторов адаптивных уровней системы активного тестирования (САТ), полученные с помощью диалоговой программной среды моделирования САТ корпоративной сети. Среда используется в качестве лабораторного практикума для студентов кафедры «Безопасные информационные технологии».

Введение

Средства сканирования и активного тестирования получили широкое распространение в комплексных системах информационной безопасности (СИБ) корпоративных сетей [1, 2]. Для обеспечения оперативной реакции на изменение поля угроз и высокого уровня защищенности корпоративных сетей используются адаптивные средства защиты информации [3]. Адаптивность СИБ, как правило, связана с применением нейронных сетей (НС) в составе средств защиты информации [4]. Нейросетевые средства защиты информации позволяют автоматически реагировать на изменение поля угроз и накапливать в информационных полях НС опыт по нейтрализации атак на корпоративную сеть [3, 4]. Для обеспечения возможности анализа опыта по нейтрализации атак, накопленного в информационных полях НС, в качестве нейросетевых средств защиты необходимо использовать нейро-нечеткие сети (ННС) [5, 6].

Рассматриваемые в работе программные средства моделирования процесса обучения нейро-нечетких сетей в составе системы активного тестирования (САТ) корпоративной сети позволяют:

- знания экспертов информационной безопасности (исходную базу знаний) представить в виде системы правил логического вывода IF-THEN,
- систему правил логического вывода отразить в структуре нейро-нечеткой сети,
- обучить нейро-нечеткую сеть на заданном подмножестве пар векторов обучающей выборки посредством выбранного алгоритма обучения НС,
- провести анализ информационного поля нейро-нечеткой сети после завершения процесса обучения,
- откорректировать исходную базу знаний экспертов информационной безопасности (ИБ).

Исходя из базы знаний экспертов информационной безопасности, представленной в виде системы правил нечеткого логического вывода, программные средства автоматически формируют топологию нейро-нечеткого классификатора. Таким образом, база знаний экспертов ИБ отражается в структуре нейро-нечеткого классификатора (рис. 1). Обучение нейро-нечеткого классификатора на обучающей выборке изменяет информационное поле нейро-нечеткой сети (матрицу весов связей ННС), анализ которого позволяет судить о корректности исходной базы знаний.

Диалоговая среда моделирования адаптивных средств САТ

В среде Java™ 2 Platform разработана диалоговая среда (далее – программа), которая позволяет моделировать работу системы активного тестирования. Адаптивные уровни уязвимостей и угроз в составе модели системы активного тестирования [3] используются для решения задач:

- классификации уязвимостей и классификации известных угроз информационной безопасности корпоративной сети,

- адаптации структуры нейро-нечетких классификаторов к изменению выявленных уязвимостей и расширению поля угроз.

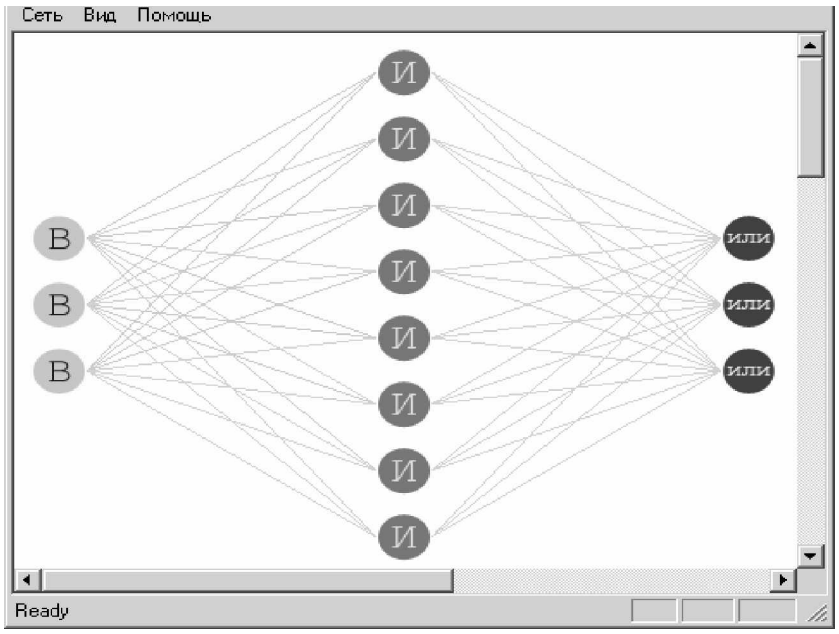


Рис. 1. Окно программных средств для визуализации топологии НС

После активации программы открывается главное окно, предназначенное для организации интерфейса с пользователем в процессе формирования и обучения нейросетевых средств системы активного тестирования. При запуске программы автоматически открывается вкладка **model,** предлагающая загрузить файл с топологией исследуемой нейронной сети (кнопка **Load model**).

Файл с топологией нейро-нечеткой сети создается с помощью редактора топологии НС, который формируется по системе правил нечеткого логического вывода. Результат работы редактора топологии НС представлен на рис. 2 в виде трехслойной структуры, содержащей узлы В, которые образуют входной слой нейро-нечеткой сети, узлы И – слой логического вывода ННС, соответствующий этапу процедуры нечеткого логического вывода, узлы ИЛИ – слой композиции нейро-нечеткой сети.

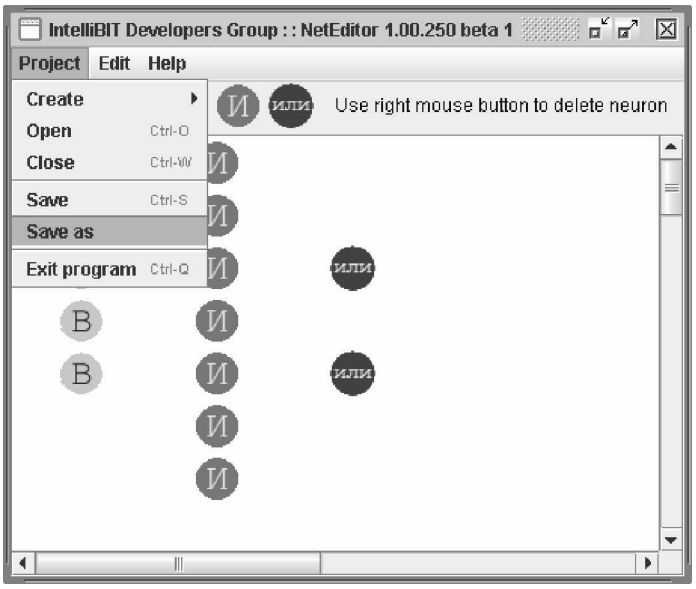


Рис. 2. Окно редактора топологии НС

Редактор топологии НС позволяет создавать нейро-нечеткую сеть (меню **Project - Create**) путем помещения требуемого числа узлов в соответствующие слои ННС. Для сохранения топологии нейро-нечеткой сети в главном меню редактора следует выбрать меню **Project- Save** или **Save as** (рис. 2).

В окне сохранения топологии нейро-нечеткой сети необходимо задать имя файла с расширением *.nns. Следует также отметить, что с помощью редактора топологии НС можно осуществить коррекцию существующей структуры нейро-нечеткой сети, ранее сохраненной в файле с расширением *.nns. Для этого достаточно через меню **Project - Open** выбрать в открывшемся окне файл с расширением *.nns.

Для загрузки ранее созданной топологии нейро-нечеткой сети следует активировать кнопку **Load model** и в появившемся окне выбрать файл топологии НС, например, Fuzzy_Net_1.nns.

Для обучения нейро-нечеткой сети следует загрузить файл обучающей выборки, созданный заранее текстовым редактором и сохраненный в формате *.txt. Для этого необходимо выбрать вкладку **input** главного окна программы. На вкладке открывается таблица, содержащая перечень входов и выходов. Нажатие на кнопку **import data** приводит к открытию окна, в котором следует выбрать файл с импортируемыми данными, например, train.txt. В таблице, расположенной на вкладке **input**, отражаются значения данных обучающей выборки, которые используются для обучения ранее загруженной нейронной сети (рис. 3).

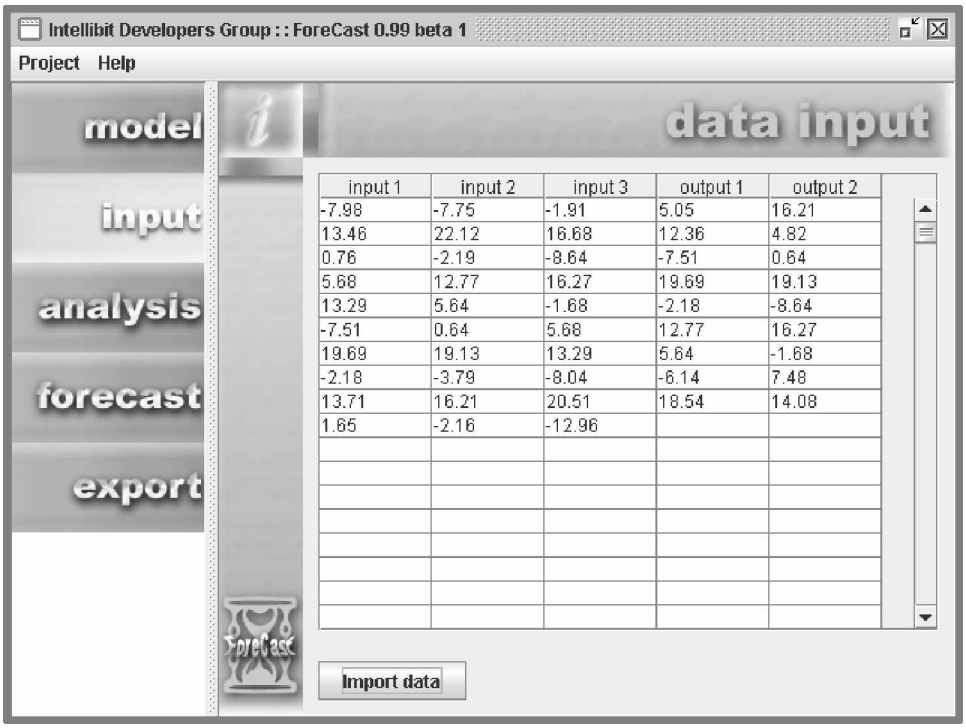


Рис. 3. Вкладка **input** главного окна программы

Для проведения обучения исследуемого варианта топологии нейронной сети необходимо выбрать вкладку **analysis** и запустить процесс адаптации нажатием на кнопку **Start**. Процесс обучения НС сопровождается выводом сообщений о создании (**Creating of Neural Network**) и обучении нейронной сети (**Teaching of Neural Network**), а также отражением прогресса обучения нейронной сети.

После завершения обучения нейронной сети на вкладке **analysis** появится сообщение **Ready**. Вкладка **forecast** позволяет протестировать обученную нейронную сеть.

Качество адаптационных процессов в нейронной сети можно оценить с помощью графических зависимостей, отражающих динамику ошибки обучения НС в зависимости от количества эпох обучения (в случае использования метода обратного распростране-

ния ошибки) или количества поколений нейронных сетей при использовании метода генетических алгоритмов (ГА).

Для сохранения результатов обучения исследуемой нейронной сети следует выбрать вкладку **Export** главного окна программы, выбрать формат представления результатов (например, в виде web-страницы) и нажатием кнопки **Export** инициировать появление окна сохранения результатов исследования НС.

Моделирование адаптивных уровней САТ

Были проведены исследования процессов обучения нейро-нечеткого классификатора при различном сочетании числа формальных нейронов в слоях логического вывода и композиции нейро-нечеткой сети.

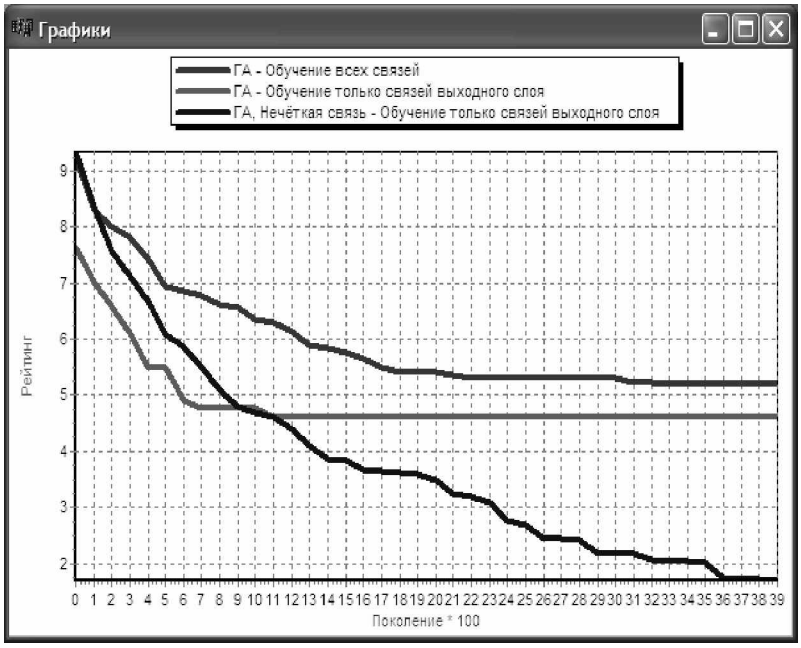


Рис. 4. Окно динамики ошибки обучения нейро-нечеткого классификатора

В качестве иллюстрации на рис. 4 приведена динамика ошибки обучения нейро-нечеткой сети 5-32-8 (5 входов, 32 формальных нейрона в слое логического вывода и 8 выходов), обучаемой по методу генетических алгоритмов.

Проведение эволюционного отбора (по методу рулетки) в популяции нейро-нечетких сетей по критерию минимизации среднеквадратичного отклонения, включение в последующие популяции решений нейро-нечетких сетей с лучшим сочетанием генов в хромосоме приводит к устойчивому снижению ошибки обучения нейро-нечеткого классификатора при генерации произвольной начальной комбинации из случайных значений достоверности межнейронных связей классификатора.

Из приведенных результатов компьютерного моделирования следует, что генетические алгоритмы являются наиболее скоростным методом адаптации нейросетевых классификаторов для системы активного тестирования корпоративной сети.

Заключение

В среде Java™ 2 Platform разработана диалоговая среда для моделирования адаптивных уровней системы активного тестирования корпоративной сети. Диалоговая программная среда позволяет моделировать работу адаптивных уровней САТ, которые в составе модели системы активного тестирования используются для решения задач классификации уязвимостей и классификации известных угроз информационной безо-

пасности корпоративной сети, а также адаптации структуры нейро-нечетких классификаторов к изменению выявленных уязвимостей и расширению поля угроз.

Проведено компьютерное моделирование процессов обучения нейронных и нейро-нечетких сетей в составе адаптивных уровней системы активного тестирования с использованием алгоритмов обратного распространения ошибки и метода генетических алгоритмов. Показано, что генетические алгоритмы дают лучшие результаты по критерию динамики снижения ошибки обучения информационных полей нейросетевых классификаторов для САТ.

Литература

1. Звонов Д.В. Системы активного тестирования при построении комплексных систем обеспечения безопасности корпоративных сетевых технологий // Материалы 2-го международного научно-практического семинара «Безопасные информационные технологии – XXI». СПб, 2000.
2. Обзор сетевых сканеров (http://www.3dnews.ru/reviews/software/netscan/).
3. Нестерук Г.Ф., Осовецкий Л.Г., Нестерук Ф.Г. О применении нейро-нечетких сетей в адаптивных системах информационной защиты. // Нейроинформатика-2005: Материалы VII всероссийской научно-технической конференции. М.: МИФИ, 2005. Ч.1. С. 163–171.
4. Нестерук Г.Ф., Осовецкий Л.Г., Харченко А.Ф. Информационная безопасность и интеллектуальные средства защиты информационных ресурсов. (Иммунология систем информационных технологий). СПб.: Изд-во СПбГУЭФ, 2003. 364 с.
5. Negnevitsky M. Artificial intelligence: a guide to intelligent systems. Addison-Wesley, 2002. 394 p.
6. Нестерук Г.Ф., Молдовян А.А., Костин А.А., Нестерук Ф.Г., Воскресенский С.И. Организация иерархической защиты информации на основе интеллектуальных средств нейро-нечеткой классификации. // Вопросы защиты информации. 2005. № 4.

К ОЦЕНКЕ УРОВНЯ ЗАЩИЩЕННОСТИ КОРПОРАТИВНОЙ СЕТИ

Д.В. Звонов, Ф.Г. Нестерук

Научный руководитель – доктор технических наук, профессор Л.Г. Осовецкий

В работе предложено при оценке защищенности корпоративной сети учитывать динамику поля угроз, величину предотвращенного ущерба и значения экспертных оценок, отражающих взаимосвязь, с одной стороны, множества действующих на корпоративную сеть угроз с потенциальным ущербом от их реализации, с другой стороны, интенсивности использования механизмов защиты в структуре корпоративной сети для нейтрализации атак.

Введение

Показателям защищенности компьютерных систем удаляется достаточно большое внимание специалистов информационной безопасности (ИБ) [1–5], что обусловлено важностью корректного оценивания качества системы защиты информации корпоративных сетей.

В модели системы активного тестирования (САТ) оценка качества защиты обеспечивается расчетом рейтинга защищенности корпоративной сети. Рейтинговые показатели позволяют учитывать структурные характеристики системы защиты информации, изменение вероятности достижения злоумышленником объекта [3], частоту осуществления атак и ущерб от реализации угроз информационной безопасности [4], причем оценивается ущерб как в стоимостном исчислении, так и нематериальные потери, связанные с ущербом репутации и конкурентными возможностями хозяйствующего субъекта [6].

Комплексный характер рейтинговых показателей позволяет использовать их в качестве целевой функции при решении оптимизационных задач [7].

Оценка динамики защищенности корпоративной сети

Будем считать, что защищенность корпоративной сети оценивается количественно значением показателя (например, рейтинга), величина которого зависит от характеристик объекта, от поля угроз и от системы защиты.

$$Z = f(O, Y, S).$$

Характеристики объекта, поле угроз и свойства системы защиты

$$O = f_1(t),$$

$$Y = f_2(t),$$

$$S = f_3(t)$$

являются функциями времени.

Характеристики системы защиты информации зависят от характеристик системы активного тестирования:

$$S_{am} = f_4(t),$$

т.е. система защиты зависит от характеристик САТ:

$$S = f_5(S_{am}).$$

При таких исходных положениях цель моделирования сводится к решению оптимизационной задачи, когда защищенность объекта стремится к максимуму в соответствии с критерием возрастания рейтинга защищенности

$$S \to opt,$$

$$R \to max.$$

Качество системы активного тестирования можно оценивать частотой отражения известных угроз.

Пусть в начальный момент времени $t = t_0$ частота отражения известных угроз равна $S = \frac{n_{оу}}{n_{ду}} = 1$, так как $n_{оу} = n_{ду}$, где $n_{оу}$ – число отраженных угроз, $n_{ду}$ – суммарное количество известных (действующих) угроз. Тогда частота отражения известных угроз в момент времени $t = t_0 + \Delta t$ может быть описана выражением

$$S(t_0 + \Delta t) = \frac{n_{оу}(t_0)}{n_{ду}(t_0 + \Delta t)}.$$

Если ввести коэффициент k, характеризующий активность поля действующих угроз, а в качестве интервала дискретизации по времени выбрать величину, равную циклу активного тестирования $\Delta t = \frac{1}{f_{CAT}}$, то

$$n_{ду}(t_0 + \Delta t) = n_{ду}(t_0) + k \cdot \Delta t,$$

$$n_{ду}(t) = n_{ду}(t_0) + k \cdot t.$$

Здесь f_{CAT} – частота проведения циклов активного тестирования. Тогда через промежуток времени Δt частота отражения известных угроз равна:

$$S(t_0 + \Delta t) = \frac{n_{оу}(t_0)}{n_{ду}(t_0) + k \cdot \Delta t} = \frac{n_{оу}(t_0)}{n_{ду}(t_0) + k / f_{CAT}}.$$

Таким образом, частота отражения известных угроз есть функция от частоты проведения циклов активного тестирования, динамики изменения поля угроз и качества работы САТ.

Дальнейшая динамика показателя частоты известных угроз будет зависеть от качества работы адаптивных уровней системы активного тестирования, которые в течение цикла оптимизации структуры системы защиты информации идентифицируют наличие новых уязвимостей и угроз информационной безопасности корпоративной сети и ставят им в соответствие механизмы защиты, нейтрализующие их воздействие.

Другими словами, если число отраженных атак (числитель последнего выражения) будет соответствовать количественному показателю роста угроз (знаменатель последнего выражения), то показатель частоты отражения известных угроз будет стремиться к единице.

Комплексные показатели защищенности корпоративной сети

В соответствии с моделью системы активного тестирования решение о расширении классификации уязвимостей и классификации известных угроз информационной безопасности корпоративной сети производится в соответствии с системой экспертных оценок достоверности обнаружения угроз в разрезе отдельных механизмов защиты или отдельных уровней и аналогичных оценок ущерба, также соотносимых с отдельными механизмами защиты или отдельными уровнями системы защиты информации [7]. Экспертные оценки потенциального ущерба будем учитывать в относительных величинах, к примеру, по отношению к значению максимально допустимого ущерба в защищаемой корпоративной сети.

Результаты преобразований матриц экспертных оценок можно представить в виде матриц «угрозы – механизмы защиты»:

$$A_{m\times n} = \begin{pmatrix} a_{11} & a_{12} & a_{1\,n} \\ a_{21} & a_{22} & a_{2\,n} \\ a_{m1} & a_{m2} & a_{m\,n} \\ (\,x_1 & x_2 & x_n\,) \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_m \end{pmatrix},$$

где m – число механизмов защиты, n – число уровней системы защиты информации.

Если матрицу «угрозы – механизмы защиты» анализировать в разрезе уровней системы защиты информации, то каждый столбец матрицы – вектор активности уровня системы защиты информации, модуль которого рассматривается в качестве показателя значимости отдельного уровня системы защиты информации

$$x_j = \sqrt{\sum_{i=1}^{m} a_{ij}^2}\ ,\ \ j\ = 1,...,n\,,$$

Сопоставление показателей значимости в пределах строки показателей значимости позволяет выявить наиболее используемые уровни системы защиты информации по нейтрализации поля угроз.

Если же матрицу «угрозы – механизмы защиты» анализировать в разрезе механизмов защиты, то каждая строка матрицы – вектор активности отдельного механизма защиты, модуль которого рассматривается в качестве показателя значимости данного механизма защиты в системе защиты информации

$$x_i = \sqrt{\sum_{j=1}^{n} a_{ij}^2}\ ,\ \ i\ = 1,...,m\,.$$

Сопоставление показателей значимости в пределах столбца показателей значимости позволяет выявить наиболее задействованные механизма защиты в системе защиты информации.

Дальнейшие операции над матрицами достоверности обнаружения угроз и предотвращенного САТ ущерба дают возможность обобщить в элементах главной диагонали итоговой матрицы как показатели достоверности обнаружения угроз, так и показатели ущерба.

$$H_{m\times m} = \left(\begin{array}{ccc} h_{11} & h_{12} & h_{1\,m} \\ h_{21} & h_{22} & h_{2\,m} \\ \hline h_{m1} & h_{m2} & h_{m\,m} \end{array}\right),$$

где m – число механизмов защиты. В качестве обобщающего показателя защищенности корпоративной сети АКБ в разрезе механизмов защиты можно использовать рейтинговый показатель R_M – длину m-мерного вектора

$$R_M\ = \sqrt{\sum_{i=1}^{m} h_{ii}^{\,2}}\ ,\ \ i\ = 1,...,m\,.$$

Предложенные показатели защищенности могут учитывать не только динамику атак на корпоративную информационную сеть, но и распределение предотвращенного системой активного тестирования ущерба по механизмам защиты и уровням системы защиты информации, а рейтинговые показатели дают обобщенную оценку качества обеспечения информационной безопасности сети АКБ.

Заключение

Оценку защищенности корпоративной сети можно связать с предотвращением ущерба и, кроме того, использовать экспертные оценки для сопоставления, с одной стороны, множества действующих угроз с возможным ущербом от их реализации, с

другой стороны, размера ущерба с местом реализации атаки в структуре корпоративной сети. Приведенные показатели защищенности являются более информативными по сравнению с известными подходами, так как помимо динамики поля угроз учитывают достоверность использования механизмов защиты в структуре СЗИ, а также показатели ущерба, возникающего в результате реализации действующих угроз, который может быть предотвращен системой активного тестирования.

Литература

1. Девянин П.Н. и др. Теоретические основы компьютерной безопасности. М.: Радио и связь, 2000.
2. ГОСТ / ИСО МЭК 15408-2002 «Общие критерии оценки безопасности информационных технологий».
3. Осовецкий Л., Шевченко В. Оценка защищенности сетей и систем. // Экспресс-электроника. 2002, № 2–3. С. 20–24.
4. Жижелев А.В., Панфилов А.П., Язов Ю.К., Батищев Р.В. К оценке эффективности защиты информации в телекоммуникационных системах посредством нечетких множеств // Изв. вузов. Приборостроение. 2003. Т. 46. № 7. С. 22–29.
5. Звонов Д.В. Сетевой и системный уровень анализа защищенности в системах активного тестирования корпоративной сети. // Материалы 6 Международной научно-практической конференции «Безопасность и защита информации сетевых технологий. Common criteria». СПб, 2001. С. 19–21.
6. Карпычев В.Ю., Минаев В.А. Цена информационной безопасности. // Системы безопасности. 2003. № 5. С. 128–130.
7. Нестерук Ф.Г., Осовецкий Л.Г., Штрик А.А., Жигулин Г.П. К оценке информационных ресурсов и безопасности глобальных компьютерных систем. // Перспективные информационные технологии и интеллектуальные системы. 2004. № 4.
8. Корнеев В.В., Гареев А.Ф., Васютин С.В., Райх В.В. Базы данных. Интеллектуальная обработка информации. М.: Нолидж, 2001.

МОДЕЛЬ АКТИВНОГО ТЕСТИРОВАНИЯ КОРПОРАТИВНОЙ СЕТИ С ИСПОЛЬЗОВАНИЕМ АДАПТИВНЫХ СРЕДСТВ

Д.В. Звонов

Научный руководитель – кандидат технических наук, доцент Г.Ф. Нестерук

Представлена к обсуждению модель активного тестирования информационной сети акционерного коммерческого банка на основе интеллектуальных средств, согласно которой система активного тестирования (САТ) рассматривается как иерархия адаптивных уровней САТ, а именно: адаптивного уровня классификации уязвимостей и адаптивного уровня классификации угроз.

Введение

Разработку модели активного тестирования информационной сети акционерного коммерческого банка (АКБ) следует проводить в соответствии с системным подходом, определяющим методологию изучения сложных систем, к которым можно отнести современные человеко-машинные комплексы обеспечения информационной безопасности финансово-экономических структур, предприятий критических сфер производства, властных и управленческих структур [1].

Базовыми положениями системного подхода к исследованию сложных кибернетических комплексов являются такие принципы, как [2] целеобусловленность, относительность, управляемость, связность и моделируемость.

Помимо основных положений системного подхода, при построении адаптивной системы активного тестирования (САТ) информационной сети АКБ следует руководствоваться рядом принципов.

Прежде всего, для придания информационной сети АКБ адаптивных качеств следует воспользоваться арсеналом средств интеллектуального анализа данных, которые позволят, с одной стороны, изменять состав и функции САТ в соответствии с изменением поля угроз и перечня выявленных уязвимостей, а, с другой стороны, накапливать, например, в информационных полях нейронных сетей, и использовать опыт нейтрализации угроз, поиска и устранения уязвимостей.

Кроме того, необходимо учитывать, что модернизация системы обеспечения безопасности информационной сети должна проводиться одновременно с процессом нормального функционирования АКБ таким образом, чтобы в ходе модернизации уровень защищенности информационных ресурсов АКБ не ухудшился.

Следует объединить возможности имеющихся механизмов защиты СЗИ (рис. 1) с возможностями адаптивных средств активного тестирования в единой модели адаптивной САТ информационной сети АКБ [3] (рис. 2).

Модель активного тестирования информационной сети АКБ

Защищенность биосистем обеспечивается сочетанием эволюционных процессов, а также информационно-полевых, программных и структурных методов [4].

Информационно-полевые методы обеспечения защищенности биосистем связаны с представлением, хранением и передачей информации в пределах биологического вида в форме распределенных избыточных информационных полей [4, 5].

Обеспечение защищенности информационных процессов и ресурсов в биосистемах заключается не в задании алгоритмов поведения в виде определенной последовательности действий, а в описании структуры информационных полей.

На молекулярном уровне структура информационных полей биосистемы реализована в пространственной организации ДНК, а на уровне нервной системы – в распределенных межнейронных связях нейронных сетей [5, 6]. В обоих случаях для обеспечения защищенности используются механизмы избыточности, распределенный

характер представления и обработки информации, адаптивность, возможность передачи знаний в виде накопленного жизненного опыта в форме структурированных информационных полей, например, информационных полей ДНК и нейронных сетей. Структурные методы обеспечения защищенности биосистем связаны с формой представления информации (распределенных избыточных информационных полей) и иерархической организацией средств защиты.

Рис. 1. Система защиты информации информационной сети АКБ

Иерархия средств активного тестирования информационной сети АКБ

Для реализации адаптивных средств активного тестирования информационной сети АКБ иерархические уровни САТ, а именно адаптивный уровень классификации уязвимостей и адаптивный уровень классификации угроз (рис. 2), должны содержать [3, 7, 8]:

- систему правил нечеткого логического вывода, описывающих работу классификатора с учетом экспертных оценок;
- нейро-нечеткую сеть, в структуре которой отражена система правил нечеткого логического вывода;
- самообучаемую НС для решения задачи кластеризации входных векторов.

Основными механизмами реализации модели адаптивной системы активного тестирования являются:

- представление априорного опыта экспертов информационной безопасности в виде базы знаний, описанной системой правил IF-THEN;
- нечеткий логический вывод, который позволяет использовать опыт экспертов информационной безопасности в виде системы правил IF-THEN для начальной настройки информационного поля нечеткой НС;
- способность нейронных сетей к классификации и кластеризации;
- способность информационного поля нейронных сетей к накоплению опыта в процессе обучения.

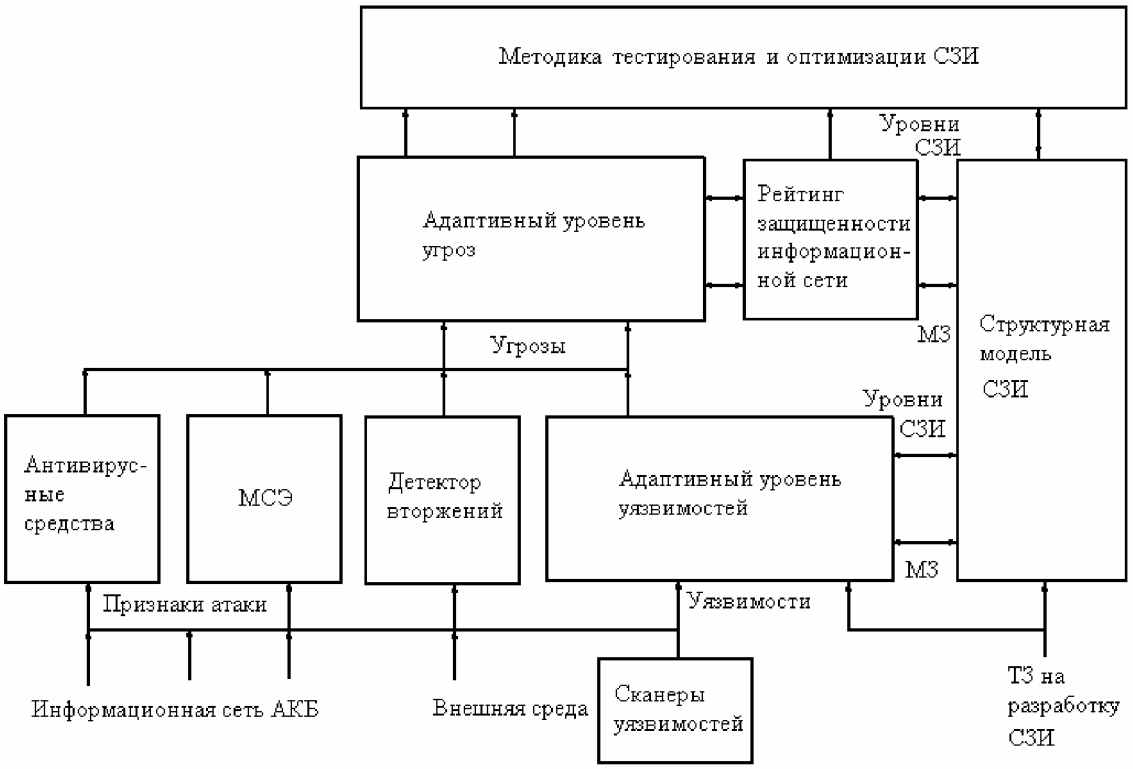


Рис. 2. Модель адаптивной CAT информационной сети АКБ

Нечеткая логика позволяет формализовать качественную информацию, полученную от экспертов информационной безопасности, и описать ее в виде системы правил IF-THEN, позволяющих анализировать результаты работы системы активного тестирования информационной сети АКБ. Нейронные сети дают возможность отобразить в структуре нейро-нечеткой сети априорную информацию.

Важной для модели адаптивной CAT особенностью нейро-нечетких сетей является способность автоматически генерировать систему правил нечеткого логического вывода в процессе обучения, извлекая скрытые закономерности из данных входной обучающей выборки. При отсутствии априорного опыта, но при достаточном объеме обучающей выборки нейро-нечеткая сеть преобразует скрытые во входных данных закономерности в систему правил нечеткого логического вывода [9].

С другой стороны, знания квалифицированных специалистов информационной безопасности, представленные в форме правил нечеткого логического вывода, прозрачным способом отражаются в структуре нейро-нечеткой сети (информационном поле НС). Предэксплуатационное обучение нейро-нечеткой сети позволяет настроить веса связей (откорректировать достоверность посылок и заключений нечетких правил) и устранить противоречивость системы правил нечеткого логического вывода.

Заключение

В соответствии с предложенной моделью CTA структуру СЗИ представляют в виде иерархии уровней, каждый из которых содержит аппаратно-программные механизмы защиты, а опыт специалистов информационной безопасности – системами правил нечеткого логического вывода для классификации уязвимостей и угроз безопасности корпоративной сети, а также значениями экспертных оценок.

Системы правил нечеткого логического вывода представляются в виде нейро-нечетких классификаторов, которые обучают на заданном подмножестве входных векторов известных уязвимостей корпоративной сети и векторах известных угроз.

Одновременно обучаются кластеризаторы в виде самообучающихся НС таким образом, чтобы число образуемых кластеров равнялось числу правил в системе правил нечеткого логического вывода.

В процессе обучения адаптивных уровней САТ изменяются информационные поля нейронных и нейро-нечетких сетей, адекватно видоизменяются системы правил нечеткого логического вывода и значения экспертных оценок.

Коррекция экспертных оценок вызывает изменение рейтинга защищенности корпоративной сети, который используется в качестве целевой функции методикой тестирования и оптимизации СЗИ. Методика определяет оптимальное расположение МЗ на иерархии уровней СЗИ по критерию максимизации рейтинга защищенности корпоративной сети.

Литература

1. Карпычев В.Ю., Минаев В.А. Цена информационной безопасности. // Системы безопасности. 2003. № 5. С. 128–130.
2. Красносельский Н.И. и др. Автоматизированные системы управления и связи: Учебник для вузов / Н.И. Красносельский, Ю.А. Воронцов, М.А. Аппак. М.: Радио и связь, 1988. 272 с.
3. Нестерук Г.Ф., Осовецкий Л.Г., Нестерук Ф.Г. О применении нейро-нечетких сетей в адаптивных системах информационной защиты. // Нейроинформатика-2005: Материалы VII всероссийской научно-технической конференции. М.: МИФИ (ТУ), 2005. Ч.1. С. 163–171.
4. Мелик-Гайназян И.В. Информационные процессы и реальность. // М.: Наука, 1998. 192 с.
5. Лобашев М.Е. Генетика. Л.: Изд-во ЛГУ, 1969.
6. Нестерук Г.Ф., Осовецкий Л.Г., Харченко А.Ф. Информационная безопасность и интеллектуальные средства защиты информационных ресурсов. (Иммунология систем информационных технологий). СПб: Изд-во СПбГУЭФ, 2003. 364 с.
7. Звонов Д.В. Системы активного тестирования при построении комплексных систем обеспечения безопасности корпоративных сетевых технологий. // Материалы 2-го международного научно-практического семинара «Безопасные информационные технологии – XXI». СПб, 2000.
8. Нестерук Ф.Г., Осовецкий Л.Г., Нестерук Г.Ф., Воскресенский С.И. К моделированию адаптивной системы информационной безопасности. // Перспективные информационные технологии и интеллектуальные системы. 2004. № 4. С. 25–31.
9. Negnevitsky M. Artificial intelligence: a guide to intelligent systems. Addison-Wesley. 2002. 394 p.
10. Нестерук Ф.Г., Осовецкий Л.Г., Штрик А.А., Жигулин Г.П. К оценке информационных ресурсов и безопасности глобальных компьютерных систем. // Перспективные информационные технологии и интеллектуальные системы. 2004. № 4.

МЕТОДИКА АНАЛИЗА ЗАЩИЩЕННОСТИ В ИНТЕРНЕТЕ

Т.А. Биячуев

Научный руководитель – доктор технических наук, профессор Л.Г. Осовецкий

В статье представлен подход к разработке методики анализа защищенности в Интернете, основанной на положениях корпоративной теории информации. Рассмотрены этапы методики, полученные результаты и перспективные направления дальнейших исследований.

Введение

Целью статьи является представление подхода к методике анализа защищенности в сети Интернет на основе корпоративной теории информации, предложенной в [1]. Глобальная сеть Интернет в ее нынешнем виде является сложной многосубъектовой системой. По некоторым оценкам, а дать точные цифры здесь практически невозможно, сеть Интернет составляют около 300 миллионов узлов. Конечно, даже такое большое количество подключенных компьютеров не делает Интернет сложной системой в понятиях общей теории систем. Однако три года назад ученые из японских университетов Tsukuba и Nihon обнаружили ряд общих черт между возникновением пиков трафика в Интернете и землетрясениями [2]. Ими было показано, что и землетрясения и «интернетотрясения» – так ученые назвали периодически возникающие заторы в Интернете – подчиняются теории сложных систем. Как и в любой многосубъектовой системе, здесь имеет место конкуренция за конечное количество ресурсов. В процессе взаимодействий происходит образование групп субъектов – в корпоративной теории информации они называются корпорациями, целью которых является перераспределение ресурсов для контроля над ними. Рассмотрим основные этапы анализа защищенности для данной модели, приложенные к сети Интернет. Отметим, что актуальность задачи обусловлена тем, что Интернет как часть киберпространства становится в настоящее время важнейшей сферой конфликтного информационного взаимодействия различных объединений субъектов [3].

Методика анализа защищенности в Интернете

Считается, что наиболее эффективной системой защиты является такая, которая обладает комплексом средств защиты информации и при этом способна не только отражать угрожающие воздействия, но и прогнозировать их появление для принятия упреждающих мер [4]. Это общий принцип стратегической защиты, в отличие от наиболее распространенного подхода применения разрозненных или комплексных средств защиты для противодействия текущим угрозам безопасности информации. Определим формализованное понятие системы защиты в виде тройки (S,P,F), где $S=(s_1,s_2..,s_n)$ – реализованные механизмы защиты, P – подсистема прогнозирования поля угроз для объекта защиты, F – некоторая функция от S и P, определяющая способность механизмов защиты S противостоять угрозам безопасности. Задача подсистемы прогнозирования угроз заключается в предупреждении возникновения возможных опасностей на заданном периоде упреждения (t_1,t_2), где t_1 – текущий момент времени, t_2 – предельный момент времени, на который распространяется прогноз. Тогда, если функция $F(S,P)>0$ для $\forall t \in [t_1,t_2]$, то система защиты (S,P,F) является «стратегической» в смысле способности выполнения заданных функций для актуального поля угроз в течение заданного времени. Таким образом, для решения задачи построения стратегической защиты в Интернете требуется провести комплексный анализ необходимых мер и средств. Декомпозиция исходной задачи приводит нас к методике анализа защищенности в сети Интернет, представленной на рис. 1.

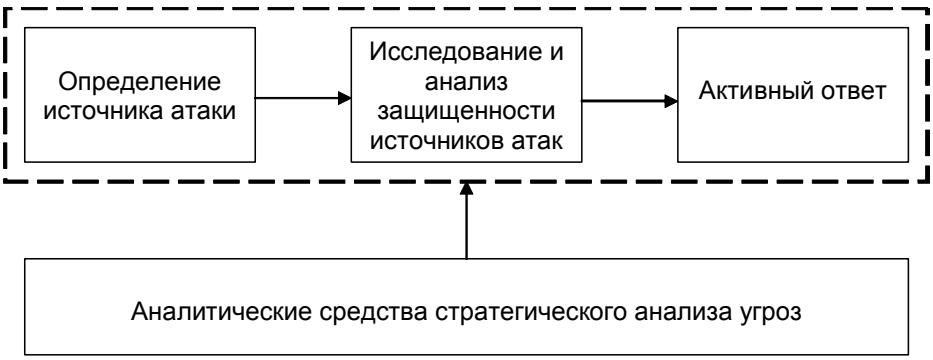


Рис. 1. Методика анализа защищенности в Интернете

Первым этапом анализа является определение источника угрозы. Современные информационно-телекоммуникационные сети предоставляют широкие возможности для эффективных, удаленных и анонимных атак. Поэтому важно разработать или улучшить существующие подходы к наиболее достоверному определению источника атаки. Следующей фазой, предваряющей ответные меры, согласно логике должны быть разведывательные действия относительно субъекта – источника атаки. Эти действия включают изучение противника, анализ его защищенности и уязвимых мест в защите. Здесь важна, с одной стороны, скрытность проведения подобных действий, а с другой – максимальная достоверность и полнота добываемой информации. Собрав доступную информацию, можно перейти к завершающей стадии – активному ответу, целью которого является прекращение вредоносной деятельности субъекта атаки.

Применение данной методики целесообразно не только после произошедшего инцидента, но также и при моделировании угроз со стороны потенциальных противников. Как констатируется в китайских стратагемах войны, манипуляции и обмана – друзей и союзников нет, есть только враги – явные, скрытные и потенциальные [5]. И здесь неизбежно поднимается вопрос стратегической защиты, которая предусматривает прогнозирование угроз и анализ выигрышной стратегии. Эти задачи выполняются по данной методике с помощью комплекса инструментальных средств стратегического анализа угроз. Раскроем последовательно этапы методики.

Исследование и анализ источников атак

В настоящее время актуальной задачей является не только отражение сетевой атаки (вирусной, хакерской), но и идентификация ее источника. Современные системы защиты умеют обнаруживать атаки, блокировать или обезвреживать их, а самые передовые, так называемые проактивные (proactive defense) или превентивные (intrusion prevention systems) системы защиты умеют также предотвращать некоторые типы атак, например, путем избирательной блокировки уязвимых портов. В отдельных применяемых на практике системах защиты предусмотрено обнаружение источника атаки и блокирование сетевых взаимодействий с ним [6]. В рамках конфликтного взаимодействия требуется рассмотреть проблему определение источника атаки и причин, вызвавших данные воздействия с его стороны. Задача достоверного определения источника атаки важна также потому, что в случае ошибочного определения ответные действия могут быть направлены против невиновных субъектов. При этом следует принять во внимание противодействие атакующего субъекта, затрудняющего свое обнаружение.

Атакованный субъект, достоверно определив источник атаки, проводит рекогносцировку, которая может осуществляться скрытно (пассивными методами) или частично скрытно (активными методами). Источником атаки в общем случае может быть любой сетевой узел, и следует рассматривать все варианты. Однако в связи с широким распространением службы web, число атак, обусловленных уязвимостями в программном обеспечении (ПО) веб-браузеров, имеет тенденцию к устойчивому росту в последние несколько лет [7]. Веб-сайты являются главным источником вредоносного ПО (malware), такого как троянские программы и программы-шпионы SpyWare (AdWare). В этой связи следует отметить проект Strider Honey Monkeys Exploit Detection Systems, запущенный в 2003 году исследовательским отделением компании Microsoft – лабораторией Microsoft Research в Редмонде, США. В исследовании используется система активных приманок (honeypots), путешествующих по вебу с целью обнаружения сайтов, которые являются источниками заражения вредоносным ПО. За год функционирования системы были обнаружены сотни сайтов – источников атак, а также, что особенно важно, «0-day» эксплойты, т.е. вредоносное ПО, использующее неизвестные пока широкой общественности уязвимости [8].

На кафедре безопасных информационных технологий (БИТ) СПбГУ ИТМО проводится ряд исследования в этой области, в том числе по механизмам пассивной разведки, т.е. максимально скрытного сбора информации на основе открытых источников без привлечения внимания. Также с 2003 г. нами проводится исследование веб-сайтов российского и внегосударственного (международного) сегментов Интернета в области анализа отдельных аспектов защиты, применяемых на веб-сайтах. Результаты исследования показали, что в целом уровень безопасности в российском сегменте в три раза меньше среднего международного уровня [9].

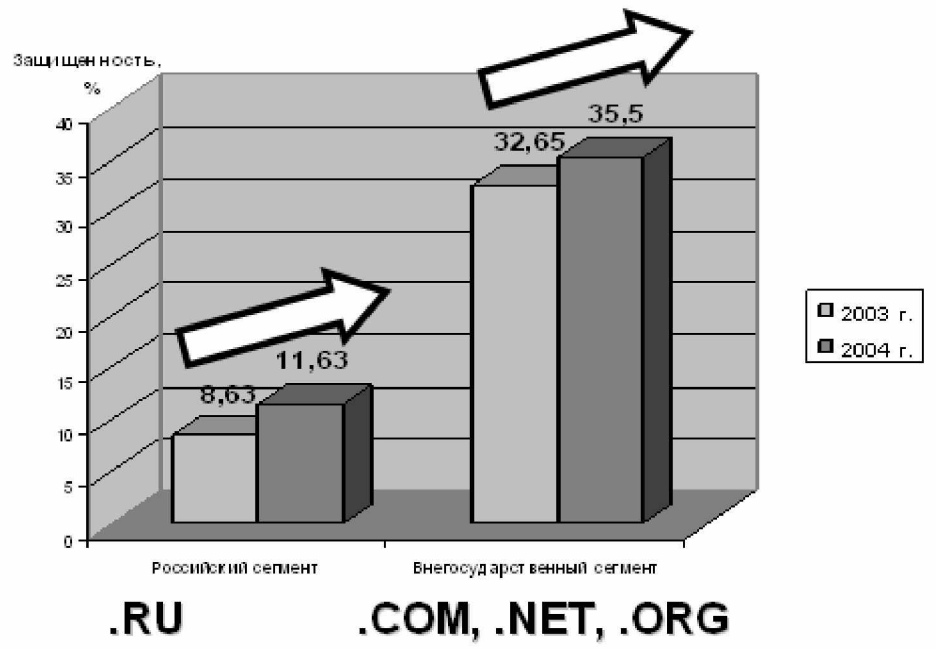


Рис. 2. Динамика средней защищенности веб-сайтов в российском и внегосударственном сегментах сети Интернет в 2003-2004 гг.

Методы активного ответа

Под активным ответом здесь понимаются целенаправленные действия по отношению к источнику атаки для прекращения вредоносного воздействия с его стороны. В зависимости от ситуации возможные адекватные меры могут включать как индивиду-

альное, так и распределенное ответное воздействие, в том числе и с участием других субъектов. Здесь, в отличие от предыдущего этапа, важна уже не скрытность, а скорость устранения угрозы.

Аналитические средства стратегического анализа угроз

Применение современных средств защиты информации носит индивидуальный, а зачастую и фрагментарный характер. Типовая схема защиты включает установку антивирусного средства для защиты от вирусных угроз, межсетевого экрана для защиты от сетевых атак и т.д. Сложившаяся ситуация представляет собой тактическую схему защиты от отдельных текущих угроз, в то время как комплексная защита должна выражать стратегическую направленность, а стратегия защиты включает прогнозирование поля угроз и адекватное реагирование. На кафедре БИТ исследования в данном направлении ведутся уже несколько лет, и на настоящий момент разработаны аналитические инструментальные комплексы «Морской бой» и «36 китайских стратагем». В многосубъектовом комплексе «Морской бой» реализованы модели рефлексивного взаимодействия субъектов, позволяющие в конкретных ситуациях выбрать оптимальную стратегию, а в комплексе «36 китайских стратагем» стратегии, применявшиеся в Древнем Китае в военном деле и политике и получившие название стратагем, формализованы в алгебре Лефевра и адаптированы для субъектов сетевых взаимодействий. Дальнейшие исследования в области стратегического анализа угроз будут расширены, в том числе и в направлении прогнозирования вирусных угроз совместно и на основе обширного опыта лаборатории Касперского, сотрудничество которой с СПбГУ ИТМО нашло отражение в организации тестовой антивирусной лаборатории на базе кафедры безопасных информационных технологий.

Заключение

С точки зрения информационной безопасности рассмотренный в настоящей статье подход к анализу защищенности в сети Интернет в соответствии с идеологией корпоративной теории информации обеспечивает стратегическую защищенность субъектов и открывает большие перспективы в создании принципиально новых средств защиты.

Литература

1. Осовецкий Л.Г., Немолочнов О.Ф., Твердый Л.В., Беляков Д.А. Основы корпоративной теории информации. СПб: СПбГУ ИТМО, 2004. 85 с.
2. Sumiyoshi Abe, Norikazu Suzuki. Omori’s law in the Internet traffic. // Europhysics Letters. Bologna. 2003, №61 (6). P. 852–855.
3. Гриняев С.Н. Информационная война: история, день сегодняшний и перспектива. СПб: Арлит, 2001. 240 с.
4. Расторгуев С.П. Инфицирование как способ защиты жизни. Вирусы: биологические, социальные, психические, компьютерные. М.: Изд-во агентства "Яхтсмен", 1996. 336 с.
5. Воеводин А.И. Стратагемы – стратегии войны, манипуляции, обмана, Красноярск: Кларетианум, 1999. 362 с.
6. Никишин А. Проактивная защита как она есть. // Internet URL http://www.viruslist.com/ru/analysis?pubid=170273483, 2005. 11 с.
7. Symantec Internet Security Threat Report. Trends for July 04 – December 04 // Internet URL http://www.securityfocus.com/corporate/products/tc.shtml, 2005. 96 p.

8. Yi-Min Wang, Doug Beck, Xuxian Jiang, Roussi Roussev. Automated Web Patrol with Strider HoneyMonkeys: Finding Web Sites That Exploit Browser Vulnerabilities, Microsoft Research, Redmond, 2004. 12 p.
9. Биячуев Т.А., Осовецкий Л.Г. Защищенность Web-сайтов. // Системы безопасности, 2003. №5. С. 124–126.

ОБЗОР И АНАЛИЗ ХАКЕРСКИХ КОНФЕРЕНЦИЙ

А.А. Воробьева

Научный руководитель – доктор технических наук, профессор Л.Г. Осовецкий

Статья представляет собой обзор существующих хакерских конференций, в ней сделана попытка показать важность данных мероприятий для улучшения обстановки в области защиты информации и доказать, что хакеры – специалисты в области информационной безопасности, заинтересованные в решении проблем этой области и в повышении уровня защиты информации.

Введение

В начале 1980-х гг. началось переосмысление роли компьютеров в жизни общества и государства. Массовое освоение информационного пространства должно было происходить с учетом факторов, формирующих современное состояние информационной безопасности, именно поэтому и стали проводиться различные конференции по компьютерной безопасности, в том числе и хакерские конференции.

К сожалению, в последние годы слова «хакер» и «преступник» стали практически синонимами. Но на этих конференциях хакеры – это специалисты в области информационной безопасности, заинтересованные в решении проблем этой области и в повышении уровня защиты информации. В наши дни крайне сложно найти человека, который воспринимал бы хакеров именно так, большинство из нас привыкли считать их правонарушителями, мешающими нашим интересам. Но эта позиция не совсем верна, так как именно из хакеров и получаются лучшие защитники информации.

Именно с этой точки зрения хакерские конференции – одни из самых важных мероприятий, способствующих повышению уровня защиты информации и квалификации специалистов.

Основная часть

История хакерских конференций начинается в 1980-х годах.

Chaos Communication Club. В 1984 году Берлин стал первым в мире городом, рискнувшим провести столь новое, ответственное и знаковое мероприятие. Конференция Chaos Communication Congress привлекла внимание всей хакерской общественности и объединила два враждующих лагеря – хакеров и специалистов по безопасности – для достижения одной общей цели – повышении уровня защищенности информации. Организаторы конференции (хакерское сообщество Chaos Communication Club) попытались донести до всего мира главную идею, что хакерство – не преступление. И сегодня конференция проходит под этим девизом. Стоит обратиться к ее тематике, чтобы понять истинность этого высказывания: Bluetooth, системы обнаружения вторжений, анонимность в Интернете, патенты, открытое программное обеспечение (ПО), криптография, надежность ПО для авиации, data-mining, нанотехнологии, политика и право в электронной сфере [1].

What the Hack? Страной, последовавшей за Германией в проведении хакерских конференций, стали Нидерланды. Здесь в 1989 г. под открытым небом прошел первый неформальный слет хакеров. Изначально это мероприятие было организовано группой людей, сосредоточенных вокруг небольшого андеграундного журнала Hack-tic. Этот журнал перестал издаваться в 1994 г., но, несмотря на это, конференция продолжает существовать и с каждым разом привлекает все больше участников. Так называемый фестиваль хакеров проходит раз в четыре года и каждый раз носит новое оригинальное название. Так, в 1989 г. прошел Galactic Hacker Party, за ним в 1993 г. – Hacking at the

End of the Universe, в 1997 г. – Hacking In Progress, в 2001 г. – Hackers at Large, в 2005 г. за ними последовал What the Hack?

Одновременно с ростом популярности расширяется и круг рассматриваемых вопросов, давно вышедших за рамки исключительно компьютерной тематики. В программе последней конференции, помимо исключительно компьютерных тем (ненадежность компьютеров, открытое программное обеспечение, открытые стандарты и программные патенты, электронно-цифровая подпись, биометрия и криптография), присутствуют следующие темы: свобода слова, прозрачность деятельности государства, тайна личной жизни, сети для социальных групп [2].

DefCon. В 1990-х гг. США переняли опыт проведения хакерских конференций. В 1993 г. прошла первая хакерская конференция на Северо-американском континенте – DefCon. Она сразу же выделилась своей массовостью и разнообразностью тематик. Сегодня это крупнейший съезд хакеров, объединяющий не только тех, кто взламывает сети, но и тех, кто их защищает и поддерживает. Ежегодно на конференцию собирается более 5000 участников со всего мира, и с каждым разом их количество растет. DefCon привлекает огромное внимание мировой общественности и является одним из ключевых мероприятий в области защиты информации. Конференция призвана служить обмену идеями о совершенствовании безопасности Интернет. Кроме теоретической части, состоящей из различных докладов и презентаций, конференция имеет также и практическую, представляющую собой различные конкурсы, состязания и дающая реальную возможность протестировать компьютерные системы на уязвимости. На конференции звучат доклады на актуальнейшие темы современной информационной безопасности – квантовый взлом, безопасность биометрических систем, защищенность смарт-карт, взлом Интернет-телефонии, реальные и потенциальные уязвимости систем электронного голосования, DNS, уязвимости Bluetooth и WiFi [3].

HOPE. Вторая, менее масштабная, но не менее интересная американская конференция – это HOPE (Hackers On Planet Earth). Впервые конференция прошла в 1994 г. в Нью-Йорке, после этого хакеры встречались еще трижды: в 1997 г. на конференции Beyond HOPE, в 2000 г. на HOPE2000 и в 2002 г., когда мероприятие назвали H2K2 [4].

На конференции были представлены доклады, посвященные актуальным темам в сфере компьютерной и информационной безопасности и права. Обсуждались проблемы защиты авторских прав на музыку и видео, а также торговых марок в Интернете. Большое внимание было уделено повышению активности спецслужб в сфере информационных технологий и телекоммуникаций. Хакеры обсудили организационные изменения в ФБР, направленные на борьбу с киберпреступностью, новые законодательные инициативы в этой области, системы слежения и связанные с этим проблемы ограничения гражданских свобод. Не остались без внимания и традиционные проблемы взлома информационных и телекоммуникационных систем, криптография и защита информации. На конференции также проводилась демонстрации приемов социотехники (social engineering) – обмана сотрудников какой-либо организации с целью получения паролей или другой конфиденциальной информации.

Toorcon. Конференция Toorcon, проходящая каждый сентябрь с 1998 г. в солнечном Сан-Диего, предоставляет возможность многим хакерам и специалистам по безопасности собраться для обсуждения важнейших тем и событий из области безопасности. В ней участвует около 500 человек. Цель Toorcon-а – организация недорогой и высококлассной конференции. На последней конференции (2005 г.) все доклады были разбиты на два потока – «Smoke» and «Mirrors» («Дым» и «Зеркала»). Раздел «Smoke» – это раздел, посвященный электронной безопасности: секретность, анонимность, криптография и криптоанализ, аппаратные атаки. В раздел «Mirrors» попадают доклады по сетевой безопасности и доверию: IDS, IPS, HoneyPots, фишинг-атаки, анти-фишинг, техники сетевых атак [5].

LayerOne. Последняя хакерская конференция, проводимая в США, – это LayerOne – техническая конференция и социальное событие, впервые прошедшее в 2004 г. в Лос-Анджелесе. Основная цель конференции – образовательная. Она охватила самые различные темы: информационная безопасность, шифрование, безопасность Bluetooth, взлом SIM-карт, безопасность почтовых серверов, сенсорные сети, военный тренировочный комплекс, социальные аспекты хакерства и воздействие технологии на общество [6]. Состав участников был разнообразным – от традиционных хакеров до специалистов по безопасности и представителей спецслужб.

Y2Hack. В 2000 г. в Иерусалиме прошла первая хакерская конференция в странах Востока, она получила название Y2Hack. Около 350 человек приняли участие в этой международной конференции. На конференции молодые хакеры имели возможность поучиться у классиков основам хакерской этики. Одна из главных целей мероприятия – это попытка улучшить образ хакера и дать верное представление о том, что делают эти люди: «Хакер – это человек, который работает с программами, чтобы их усовершенствовать» [7].

На конференции обсуждались новые технологии, проводились хакерские конкурсы. Одним из результатов мероприятия стал список коммерческих сайтов израильского Интернета, безопасность которых, по мнению хакеров, оставляет желать лучшего. Израильские правоохранительные органы запретили публикацию этого списка, полученного после проведения конкурса «HackTheseSites» («Взломай эти сайты»). Израильские компании сами предложили проверить их сайты на уязвимости. Кроме того, впервые открыто проведен хакерский аудит Y2sKan: просканировав все IP-пространство израильской сети, конференция определила процентное соотношение уязвимых систем и опубликовала статистику (не называя конкретных адресов). Еще одним ярким событием Y2hacK стала телефонная конференция с американским «хакером номер 1» Кевином Митником.

Перспективы проведения конференции в России

В России только несколько лет назад произошло переосмысление роли Интернета в жизни общества и государства. Массовое освоение информационного пространства должно происходить с учетом факторов, формирующих современное состояние информационной безопасности. Россия занимает одно из первых мест по количеству специалистов в области информационных технологий, и важно понимать важность решения проблем информационной безопасности. Поэтому и следует обратиться к мировому опыту проведения хакерских конференций. Эти мероприятия крайне положительно сказываются на обстановке в области защиты информации и рейтинге этих стран на мировой арене. Конференция позволит найти решения актуальнейших проблем современной информационной безопасности, таких как киберпреступность, угрозы беспроводной связи, безопасности Интернет и сетевой безопасности.

Заключение

Хакерские конференции – одни из важнейших мероприятий в области информационной безопасности, значительно способствующие повышению уровня защиты информации и квалификации специалистов.

Проблемы информационной безопасности уже давно находятся в центре внимания международного сообщества, и неудивительно, что актуальнейшие вопросы современной информационной безопасности поднимаются на хакерских конференциях.

Все доклады посвящены предотвращению компьютерных преступлений и борьбе с кибер-терроризмом. На конференциях последних двух лет поднимались вопросы и решались проблемы защиты беспроводной связи (WI-FI и BLUETOOTH), очевидна важность данных тем как перспективнейших технологий передачи данных. Не стоит недооценивать угрозы и опасности данной сферы. К ним относятся несанкционированный доступ (НСД) к аппаратным объектам, мониторинг и контроль исходящей и входящей информации объекта. Также звучали доклады, посвященные электронному голосованию, правовым и техническим аспектам данной темы. Обсуждения велись исключительно в русле устранения уязвимостей, возникающих в процессе разработки, совершенствования существующих систем (технических и программных средств) и предотвращения подделки результатов голосования. На конференциях прозвучали доклады по информационным технологиям, безопасности информации, криптографии, вопросам безопасности Интернет, защищенности смарт-карт, протокола IP, возможности предотвращения взлома систем Интернет-телефонии.

Результаты проведения хакерских конференций:

(а) привлечение внимания общественности к проблемам защиты информации;

(б) повышение мирового статуса страны, проводящей конференцию, в области информационной безопасности;

(в) привлечение дополнительных государственных и зарубежных инвестиций в область защиты информации, что качественно повышает уровень информационной безопасности и дает возможность более интенсивно развивать данную область.

Литература

1. www.ccc.de
2. www.whatthehack.org
3. www.defcon.org
4. www.h2k2.net
5. www.toorcon.org
6. www.layerone.info
7. www.y2hack.com

ЗАЩИЩЕННЫЙ ФОРУМ

В.О. Доскач, И.М. Инюшин, Р.В. Леонтьев
Научный руководитель – доктор технических наук, профессор Л.Г. Осовецкий

В статье обсуждается проблема защиты информационных систем от несанкционированного воздействия злоумышленников. На примере форума рассматриваются основные угрозы информационной безопасности, а также методы противодействия им. Предлагаются новые системы защиты от несанкционированного воздействия, рассматривается вопрос создания защищенного форума.

Введение

Защита конфиденциальной информации в Интернете приобретает в последнее время большую актуальность. На примере ведения защищенного форума можно наглядно показать, какие угрозы информационной безопасности на сегодняшний день наиболее опасны.

Опасность угрозы можно оценить, исходя из следующих показателей:

- частота активации угрозы;
- ущерб, который может нанести активация угрозы.

Ущерб условно можно ранжировать так:

- большой – кража всей информации, в том числе имен пользователей и паролей, или же уничтожение всей информации;
- средний – уничтожение или кража части информации, но не паролей и имен пользователей;
- незначительный – так называемый «spam». В случае форума это реклама, которую размещают на открытой части форума компьютеры-роботы или пользователи, которым нечего делать. При этом не происходит кражи или уничтожения информации.

Распространенные форумы

Распространенным явлением на сегодняшний день является использование так называемых готовых «движков» для создания форума. Этот подход позволяет сэкономить огромное количество времени, так как установка такого «движка» занимает считанные минуты, да и дальнейшая настройка и подстройка под общий дизайн сайта не занимает много времени. Но данный подход имеет и ряд очень серьезных недостатков, особенно с точки зрения обеспечения защищенности информации.

Чтобы обеспечить достойную защиту информации на форуме, можно пойти двумя путями – либо создавать свой собственный форум, либо писать дополнительные модули защиты для готового «движка».

Ясно, что первый подход предпочтительнее, так как в этом случае вы будете точно знать, как функционирует форум, все его варианты использования и возможные состояния. Соответственно можно определить набор угроз и предпринять меры для их нейтрализации. Если же пойти по пути создания дополнительных модулей защиты, то всегда возникает вопрос – а все ли защитили, все ли ситуации предусмотрели, так как в этом случае невозможно знать все варианты использования и возможные состояния системы.

На данный момент существует множество «движков» для построения форума. Наиболее распространенными из них являются PHPbb (PHP), Invision Power Board (PHP), VBulletin и др.

Виды угроз

Если посмотреть на все существующие платные и бесплатные форумы, то можно увидеть, что каждый из них имеет хотя бы самый низкий уровень защиты. Но нельзя не упомянуть о том, что они совершенно не справляются с более серьезными нападениями. Существует несколько видов атак, перечисленных ниже.

- Несанкционированная реклама, оставляемая программами-роботами. Иначе эту проблему можно назвать спамом.
- Похищение идентификатора сессии при помощи JavaScript или любого другого скрипта с последующим несанкционированным входом от имени атакуемого. Данный вид атаки называется XSS.
- SQL-инъекции. Данный вид атаки подразумевает получение доступа к базе данных или получение/модификацию части данных посредством ошибок серверного скрипта.
- Флуд – очень частое оставление сообщений на форуме одним и тем же пользователем (чаще всего они не несут в себе смыслового содержания).
- Получение повышенных прав или понижение работоспособности посредством использования ошибок в программном коде.
- Внедрение произвольного кода.

Обзор методов защиты

Большинство существующих форумов не реализуют в полной мере защиту. Также большим недостатком многих форумов является их открытость. Открытый исходный код позволяет любому человеку изучать его и находить ошибки в коде раньше самих разработчиков. Через эти ошибки в коде несанкционированный пользователь может нарушить работоспособность форума. Даже если ошибки были устранены, то все равно остается вероятность того, что не все пользователи вовремя обновляют свой экземпляр форума, вследствие чего форум может быть подвержен атаке через ошибку, которая была разглашена и исправлена в новых версиях форума.

Все форумы имеют базовый уровень защиты как на этапе регистрации, так и на этапе использования форума. Каждый пользователь для получения определенных прав должен быть зарегистрирован. При регистрации, помимо его настроек и имени, указывается контактный адрес электронной почты. Пользователь не может указать подложный адрес, так как необходимо перейти по специальной ссылке, указанной в письме. Это обходится разными способами, но самым простым является открытие временного электронного адреса на бесплатном почтовом сервере. Хоть этот уровень защиты очень легко обходится, но все же это является первым рубежом. Также при регистрации ограничиваются все программы-роботы, так как требуется ввести код на изображении, который был сгенерирован программой, и его весьма сложно считать автоматически, поскольку на текст накладываются помехи.

В большинстве форумов присутствует защита от подмены сессии. Если пользователь вдруг поменял свой IP-адрес, что бывает при работе через прокси-сервер, а также при получении доступа в Интернет через модем IP-адрес назначается пользователю автоматически и может быть изменен во время использования Интернет (но, по правде, случается это не очень часто).

Также присутствует защита и от флуда. Существует интервал времени, чаще которого сообщения от пользователя не принимаются. Чаще всего период блокировки невелик, но этого времени хватает, чтобы прочесть сообщение, а потом на него ответить. Кстати говоря, на всех форумах есть такая вещь, как цензура: недопустимые слова заменяются другими, чтобы оградить пользователей от ненормативной лексики.

В некоторых форумах присутствует частичная защита от SQL-инъекций. Суть защиты состоит в том, чтобы в форму не позволять вводить недопустимых символов. Иногда это опрометчиво реализуется через JavaScript, а иногда в самих серверных скриптах.

На этом чаще всего защита заканчивается. Даже вышеперечисленные методы защиты реализованы не на всех форумах. На форум также может быть осуществлена атака посредством получения доступа к серверу через уязвимости в программном обеспечении (Web-сервере, сервере баз данных, а также некорректной настройке операционной системы). Но это выходит за рамки данной статьи.

Обзор платформ

Языки, на которых написаны форумы – самые разные, начиная от PHP и заканчивая Java. Но ни один из известных форумов не может похвастаться стопроцентной защищенностью. В чем же связь такой «незащищенности» форумов и языка их реализации?

Рассмотрим поэтапно некоторые платформы и их слабые места.

PERL. Один из самых ранних языков, применявшихся при написании форумов – это Perl. На данном языке написан один из известнейших форумов – Ikonboard. Он также чаще остальных применялся и при написании гостевых книг. Какие самые частые ошибки встречаются при использовании данного языка?

Во-первых, это недостаточность проверки входных данных при подключении какого-либо файла, так как злоумышленник может вместо предполагаемого вами файла встроить свой собственный – это достигается при помощи использования внешних URL'ов. К примеру, http://yoursite.com/cgi-bin/some.pl?file=http://attacker.com/file.txt . В данном случае необходимо фильтровать переменные на наличие http:// или ftp://.

Либо, если ваш скрипт читает какой-либо файл с сервера, злоумышленник может прочитать другой файл, содержащий конфиденциальную информацию. Достигается это с использованием ..\ и ../ – переход в папки высшего уровня (зависит от операционной системы). Например, http://yoursite.com/cgi-bin/some.pl?file=../../../../../etc/passwd. В данном случае необходимо фильтровать переменные на наличие ../ ..\.

Кроме того, в Perl высока опасность того, что злоумышленник сможет встроить в переменную системные команды, которые будут выполняться от имени вашего скрипта. Это достигается использованием «пайпа» – вертикальной черты. Например, http://yoursite.com/cgi-bin/some.pl?some=|id; ls –la|. В данном случае необходимо фильтровать пайпы. Это уникальная уязвимость, относящаяся только к данному языку.

Еще в Perl есть уязвимость под названием «нулевой байт» %00 (\n в Си). Если встроить этот байт в переменную, то она оборвется там, где этот байт встроен, что может привести, опять таки, к чтению конфиденциальных файлов. Например, http://yoursite.com/cgi-bin/some.pl?file=../../../../../etc/passwd%00.

Встречаются и SQL инъекции, однако реже, чем в PHP, потому что многие форумы на Perl используют только текстовые базы данных.

PHP. Самый популярный на данный момент язык написания форумов, PHP, подвержен более всего двум видам уязвимостей.

Первая – SQL-инъекции, когда злоумышленник может встроить в ваш SQL-запрос свою команду. Это одна из самых частых и опасных уязвимостей, которая позволяет злоумышленнику читать и изменять вашу БД. Во многом это происходит из-за автоприведения типов в PHP (удобство в ущерб безопасности).

Вторая – это подключение «внешних файлов» функцией include или require аналогично Perl. Кроме того, возможна запись в PHP-файл вредоносного кода (редко встречающаяся в профессиональных разработках ошибка).

Можно отметить, что ни в PHP, ни в PERL нет встроенных функций защиты.

СИ. Возможно написание форумов на Си, но так как этот язык не приспособлен для работы в Интернете, практически нет форумов на этом языке. Поэтому рассматривать защищенность данной платформы нет особого смысла.

JAVA. Уязвимость форумов на JAVA исследована не слишком хорошо, но, как показывает практика, такие форумы гораздо лучше защищены за счет имеющихся встроенных средств защиты (например, от SQL-инъекции), а также благодаря более четкой структурированности языка и его строгости (отсутствию автоприведения переменных, что не позволяет злоумышленнику встроить, например, строковую переменную в числовую ячейку).

Все форумы также подвержены уязвимости под названием XSS (Cross-Site Scripting), основанной на использовании языков «скриптования» (в основном Java Script), которые встраиваются в сообщения форума и позволяют злоумышленнику красть конфиденциальные данные (Cookie или сессии).

Созданная нами защита

В качестве примера можно привести то, что было реализовано для защиты форума, созданного на кафедре безопасных информационных технологий СПбГУ ИТМО, от несанкционированного доступа. Мы столкнулись с проблемой использования открытого форума – большое количество общеизвестных уязвимостей. Самая насущная из них – межсайтовый скриптинг (иначе XSS). Она реализуется через ошибку в реализации обработки BB-тегов (это специальные теги для форматирования введенного текста как альтернатива HTML). Они были отключены вплоть до исправления ошибок в коде.

Также присутствовала проблема SQL-инъекций. В скрипты PHP очень легко встроить свой код, и при неправильной настройке интерпретатора он готов заменить любую переменную в скрипте на переменную из POST/GET данных. На этом форуме, как и на многих других, на протяжении всех версий появлялась информация о новых найденных ошибках. Казалось бы, им нет конца. Но нашелся один простой способ решения. К ядру форума был встроен скрипт, который проверяет все полученные от пользователя данные и ищет в нем ключевые слова языка SQL. Это не позволяет пользователям использовать эти слова в сообщениях, но зато делает форум неуязвимым от такого рода атак. Также в категорию запрещенных слов были введены HTML-символы и ключевые слова JavaScript, что позволяет отгородиться также и от XSS.

Хотя подобная защита не делает форум полностью защищенным, но это дает небольшие гарантии того, что он не будет «повержен» стандартными способами.

Также следует отметить, что на уровне программного обеспечения кафедральный WEB-сервер защищен от внешних атак, так как большинство портов закрыты, поэтому злоумышленник не может войти в прямой контакт с сервером баз данных.

Новые рубежи защиты

Но мы не останавливаемся на достигнутом уровне и придумываем новые рубежи защиты. С некоторыми из них мы вас сейчас познакомим.

Помимо методов пассивной защиты, необходимо упомянуть активную защиту. Будем отталкиваться от того, что форум может быть взломан в течение получаса посредством анализа и реализации уязвимости в коде сайта. Пусть его взломали и стерли всю информацию. Но разве это является проблемой, если у нас будут создаваться резервные копии базы данных параллельно работе форума каждые 5 минут. Естественно, для таких операций нужен мощный сервер. Поскольку сервер с PHP-сайтом настроен

обычно плохо и скрипт может перезаписывать себя другим, т.е. смысл – проверять время модификации файла или контрольную сумму. В случае малейшего изменения все возвращается назад. Допустим, что пользователь смог аварийно завершить работу программы (например, HTTP-сервера). Мы можем также создать процесс, который будет запускать программу заново. Естественно, пользователь, который совершил атаку, должен быть заблокирован файерволом или форумом для исключения повтора реализации уязвимости. После того, как система смогла восстановить рабочее состояние, администратор должен быть уведомлен по электронной почте о проблемах с сервером. При таком раскладе вероятность вывода из строя форума на длительное время уменьшается.

Но в этой системе не хватает еще одного элемента (механизма вычисления нарушителя). Естественно, нужно получить максимальную информацию о нарушителе. Во-первых, необходимо узнать все об IP-адресе: по какому маршруту он проходит, где находятся сервера из цепочки, используются ли прокси-сервер и так далее. Если нам повезет, то мы выйдем на провайдера в нашем государстве. В этом случае можно обратиться в компетентные органы. Но, чаще всего, этого не случается. Скорее всего, пользователь сидит через прокси-сервер, а то и через несколько, да еще и в разных частях планеты. А потом может оказаться, что он сидел за соседним компьютером. В случае анонимных прокси-серверов, да еще и в другой стране, мы вряд ли сможем найти пользователя. Но почему бы не проверять всех пользователей и не допускать к форуму тех, кто заходит на сервер по подобной схеме? Процесс анализа цепочки IP-адреса – долгий, а потому он должен выполняться в фоновом режиме. Тогда у нарушителя, в лучшем случае, при первом заходе с такого IP будет всего несколько десятков секунд на попытку взлома. После этого он будет заблокирован полностью.

Таким образом, предлагается ввести схему анализа источника угрозы и последующего блокирования всех пакетов с данного адреса на некоторое время. Но данный механизм можно упростить, если воспользоваться ошибками атакующего. Во-первых, можно встроить в страницу Java-апплет и JavaScript. Оба они будут заниматься сбором дополнительной информации о пользователе. С помощью JavaScript нельзя много узнать (только версию браузера, название ОС, а также разрешение экрана), а с помощью апплета можно узнать имя компьютера, язык системы, регион, а также реальный IP-адрес. Это поможет вычислить нарушителя.

Распространенной ошибкой хакеров является недосмотр за COOKIES. Можно помечать пользователя, и, если он потом зайдет через прокси-сервер для реализации уязвимости, то будет моментально опознан и заблокирован. Но такое бывает крайне редко.

Второй решаемой проблемой является флуд – публикация пользователями бессмысленных сообщений. На форумах имеется базовый вид защиты от таковых при помощи введения задержки публикации.

Нами предлагается развитие метода борьбы с флудом. Это, прежде всего, автоматическое и ручное соотнесение с пользователям уровня доверия, чтобы исключить излишние проверки. Во-вторых, подобные сообщения зачастую содержат специфическую лексику, на основе которой можно блокировать сообщение. Это не решает всех проблем, однако упрощает жизнь модераторам. В принципе, в целях борьбы с флудом установлена вполне устойчивая система назначения на каждый раздел модератора, который следил бы за выполнением установленных правил.

Также следует отметить, что техника фильтрации по ключевым словам и шаблонам сообщений дала положительные результаты при защите от роботов, которые оставляют информацию рекламного характера на форуме. Блокируются все 100% роботов. Если говорить обо всем объеме спама (в том числе и опубликованного вручную),

то фильтруется 99% всех сообщений. Удалением одного – двух сообщений в месяц занимается модератор.

Важность аудита

Во многих информационных системах ведется тщательная слежка за событиями, которые исходят от пользователя. Так почему бы не применить подобную политику и к форуму?

На большинстве готовых форумов существуют базовые события, такие как вход в систему, выход из нее, добавление/удаление/модификация сообщений (как общедоступных, так и личных), смена IP-адреса, изменение личных параметров и так далее. В принципе можно хранить даже все переменные, передаваемые скрипту от пользователя.

При наличии журнала есть возможность анализа действий пользователя. Любое отклонение от стандартного протокола работы должно быть зафиксировано и обработано. Это должно производиться исключительно автоматически, лишь в редких случаях эта информация должна быть проанализирована пользователем.

На основе получе нной информации программа может управлять правами пользователей на допуск к определенному виду информации, а также на выполнение каких-либо операций.

К сожалению, не все форумы обладают средствами аудита, что значительно усложняет работу администратора.

Заключение

В статье были рассмотрены основные методы защиты форума от несанкционированных действий со стороны пользователя. Обобщив все вышесказанное, можно говорить об эффективности новых уровней защиты, таких как:

- фильтрация ключевых слов (для защиты от XSS и SQL-инъекций);
- использования средств контроля целостности и работоспособности;
- использование средств анализа IP-адреса и последующего блокирования несанкционированных пользователей;
- автоматической пометки подозрительных пользователей;
- фильтров рекламы и флуда;
- ведение аудита.

Задействовав данные уровни защиты, можно получить возможность в большей мере контролировать действия пользователей и защитить форум от попыток нарушения работоспособности и чистоты информационной среды.

В планах авторов – разработка адаптивной системы защиты с использованием интеллектуальных технологий, таких как нейро-нечеткие сети и система экспертных оценок. Эти технологии должны значительно увеличить функциональность и «выживаемость» информационной системы кафедры.

ПРОГНОЗИРОВАНИЕ ВИРУСНЫХ АТАК: АНАЛИЗ, РЕШЕНИЕ, ВЫВОДЫ

А.В. Захаров

Научный руководитель – доктор технических наук, профессор Л.Г. Осовецкий

Развитие и глубокое проникновение информационных технологий в современную жизнь делает нас зависимыми от угроз информационной безопасности. Для эффективной работы в области защиты информации необходим детальный анализ источников угроз, определение причин и взаимосвязей проведения атак в зависимости от целого ряда параметров.

Наиболее распространенным видом реализуемых угроз являются вирусные атаки.

Что можно сказать о прогнозе вирусной активности? Прогнозирование – это предвидение, которое базируется на специальном научном исследовании. В теории экономического и социального прогнозирования существуют два подхода к прогнозированию. Поисковый подход – это определение возможного положения явления в будущем; нормативный подход – определение путей и сроков достижения возможного положения явления, которое принято за цель. В случае исследования вирусной активности применим исключительно поисковый подход к прогнозированию, когда оцениваются варианты и степени вероятности состояния вирусной активности в будущем.

Для успешного противодействия необходим подробный анализ вирусной активности, исследование причин возникновения и определение зависимостей. Для этого применимы следующие методы.

- Фундаментальная аналитика включает исследование факторов из области экономики и бизнеса, политики, психологии, влияющих на вирусную активность.
- Техническая аналитика – это исследование динамики вирусной активности посредством математического аппарата, графиков и формул с целью прогнозирования следующих атак.
- Моделирование информационной среды подразумевает построение различных моделей, описывающих взаимодействие субъектов информационной среды. Исследование моделей позволяет выявить тенденции развития отношений субъектов информационного процесса, определить возможные угрозы и слабые места информационной системы, охарактеризовать процесс атаки, определить методы эффективного противодействия возникшим угрозам.

Вирусная активность является следствием определенных причин, и, исследуя эти причины методами фундаментального и технического анализа, можно выявить зависимости, которые позволят с определенной вероятностью прогнозировать возникновение таких событий, как вирусные эпидемии, массовые сбои автоматизированных систем, рост вирусной активности.

Прогнозирование возникновения угроз основывается на построении и анализе поведения моделей информационной среды. Имитируя информационный процесс и акцентируя внимание на отношениях субъектов информационного пространства, можно наблюдать за реализацией угроз. Исследование проведения атаки позволяет выявить причины, описать процесс и последствия, найти уязвимости и разработать методы эффективного противодействия. Таким образом, необходимо разработать модель информационного взаимодействия.

Имитационная модель позволяет связать фундаментальные причины информационных процессов с показателями технического анализа.

Что можно сказать о характере информационного взаимодействия? Все процессы зависимы от движения информационных потоков вокруг них. В связи с этим возникают следующие вопросы.

- Как описать информационное воздействие?
- Каким законам оно подчиняется?
- Какими методами можно спрогнозировать дальнейшее развитие событий?

Чтобы ответить на эти вопросы, необходимо проанализировать информационную среду, выявить основные закономерности и на их основе разработать модель информационной системы. Имитационная модель информационного взаимодействия призвана отразить основные процессы, происходящие в информационной среде, с ее помощью можно исследовать информационные процессы, прогнозировать развитие событий, анализировать различные сценарии, линии поведения, политики в процессе взаимодействия. Перечислим основные постулаты данной модели.

- Информация является ограниченным ресурсом. Отсюда вытекает ценность информации. В условиях дефицита ресурсов возникает борьба за право их использования.
- Информация – это средство управления. С ее помощью можно влиять на процессы, происходящие в информационной среде. Влиянию информационных потоков подвержены ценностные установки, цели, приоритеты ресурсов.

Информационное взаимодействие выражено стремлением объектов к достижению поставленных целей, борьбой за ограниченные ресурсы, воздействием на объекты приоритетов и ценностей. Наиважнейшую роль играет тщательный анализ внутренней и внешней среды объекта, когда учитывается всевозможные факторы, оказывающие влияние на его деятельность.

Исследование влияния фундаментальных причин на дальнейшее развитие взаимодействия внутри информационного пространства позволяет использовать модель в качестве инструмента для прогнозирования. Введя в эту динамическую модель вероятностные коэффициенты, мы получаем возможность анализировать различные сценарии информационного взаимодействия. На этом этапе берет начало исследование прогнозирования развития событий.

Прогнозирование оценивает варианты развития информационного взаимодействия на основе анализа модели информационной среды. С помощью математического аппарата просчитываются вероятности возможных сценариев. Похожий механизм используется в экономике, когда просчитывается целесообразность инвестиций или развития спроса. На практике применение такого подхода к прогнозированию можно увидеть в трендовых моделях и методах экспертных оценок. Однако новые достижения научной мысли на стыке математики, статистики, информатики и кибернетики продолжают оставаться недостаточно востребованными большинством компаний.

Возможность практической реализации данной модели видится в использовании программно-аппаратных средств на основе методов «мягких» вычислений, среди которых можно отметить нейронные сети, генетические алгоритмы и нечеткую логику. Эти методы существуют уже несколько десятилетий, но практически применяются очень редко. Исключением являются разве что компании, чья основная деятельность – активные операции на финансовых рынках, страховщики и некоторые банки.

В сфере информационных технологий, в частности в сфере информационной сетевой безопасности, в последнее время заметна тенденция целенаправленного осуществления угроз. Если раньше нарушители преследовали цели личной славы, проверяли свои способности в написании вредоносного кода и в использовании уязвимостей, то сейчас атаки производятся, целенаправленно преследуя определенные цели. Чаще всего это попытки получения материальной выгоды в связи со сбоями работы серверов фирм-конкурентов, шантаж угрозами нарушения работы серверов, удаления или опубликования важной и секретной информации и пр.

Наиболее часто сейчас применяемый многими антивирусными компаниями абстрактный анализ среды и прогнозирование вариантов развития событий, основанный на

количестве обнаруженных угроз, атак, вирусной активности, не является объективным. Целесообразно проводить исследования в области окружающей информационной среды конкретного объекта или предприятия с целью выявления возможных угроз, уязвимостей, прогноза возможных атак (например, вызванных желанием конкурентов найти информацию о новых разработках). Такие исследования будут включать анализ ресурсов предприятий, внешнюю и внутреннюю среду фирмы, направления деятельности конкурентов в занимаемых сегментах рынка, исследования в сфере разработки инновационных программ.

Ввиду целенаправленного воздействия на объект защиты итогом работы модели будет исследование сильных и слабых сторон объекта, фундаментальный анализ причин возникновения угроз, анализ вероятностных вариантов развития событий и как итог – вывод и разработка дальнейшей стратегии укрепления положения объекта в информационной среде.

Полученные результаты позволят своевременно реагировать на возникновение новых угроз, прогнозировать их появление, разрабатывать методы эффективного противодействия.

Литература

1. Осовецкий Л.Г., Немолочнов О.Ф., Твердый Л.В., Беляков Д.А. Основы корпоративной теории информации. СПб: СПбГУ ИТМО, 2004
2. Нестерук Г.Ф., Осовецкий Л.Г., Нестерук Ф.Г. Адаптивная модель нейросетевых систем информационной безопасности. // Перспективные информационные технологии и интеллектуальные системы. 2003 №3.
3. Романенко И.В. Социальное и экономическое прогнозирование: Конспект лекций. – СПб: Издательство Михайлова В.А., 2000. 64 с.
4. Статистическое моделирование и прогнозирование / Под ред. А.Г. Гранберга. М.: Финансы и статистика, 1990. 382 с.
5. Суворов А.В. Методы построения макроэкономических сценариев социально-экономического развития. // Проблемы прогнозирования. 1993. №4. С. 27–39.
6. Домарев В.В. Безопасность информационных технологий. Методология создания систем защиты. ТИД «ДС», 2002.
7. Шабанов П.И. Методы научного прогнозирования и их практическое применение. // TOP-Manager. 2006. URL: http://www.top-manager.ru/?a=1&id=983

ИССЛЕДОВАНИЕ ВРЕДОНОСНОГО КОДА

Д.В. Черемушкин
Научный руководитель – доктор технических наук, профессор Л.Г. Осовецкий

С развитием информационных технологий все большее значение и актуальность приобретает вопрос информационной безопасности. Говоря об информационной безопасности, нельзя не упомянуть о таком источнике угроз, как вредоносные программы. При защите от подобного вида угроз большую роль играет анализ механизмов работы вредоносных программ.

Среди проведенных исследований в данной области можно выделить работы [1–6]. Однако отмеченные исследования направлены в основном на изучение определенной вредоносной программы. Конечно, это способствует наилучшему пониманию особенностей работы отдельного кода и облегчает изучение его модификаций. Но, с другой стороны, недостаточно формализована общая методология, алгоритм проведения анализа вредоносного кода (ВК). Кроме того, информация по настоящей тематике представлена в основном в зарубежных источниках и, как правило, не отличается объемом. В настоящей работе преследуется цель – описать основные этапы исследования вредоносного кода, их взаимосвязь и некоторые особенности. Приведенная в работе информация относится к исследованию вредоносных программ, представляющих собой приложение Windows (PE-EXE файл).

Пожалуй, первоочередной задачей следует считать обнаружение вредоносного кода, поиск объекта анализа. Ниже перечислены способы получения вредоносного кода, в результате которых был достигнут положительный результат:

- Интернет-сайты, содержащие в информационных целях различную информацию о вредоносных программах, их исходные коды и готовые образцы;
- Интернет-сайты различных команд, занимающихся написанием вредоносных программ;
- отчеты специализированных программ, предназначенных для предупреждения пользователей о возможной опасности при посещении ресурсов сети Интернет (McAfee SiteAdvisor, StopBadware);
- электронная почта;
- файлообменные сети общего пользования;
- поисковые системы сети Интернет.

Другим этапом, который нельзя недооценивать, является систематизированное накопление вредоносного кода и полученных при его исследования результатов. Целесообразность такого подхода может быть обоснована следующим образом:

- часто имеет место ситуация, когда большая часть кода одной вредоносной программы повторяется в другой;
- нередко несколько вредоносных программ распространяются вместе, дополняя друг друга, а для выяснения связей необходимо иметь такой набор, а не каждую вредоносную программу в отдельности;
- могут существовать несколько различающихся экземпляров одного вредоносного кода;
- вредоносные программы классифицируются по семействам и классам, что может облегчить процесс исследования схожих вредоносных программ.

Одним из возможных вариантов присваивания имен файлам в вирусной коллекции является соответствие имени файла названию ВК согласно стандарту одной из антивирусных компаний. При наличии нескольких экземпляров одного ВК в имени при таком подходе используется индекс.

Основным этапом исследования является непосредственно анализ вредоносного кода. Для этого применяется reverse code engineering (RCE) — набор методик и инструментов для анализа разработанного ПО с целью выявления его архитектуры, методов и особенностей работы. На русский язык этот термин переводится как обратное проектирование, реверсивное программирование, реинжиниринг.

Процесс анализа вредоносного кода включает в себя следующие этапы (в скобках перечислены используемые программные средства):

1. проверку целостности исследуемого объекта (эту операцию могут осуществлять некоторые антивирусы, например, Антивирус Касперского);
2. проверка на наличие слоев паковки и их идентификация (PEiD);
3. распаковка, если требуется. Этот процесс зависит от использованного паковщика, наиболее популярным является UPX (upx.sourceforge.net);
4. получение информации о поведении исследуемого объекта средствами системного мониторинга;
5. контроль изменений среды, в которой выполнен исследуемый код.

   Контролируемыми элементами в таком случае могут являться:
   - файловая система (FileMon);
   - реестр (RegMon, RegShot);
   - запущенные процессы (Process Explorer);
   - открытые порты (PortMon, netstat);
   - сетевая активность (IRIS Network Traffic Analyzer, Ethereal);
   - вызов API-функций (Kerberos, ApiMonitor, ApiSpy32);
   - данные о пользователях, группах, сервисах.
6. Совместное использование дизассемблера и отладчика для более детального изучения программы (IDA; SoftIce, OllyDbg).

Последним этапом в исследовании является оформление результатов. Ниже в качестве примера приведены результаты, полученные в результате исследования вредоносной программы Backdoor.Win32.Sumatrix.

Объект исследования. Троянская программа, предоставляющая удаленный доступ к компьютеру. Является приложением Windows (PE EXE-файл). Упакована с помощью UPX. Имеет размер в упакованном виде 4 608 байт, в неупакованном – 7 168 байт.

Инсталляция. После запуска вредоносная программа помещает свою копию в корневой каталог Windows под именем "rundll32.exe". Затем в зависимости от собственных настроек троян способен добавлять следующие данные в реестр:

- [HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]
  a = "%WinDir%\rundll32.exe"
- [HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run]
  c = "%WinDir%\rundll32.exe"

В ОС Windows NT 4.0, 2000, XP, 2003 кроме того вносятся изменения в следующие ветви реестра:

- [HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon]
  Shell = "explorer.exe rundll32.exe"
- [HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\Windows]
  load = "%WinDir%\rundll32.exe"
  run = "%WinDir%\rundll32.exe"

В ОС Windows 9x, ME модифицируются:

- секция [boot] файла "%WinDir%\system.ini",

параметр shell = "explorer.exe rundll32.exe"

• секция [windows] файла "%WinDir%\win.ini",

параметр load = "%WinDir%\rundll32.exe"

параметр run = "%WinDir%\rundll32.exe"

Также троянская программа скрывает свое присутствие в списке задач в ОС Windows 95, 98, ME.

Деструктивная активность. Backdoor.Win32.Sumatrix сочетает в себе следующие функции:

- при установке связи с сетью Интернет – отправка «хозяину» ICQ-сообщения, содержащего сетевое имя компьютера, IP-адрес, версию ОС, информацию для установки удаленного доступа (открытый порт, пароль) – функция не работоспособна;
- завершение сеанса текущего пользователя (ОС Windows NT 4.0, 2000, XP, 2003) или выключение компьютера (ОС Windows 9x, ME);
- получение сведений о местонахождении системных директорий ОС;
- загрузка на зараженный компьютер произвольной информации. Имя файла и каталог, где сохраняются данные, определяются злоумышленником;
- запуск программ, указанных злоумышленником;
- удаление ссылок на программу из секций автозапуска реестра, конфигурационных файлов.

Кроме того, вредоносная программа имеет простой механизм аутентификации (по паролю).

Рекомендации по удалению:

1. завершить процесс «rundll32.exe»;
2. удалить файл «rundll32.exe» из каталога Windows;
3. удалить указанные выше параметры в реестре и конфигурационных файлах.

В заключение следует отметить, что изучение вредоносных программ, оказываемого ими влияния важно и необходимо. Полученные в результате таких исследований результаты могут использоваться для:

- создания вирусных энциклопедий в информационных целях;
- анализа существующих и развивающихся возможностей вредоносных программ и связанных с этим угроз;
- разработки сигнатур, позволяющих идентифицировать вредоносный код;
- защиты от вредоносного ПО.

Литература

1. Rozinov Konstantin. Reverse Code Engineering: An In-Depth Analysis of the Bagle Virus. 2004.
2. Hines Eric S. MyDoom.B Worm Analysis. 2004.
3. Friedl Stephen J. Analysis & reverse engineering of the "Iraq Oil" worm. 2005.
4. Friedl Stephen J. Analysis of the Troj/Winser-A Malware. 2005.
5. Roculan Jensenn, Hittel Sean, Hanson Daniel, Miller Jason V., Kostanecki Bartek, Gough Jesse, Mario van Velzen, Friedrichs Oliver. SQLExp SQL Server Worm Analysis. 2003.
6. Eduardo Labir. VX Reversing III – Yellow Fever, 2005.

3 ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В ОБРАЗОВАНИИ

ИСПОЛЬЗОВАНИЕ ДИНАМИЧЕСКИХ ИНТЕРНЕТ-ТЕХНОЛОГИЙ В ЕСТЕСТВЕННОНАУЧНОМ ОБРАЗОВАНИИ

С.М. Вергезова
Научный руководитель – кандидат технических наук, профессор М.И. Потеев

В статье рассматриваются аспекты использования динамических Интернет-технологий в естественнонаучном образовании на примере дисциплины «Концепции современного естествознания».

Одним из наиболее продуктивных вариантов исследовательской работы студентов является работа по методике совместных проектов [1]. Она состоит в том, что студенты при помощи преподавателей должны создать, например, лабораторную работу или демонстрацию, используя уже полученные знания и умения. По такой методике удобно создавать, в частности, лабораторные работы по физике, экологии, демонстрации по биологии, химии, где используется технология Flash. Использование данной технологии обусловлено тем, что многие сложные процессы доходчиво можно показать только в динамике.

Проиллюстрируем использование этой методики при организации научно-исследовательской работы студентов, изучающих дисциплину «Концепции современного естествознания» (КСЕ), с применением современных компьютерных технологий. В этом случае преподаватель КСЕ выступает постановщиком задачи, преподаватель информатики – консультантом по программированию в определенной среде, а студенты выступают в качестве исполнителей.

Известно, что применение современных методов компьютерного моделирования и разработанные на их основе виртуальные лабораторные работы, интерактивные мультимедийные демонстрации позволяют разнообразить способы обучения и повысить его наглядность. В связи с этим для повышения успешности преподавания дисциплины «Концепции современного естествознания» предлагается использовать не только готовые виртуальные работы и демонстрации, но и те из них, которые разработаны студентами.

При моделировании физических явлений значима визуализация поведения системы в реальном масштабе времени, что позволяет непосредственно оценить физическую адекватность модели в ходе компьютерного эксперимента. Подобную возможность представляет на сегодняшний день векторная анимация Macromedia Flash MX и скриптовый язык ActionScript [www.macromedia.com, www.actionscript.org].

Главным достоинством использования среды Macromedia Flash является относительная простота создания в ее среде графических образов и их мультипликации. Благодаря векторной технологии создания графики, Macromedia Flash позволяет радикально сократить объем данных, описывающих движущееся изображение. Создав в одном из кадров векторный рисунок, можно задать (в том числе на языке ActionScript, встроенном во Flash) траекторию его движения. В файле сохраняются только первоначальное изображение и уравнение траектории его движения, а все промежуточные кадры рассчитываются в момент воспроизведения. Этим Macromedia Flash существенно отличается от обычных компьютерных видеороликов, в которых кадры последовательно сменяют друг друга. Для хранения и передачи видеозаписей требуются огромные объемы памяти и сверхскоростные каналы связи, в то время как файлы Flash загружаются с использованием эффективной потоковой модели. При этом первые кадры становятся доступными практически сразу же.

Одним из достоинств Flash является наличие языка сценариев ActionScript. Часто достаточно стандартного набора действий, например, перехода на определенный кадр фильма, открытие html-страницы, загрузки и передачи данных, реакции на нажатие кнопок. Диапазон возможностей ActionScript очень широк. Помимо основных действий, можно управлять объектами, изменяя любые их параметры, загружать дополнительные модули, обмениваться данными со скриптами на html-странице, написанными на языке JavaScript. Подборка специальных функций служит для математических вычислений, необходимость в которых нередко возникает при работе с графикой. Встроенный контроль синтаксиса и удобные средства отладки обеспечивают возможность оперативного нахождения ошибок в сценарии. В целом ActionScript является полнофункциональным языком, позволяющим писать программы, организованные по модульному принципу, в то же время он очень прост в изучении [2].

Создание динамических демонстраций включает выполнение следующих этапов:

1) выбор темы,
2) обзор теоретического материала по теме,
3) составление сценария анимации для каждого ролика,
4) создание анимированных роликов с использованием Flash-технологий,
5) составление текстового сценария речевого сопровождения роликов,
6) запись и обработка звука,
7) создание интерфейса и html-страниц.

На первом, втором и пятом этапах студент взаимодействует с преподавателем КСЕ, на остальных – с преподавателем информатики. В случае необходимости студент должен иметь возможность обратиться за консультациями к профессиональному художнику-дизайнеру.

Созданные студентами виртуальные лабораторные работы и демонстрации могут быть объединены посредством языка гипертекстовой разметки (HTML) в единый учебный блок в виде web-страниц.

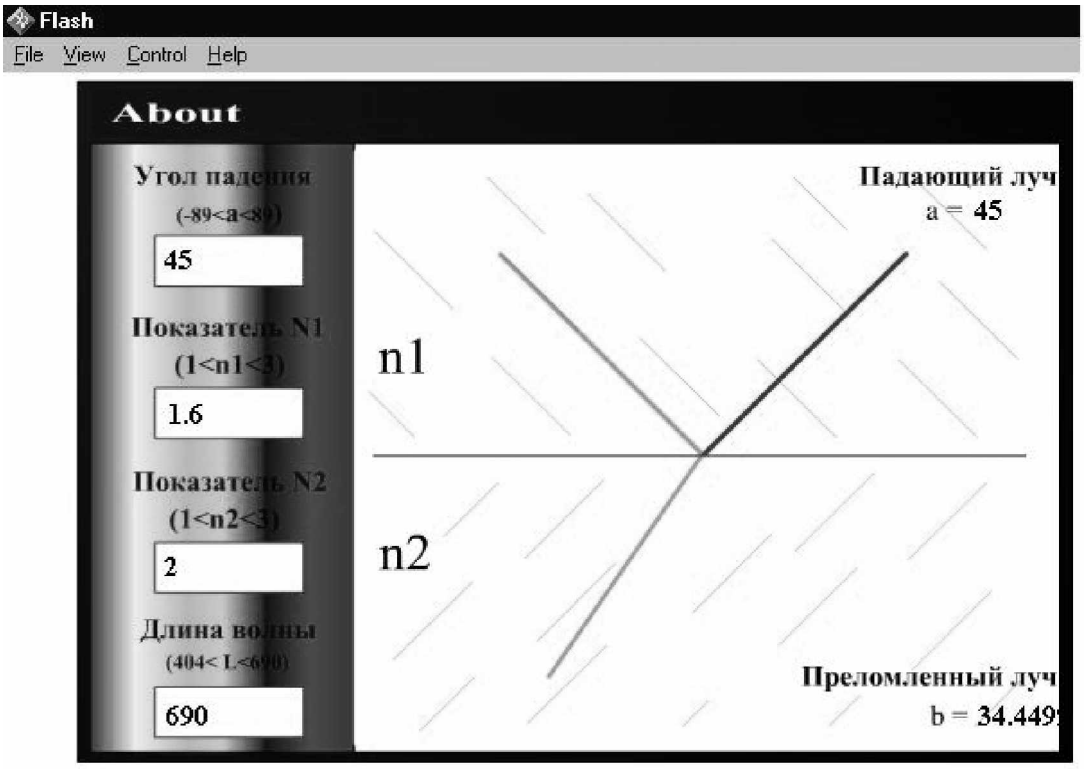


Рис. 1. Исследование хода лучей на границе двух сред

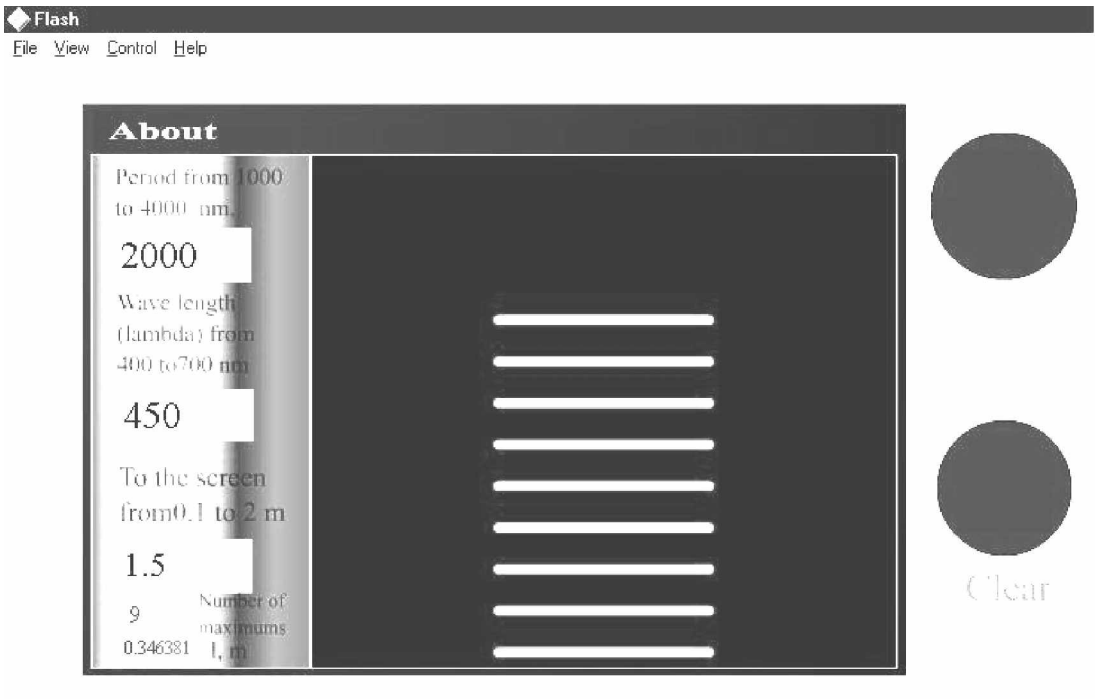


Рис. 2. Исследование дифракции на решетке

Приведем примеры некоторых разработок студентов гуманитарного факультета СПбГУ ИТМО, выполненных ими при изучении КСЕ. В процессе этих разработок использовались учебники [3, 4].

На рис. 1 и 2 показаны фрагменты интерактивных лабораторных работ по исследованию физических законов (преломление лучей и дифракция).

На рис. 3 и 4 показаны фрагменты интерактивных демонстраций процессов по биологии и генетики.

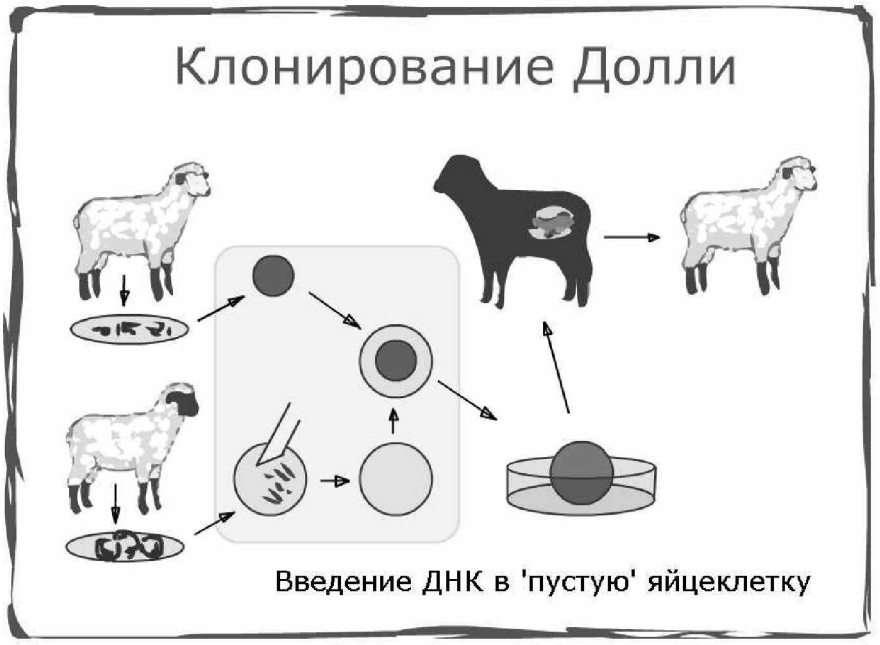


Рис. 3. Клонирование

Рис. 4. Строение клетки растения

Как показывает опыт, эффективность описанного подхода объясняется следующими обстоятельствами:

1) студенту приятно, что результаты его работы нужны кафедре и будут реально использоваться в учебном процессе, а убежденность в практической полезности своего труда – залог успеха;
2) проектирование динамических демонстраций и лабораторных работ является комплексной задачей, требующей компетенций автора учебного материала, компьютерного методиста, системотехника компьютерных средств обучения и специалиста по их реализации, и, следовательно, при описанном подходе, студент познает особенности профессионального взаимодействия;
3) студент убеждается, что приемы программирования ценны не сами по себе, а тем, насколько успешно они служат конкретным целям, в данном случае учебным и научным.

Главное же преимущество подобного вида учебной работы состоит в комплексном закреплении знаний как по изучаемой дисциплине, так и в области компьютерных технологий.

Литература

1. Полат Е.С. Метод проектов. http://www.ioso.ru/distant/project/
2. Гурский Д.А. Flash MX 2004 и ActionScript 2.0: обучение на примерах. М.: Новое знание, 2004. 446 с.
3. Потеев М.И. Концепции современного естествознания: Учебник для студентов вузов. СПб, 1999. 350 с.
4. Дубнищева Т.Я. Концепции современного естествознания: Учебник для студентов вузов. М., 2001. 827с.

ТЕХНОЛОГИЯ АВТОМАТИЗИРОВАННОГО ПОСТРОЕНИЯ ЛОГИЧЕСКИ НЕПРОТИВОРЕЧИВОГО УЧЕБНОГО КУРСА НА ОСНОВЕ НЕСТРУКТУРИРОВАННОГО БАНКА ТЕКСТОВ

А.А. Владыкин
Научный руководитель – С.Е. Столяр

Предложена технология автоматизированного построения учебного курса на основе имеющегося банка текстов. Описано применение технологии в сети Интернет. Возможность ее практического использования продемонстрирована на примере действующего сайта учебного назначения.

Введение

Текст учебного курса (учебного пособия) обычно состоит из отдельных блоков – лекций или уроков. Вполне естественной практикой является последовательное расположение этих блоков как в рамках учебного пособия, так и при «озвучивании» курса в регулярных занятиях. Можно, однако, предположить, что написание отдельных блоков происходит отнюдь не в порядке будущего оглавления. Причин тому немало: во-первых, курс может меняться со временем, отдельные блоки подвергаются ревизии, во-вторых, уроки могут быть написаны разными авторами, и т.д.

Логичное и последовательное изложение материала в рамках учебного курса, сформированного из подобных блоков (текстов лекций, уроков), неявно предполагает решение следующих задач.

(1) Выбор из доступного банка ограниченного набора текстов, которые затем будут включены в учебный курс.
(2) Согласование терминологии, используемой в разных текстах.
(3) Группировка близких по тематике текстов для высокоуровневого деления курса на темы (главы).
(4) Линейное упорядочение множества текстов в рамках каждой темы.

На обнаружение допущенных ошибок может уйти немало усилий, причем для редакционного вычитывания курса или его частей нередко привлекается посторонняя помощь, поскольку автор не вполне может полагаться на свой «замыленный» глаз. Мы предлагаем в определенной степени автоматизировать как решение упомянутых задач, так и проверку логической непротиворечивости результата [1].

Основная часть

Дальнейшие наши рассуждения будут основываться на критерии логической непротиворечивости учебного курса, который мы предлагаем ввести следующим образом: *любое понятие (термин), используемое в некотором тексте курса, должно быть определено в этом же тексте или в одном из предыдущих текстов*.

Критерий представляется простым и очевидным, однако с ним связана масса трудностей (часто технических), возникающих у автора/редактора курса. Нетрудно представить себе такие ситуации.

(1) Текст A использует понятие W_B, введенное в тексте B, следующем далее в курсе. Автору необходимо перенести урок в соответствующее место курса либо пересмотреть изложение материала.
(2) Тексты A и B написаны в разное время (может быть, даже разными авторами) и оперируют одним и тем же понятием, но под разными именами. Например, структура данных LIFO (last in – first out) известна под названиями «стек» и «магазин», причем оба названия широко используются. Авторам необходимо отслеживать подобные случаи и согласовывать применяемые в разных текстах названия либо явно указывать на тождественность понятий.

(3) В тексте A введено новое понятие W_A, в тексте B на его основе вводится понятие W_B, наконец, в тексте C из W_B выводится W_A. Автор должен вовремя обнаруживать и разрывать такие логические «петли».

Введенный нами критерий логической непротиворечивости позволяет убедиться в отсутствии подобных ошибок в учебном курсе. Он же позволяет алгоритмически строить учебный курс из имеющихся текстов.

Итак, каждому тексту L_i сопоставим его собственный набор *ключевых слов* $W(L_i)$, являющийся подмножеством общего тезауруса учебного курса. Каждое ключевое слово является словом или словосочетанием, именующим используемое в данном тексте понятие. Далее, каждое множество $W(L_i)$ разделим на две непересекающиеся части:

(1) подмножество $R(L_i)$ ключевых слов, используемых в тексте L_i, но не определяемых в нем (показывает, что необходимо знать, чтобы понять излагаемый материал);

(2) подмножество ключевых слов $P(L_i)$, определяемых в данном тексте (показывает, что становится известно после ознакомления с данным текстом).

Используя эти обозначения, запишем критерий логической непротиворечивости учебного курса следующим образом: *если для некоторых текстов* L_i *и* L_j *множество* $P(L_i) \cap R(L_j)$ *не пусто, то текст* L_i *должен быть расположен в курсе перед текстом* L_j.

Введенный формализм позволяет алгоритмизировать построение логически непротиворечивого курса.

Представим отдельные тексты вершинами графа (здесь мы не претендуем на новизну решения, достаточно сослаться на [2, с. 10]). Любые два текста L_i и L_j, для которых выполняется $P(L_i) \cap R(L_j) \neq \varnothing$, соединим дугой, направленной от L_i к L_j. В результате получим некоторый ориентированный граф (см. рисунок).

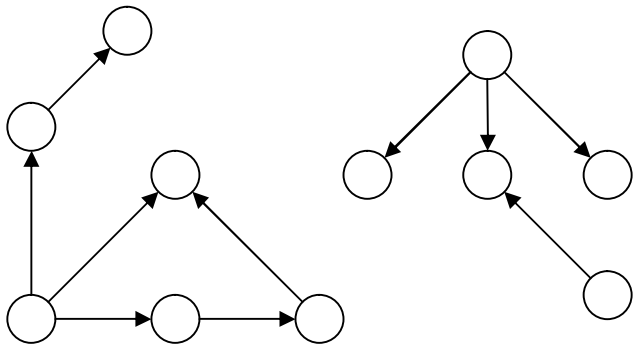


Рис. Пример графа, построенного на основе текстов учебного курса

Анализ построенного графа позволяет выявить структуру учебного курса.

Во-первых, граф может распадаться на несколько компонент. Наличие нескольких компонент связности говорит о том, что в учебном курсе присутствуют независимые разделы, которые логично оформить в виде отдельных глав, причем на порядок их следования ограничения не накладываются. В случае лекционного курса соответствующие разделы можно читать параллельно, например, в виде факультативов.

Во-вторых, применение алгоритма топологической сортировки к каждой компоненте связности в отдельности позволяет построить требуемый линейный порядок на множестве текстов учебного курса. Если в результате не будут выявлены циклы, то это

будет означать, что искомый порядок установлен. В противном случае обнаруживаются логические ошибки, которые ждут своего устранения.

Применение технологии ключевых слов в Интернете

Предложенная технология может быть эффективно применена для структуризации коллекции статей учебного содержания. Примером является раздел «Теория» нашего сайта «Дискретная математика: алгоритмы» [3].

Во-первых, сопоставление web-страницам наборов ключевых слов – хорошая основа для создания поисковой системы.

Во-вторых, статьи снабжаются рекомендациями по дальнейшему чтению. Рекомендации содержат следующую информацию.

(1) Ссылки на статьи, к которым следует обратиться, если в прочитанной статье обнаружились неизвестные пользователю термины. Набор ссылок является результатом поиска статей, вводящих используемые в данной статье термины.
(2) Ссылки на статьи, где содержится дальнейшая информация по интересующей пользователя тематике. Набор ссылок является результатом поиска статей, использующих вводимые в статье термины.

Такие рекомендации являются аналогом стандартного для Интернет-магазинов сервиса «Пользователи, заказывающие этот товар, также покупают…». Ссылки на другие товары выбираются по результатам анализа накопленной статистики продаж средствами data mining (используются сложные и трудоемкие алгоритмы). Однако на пути к созданию на сайте учебного содержания сервиса «Пользователи, читающие эту статью, также смотрят…» встают следующие проблемы:

(1) идентифицировать пользователей, например, путем введения регистрации и процедуры входа/выхода, что кажется излишним;
(2) отличать просто открытие страницы от ее прочтения, ведь содержимое могло не заинтересовать пользователя и оказаться бесполезным;
(3) иметь значительные аппаратные и вычислительные ресурсы для хранения и анализа накопленной статистики.

Предложенная система рекомендаций по дальнейшему чтению проста в реализации и дает необходимую пользователям информацию.

Выделение ключевых слов из текста

Обратимся теперь непосредственно к построению наборов ключевых слов. Эта задача также автоматизируется, если воспользоваться простой программой-экстрактором. Такая программа должна выбрать все слова из текста, затем ранжировать слова по частоте их появления в тексте (полученное распределение является распределением Зипфа [4]) и, наконец, выделить диапазон значимых слов. Окончательную выборку множества ключевых слов из полученного набора значимых слов полностью перепоручить программе уже вряд ли возможно, но, поскольку оставшийся набор значимых слов невелик, то пользователь (т.е. автор/редактор текста), будучи хорошо знаком с контекстом, легко завершит этот этап обработки.

Результаты тестирования

Тестирование обсуждаемой технологии проведено на наборе текстов незавершенного учебного курса «Введение в алгоритмику», размещенного в свободном доступе на сайте IPS [5].

Первый тест состоял в проверке выполнения критерия логической непротиворечивости при исходном порядке текстов курса. Проверка выявила три нарушения критерия:

(1) урок «От структуры данных к алгоритму» главы A оперирует термином *система счисления*, который вводится лишь в главе B далее по тексту;

(2) урок «Кучи. Основные понятия. Операция добавления элемента в кучу» главы G опирается на знания читателя о стеке, однако понятие *стек* введено позже в главе J;

(3) урок «Простые обменные сортировки вектора» использует понятия *лексикографический порядок* и *антилексикографический порядок*, не определенные нигде в курсе.

Подобные нарушения критерия непротиворечивости являются логическими ошибками курса, желательно их ликвидировать.

Следующий тест заключался в поиске повторных определений, когда определение одного понятия встречается в курсе несколько раз. Такие случаи были обнаружены:

(1) понятие *рекуррентное соотношение* вводится сначала в уроке «От структуры данных к алгоритму» главы A, затем в уроке «Понятие задачи и подзадачи» главы G;

(2) понятие *алфавит* появляется в уроке «Системы счисления» главы B, затем повторно вводится в уроке «Коды и сжатие данных (1)» главы N.

Наличие повторных определений не обязательно является ошибкой, однако автору следует обратить особое внимание на поддержание согласованности этих определений.

В рамках дальнейшего тестирования был построен граф, представляющий структуру учебного курса. Граф распался на три компоненты связности: первая компонента соответствует главе A, вторая – главе Z, третья включила в себя тексты всех остальных глав. Разрастание третьей компоненты связности объясняется наличием тесных логических взаимосвязей между текстами, например, повсеместным использованием массивов (соответствующее ключевое слово входит в R-множества значительной части уроков) и некоторых других базовых понятий.

Наконец, на текстах каждой компоненты связности был построен новый линейный порядок, удовлетворяющий критерию логической непротиворечивости. При этом термины *лексикографический порядок* и *антилексикографический порядок* были исключены из текста, поскольку их наличие делало желаемое упорядочение невозможным.

Заключение

Сопоставление текстам набора ключевых слов является традиционным инструментом. Предлагаемая нами модификация такого набора с выделением двух поднаборов существенно расширяет его роль, допуская новую функциональность. Во-первых, на основе этой информации можно связать тексты в логически непротиворечивый учебный курс, а также проверить на логическую непротиворечивость готовое оглавление. Во-вторых, перспективным является применение предложенной технологии на сайтах учебного содержания с большим количеством статей.

Литература

1. Столяр С.Е., Владыкин А.А. Автоматизация проверки логической непротиворечивости учебного курса. // Труды XII Всероссийской научно-методической конференции Телематика'2005. 6–9 июня 2005 года, Санкт-Петербург. СПб: СПбГУ ИТМО, 2005. Т. 1, С. 323–324.

2. Кристофидес Н. Теория графов. Алгоритмический подход. М.: Мир, 1978.
3. Владыкин А.А., Столяр С.Е. Новый web-ресурс по алгоритмам дискретной математики. // Труды XI Всероссийской научно-методической конференции Телематика'2004. СПб: СПбГУИТМО, 2004. Т. 2, С. 611–612.
4. Мидоу Ч. Анализ информационно-поисковых систем. Введение для программистов. М.: Мир, 1970.
5. Васильев В.Н., Казаков М.А., Парфенов В.Г., Столяр С.Е. Проблемы подготовки профессиональных программистов и дистанционное обучение в Интернет-школе СПбГИТМО (ТУ). // Дистанционное обучение. Проблемы и перспективы взаимодействия вузов Санкт-Петербурга с регионами России. Материалы II межрегиональной научно-практической конференции. СПб, 1999. С. 45–48.

О НЕОБХОДИМОСТИ ПОДГОТОВКИ В ОБЛАСТИ ЭРГОНОМИКИ БУДУЩИХ ПРОЕКТАНТОВ ЭЛЕКТРОННЫХ ОБРАЗОВАТЕЛЬНЫХ РЕСУРСОВ

М.В. Хлопотов

Научный руководитель – кандидат технических наук, профессор М.И. Потеев

В статье рассматриваются аспекты проектирования пользовательского интерфейса электронных образовательных ресурсов, показывается необходимость подготовки инженеров специальности «Информационные технологии в образовании» в области эргономики.

Введение

Согласно Государственному образовательному стандарту специальности «Информационные технологии в образовании» выпускнику, освоившему соответствующую образовательную программу, присваивается квалификация «инженер».

Инженер, по определению, является проектировщиком, в данном случае – проектировщиком информационных образовательных ресурсов. С этой точки зрения центральной является дисциплина «Проектирование информационных систем в образовании», которая относится к блоку специальных дисциплин.

К основным дидактическим единицам этой дисциплины относятся:

А общая характеристика процесса проектирования информационных систем в образовании (ИСО):
- а структура информационно-логической модели ИСО;
- б разработка функциональной модели; исходные данные для проектирования; особенности архитектуры обучающих программ;
- в обучающая, тренирующая, контролирующая подсистемы;
- г разработка модели и защита данных;
- д разработка пользовательского интерфейса; разработка проекта распределенной обработки;

Б структура программных модулей;
- а разработка алгоритмов;
- б логический анализ структур ИСО;
- в анализ и оценка производительности ИСО;
- г управление проектом ИСО;
- д проектная документация;
- е инструментальные средства проектирования ИСО;
- ж типизация проектных решений;
- з графические средства представления проектных решений; эксплуатация ИСО.

Анализ этого блока показывает, что в нем абсолютно не затронуты вопросы эргономики. Покажем целесообразность введения в эту дисциплину раздела, посвященного эргономическим аспектам проектирования информационных систем в образовании.

Эргономические аспекты проектирования электронных образовательных ресурсов

В современной мировой практике при проектировании и оценке эффективности информационных образовательных ресурсов первостепенное значение уделяется их эргономическим характеристикам. На смену расплывчатым терминам «дружественность пользователю», «дружественный интерфейс» пришла теоретически и практически обоснованная концепция «юзабилити» (от англ. «usability» – «удобство использования», «пользовательская пригодность»). Термин «пользовательская пригодность» ин-

терфейса является комплексным критерием оценки эргономичности и во многом определяется соответствием интерфейса задачам пользователя, его потребностям, а также возможностям и ограничениям.

Международный стандарт ISO 9241-11 «Guidance on Usability» определяет пользовательскую пригодность как «степень, в которой продукт может быть использован определенными пользователями при определенном контексте для достижения определенных целей с должной эффективностью, продуктивностью и удовлетворенностью».

В этом стандарте приводятся основные факторы, определяющие удобство использования программного продукта:

- эффективность – точность и полнота при достижении пользователями цели;
- продуктивность – количество психических, физических и временных ресурсов, необходимых для достижения пользователями своих целей с нужной точностью и полнотой;
- удовлетворенность пользователя – субъективное ощущение комфорта и удобства при пользовании программным продуктом;
- адекватность контексту – соответствие системы характеристикам ее пользователей, их целям, задачам, оборудованию, социальным характеристикам среды, в которой используется программный продукт.

В настоящее время общепризнанной стала необходимость включения в жизненный цикл разработки программного обеспечения работ по проектированию и тестированию удобства использования.

Стоит заметить, что удобство использования любого программного продукта связано, прежде всего, с качеством пользовательского интерфейса. Перед разработчиками всех типов электронных образовательных ресурсов встает проблема проектирования пользовательского интерфейса, позволяющего обеспечить эффективное использование информационных систем в процессе обучения.

Разработчики программных комплексов зачастую склонны рассматривать функциональность системы отдельно от ее пользовательского интерфейса. При этом предполагается, что пользовательский интерфейс является своего рода дополнением к функциональности системы. Со своей стороны, пользователи программ, как правило, не разделяют функциональность и пользовательский интерфейс. Для пользователей именно пользовательский интерфейс является программой.

Пользовательский интерфейс часто понимают только как внешний вид программы. Однако на деле пользователь воспринимает через интерфейс всю систему в целом, а значит, такое понимание является слишком узким. В действительности пользовательский интерфейс включает в себя все аспекты дизайна, которые оказывают влияние на взаимодействие пользователя и системы.

Пользовательский интерфейс состоит из множества составляющих, таких как:

- набор задач пользователя, которые он решает при помощи системы;
- используемая системой метафора;
- элементы управления системой;
- навигация между блоками системы;
- визуальный (и не только) дизайн экранов программы.

Проблему проектирования пользовательского интерфейса для электронных образовательных ресурсов, обеспечивающих условия для целенаправленной познавательной деятельности обучающегося, решают разработчики и проектировщики электронных образовательных ресурсов, т.е. инженеры специальности «Информационные технологии в образовании». Чтобы разрабатываемые электронные образовательные ресурсы стали эффективным средством обучения, необходимо учитывать эргономические аспекты проектирования таких ресурсов.

Заключение

В подготовке инженеров специальности «Информационные технологии в образовании» ключевой является дисциплина «Проектирование информационных систем в образовании». В качестве одной из дидактических единиц этой дисциплины выступает «разработка пользовательского интерфейса». Необходимо дополнить и расширить содержание этой дидактической единицы с точки зрения эргономических аспектов в проектировании электронных образовательных ресурсов.

Литература

1. Кузнецов А.М., Мартынов В.В. Требования к графическому дизайну и юзабилити образовательных порталов / Сб. научн. ст. «Интернет-порталы: содержание и технологии». Вып. 1. ГНИИ ИТТ «Информика». М.: Просвещение, 2003. С. 365–420.
2. Потеев М.И. Развитие профессионально-педагогического образования в университетах технического типа. Монография. СПб, 2005. 440с.

КОМПЬЮТЕРИЗИРОВАННЫЙ УЧЕБНО-ЛАБОРАТОРНЫЙ КОМПЛЕКС ПО ДИСЦИПЛИНЕ «ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ СЕТИ ИНТЕРНЕТ»

К.Ю. Янсон
Научный руководитель – Д.М. Гриншпун

В работе представлен компьютеризированный лабораторный комплекс, обеспечивающий проведение практических занятий при подготовке технического персонала, обслуживающего локальные вычислительные сети с подключением к Интернет.

Введение

Подготовка специалистов по техническому обеспечению вычислительных сетей определена государственным образовательным стандартом начального профессионального образования на профессию «Оператор электронно-вычислительных машин. Компьютерные сети» (ОСТ 9 ПО 02.1.9-2002). Область применения специальности – обслуживание вычислительной техники, в том числе сетей, от локальных до глобальных. Уровень квалификации – рабочие.

Квалификационная характеристика операторов ЭВМ весьма широка, набор знаний и умений разнообразен и достаточно глубок. Охватывает он и Интернет-технологии, широкое распространение которых, их развитие и внедрение в практически повсеместную деятельность определяют необходимость не просто владеть навыками обслуживания технической базы, но и уметь профессионально работать с сетевыми ресурсами. В этом заключается противоречивость требований, предъявляемых к специальности: с одной стороны, подготовка специалистов уровня рабочих, а с другой – явно интеллектуальный характер будущего труда.

Это противоречие распространяется, как следствие, и на учебную литературу по рассматриваемой специальности, и на учебно-методическую базу. Так, техническая литература, как правило, рассчитана либо на пользовательский уровень, что не может обеспечить качественную подготовку специалистов, либо на среднее или высшее образование. Еще неопределенней ситуация с технической базой – анализ предусмотренных стандартом (примерной рабочей программой) практических и лабораторных работ приводит к выводу о необходимости обеспечения изучения пользовательских программ и, в незначительной степени, средств технической диагностики. В то же время, предусмотренное освоение программирования не может рассматриваться серьезно, так как подготовка рабочих практически не включает в себя изучения математических и физических наук. Поэтому на первый взгляд очевидная и несложная задача подготовки операторов при серьезном подходе оказывается отнюдь не простой. Особенно это касается приобретения практических умений и навыков – поскольку речь идет о подготовке специалистов на уровне рабочих, требуется обеспечить не только теоретические знания, но, что значительно более важно, речь прежде всего должна идти о практике. Практические знания могут быть приобретены посредством выполнения набора учебных лабораторных работ, для которых необходим соответствующий компьютеризированный комплекс. В настоящей работе представлен такой комплекс, созданный в институте информационных технологий «СЕГРИС» для профессиональной подготовки операторов ЭВМ по специализации «Вычислительные сети».

Компьютеризированный комплекс

Реализуемая учебная программа состоит из трех разделов:

- работа с ресурсами Интернет;

- основы Web-дизайна;
- техническое обеспечение Интернет.

Если первые два раздела являются традиционными, т.е. освоенными большим количеством учебных заведений разного уровня и входящими в программу многих информационно-технологических специальностей, то для третьего требуются специализированный комплекс аппаратно-программных средств. Его структура определяется следующей учебной программой:

- технологии WWW – рассматриваются способы создания Web-страниц и протокол HTTP;
- аппаратные сети передачи данных – изучается технология Ethernet и ее реализации;
- протоколы Интернет – рассматриваются принципы адресации в сети, взаимодействие компьютеров в объединенной сети, транспортные протоколы и способы исследования сети;
- сетевые ОС – на примере многопользовательской ОС рассматриваются права доступа к файлам и папкам, политики безопасности;
- приложения Интернет – изучаются серверы и клиенты доменной системы имен, а также клиенты электронной почты, протоколы SMTP и POP3;
- проектирование корпоративных сетей – на практике рассматриваются вопросы выделения адресов компьютерам в локальной сети с учетом дальнейшего подключения их к глобальной сети Интернет, конфигурирование серверов DHCP и DNS;
- безопасность в объединенной сети – изучаются способы ограничения доступа к компьютерам, подключенным к сети с помощью брандмауэра, основы шифрования данных для безопасной передачи по открытым сетям.

Соответственно, должна обеспечиваться реализация следующих практических и лабораторных работ.

1. Технологии WWW.
    1.1. Передача файлов.
    1.2. Размещение страниц на Web-сервере.
    1.3. Размещение php-сценариев на Web-сервере.
2. Аппаратные средства передачи данных. Определение аппаратного адреса.
3. Протоколы Интернет.
    3.1. Исследование сети.
    3.2. Анализ сетевого трафика.
4. Сетевые ОС.
    4.1. Управление пользователями.
    4.2. Права доступа к файлам и папкам.
    4.3. Политики безопасности.
5. Приложения Интернет.
    5.1. Рекурсивные запросы.
    5.2. Протокол SMTP.
    5.3. Протокол POP3.
6. Проектирование корпоративных сетей.
    6.1. Конфигурирование сетевых параметров.
    6.2. Конфигурирование DHCP-сервера.
    6.3. Конфигурирование DNS-сервера.
7. Безопасность в объединенной сети.
    7.1. Правила для работы в сети Интернет.
    7.2. Асимметричное шифрование данных.

Разработанный компьютеризированный комплекс состоит из аппаратного и программного обеспечения, распределенного в корпоративной сети (см. рис.). Он включает в себя три учебных сервера и рабочие станции учебных классов.

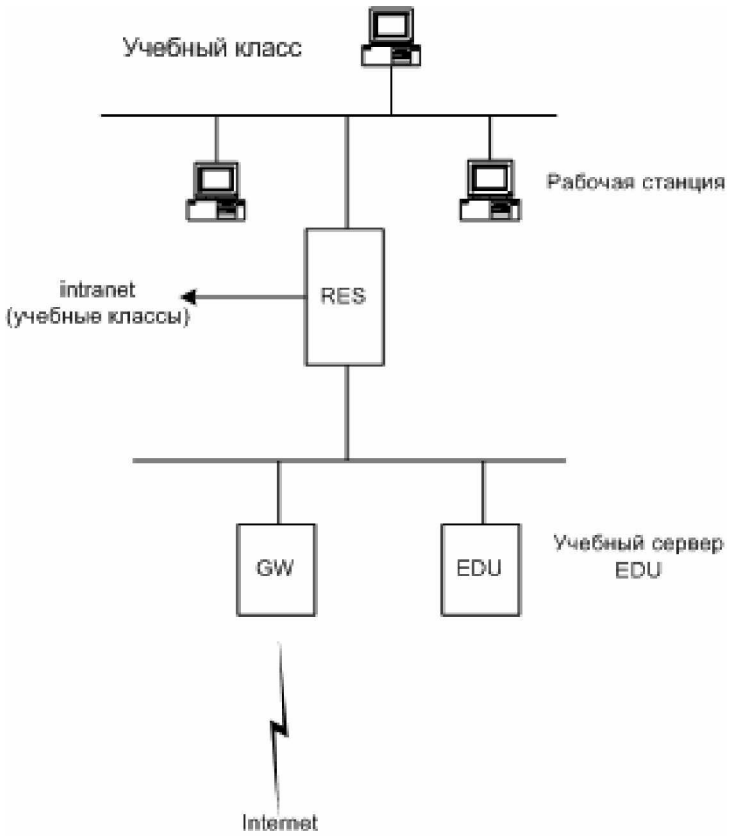


Рис. Блок-схема комплекса

Сервер GW предназначен для разграничения доступа учебных групп в глобальную сеть Интернет и учета потребленного трафика. Он же позволяет с учебной целью регулировать характеристики канала передачи данных (скорость и задержки, процент потерь пакетов) для анализа работы протокола надежной доставки сообщений. Таким образом, реализуются основные практические работы (3.1, 3.2, 5.1, 5.2, 5.3), а также любые иные, связанные с обращением к серверам глобальной сети.

Сервер RES соединяет различные учебные классы, разделяя корпоративную сеть на подсети.

Сервер EDU используется для автоматической выдачи адресов компьютерам в учебных классах. Он содержит следующее программное обеспечение:

- HTTP-сервер с поддержкой PHP и MySQL;
- сервер баз данных MySQL;
- FTP-сервер;
- сервер удаленного доступа (Telnet и SSH).

Кроме того, на нем хранятся установочные версии используемого программного обеспечения. К серверу EDU учащиеся имеют свободный доступ, поэтому на случай возможного повреждения программного обеспечения предусмотрено его быстрое восстановление с резервных копий. С его помощью реализуются практические работы 1.1, 1.2, 1.3, на нем изучаются почтовые протоколы POP3 и SMTP и протокол имен DNS. При проведении практических работ по проектированию корпоративных сетей он может быть настроен с моделируемыми параметрами.

Серверы GW и EDU с помощью дополнительных настроек маршрутизации имитируют различные ошибки, например, ошибку «превышение срока жизни пакета».

Отличительной особенностью комплекса является то, что учащиеся имеют полный неограниченный доступ к учебным компьютерам, сети, а также к серверу EDU. При возникновении программных сбоев, которые неизбежны при такой организации

работ, с помощью специальной программы обеспечено быстрое автоматизированное восстановление программного обеспечения сервера и рабочих мест практически без участия преподавателя (для этого достаточно установить в дисковод восстановительную дискету и нажать кнопку Reset). Это не только развивает самостоятельность учащихся, но и нередко провоцирует не предусмотренные учебной программой случайные ситуации, что стимулирует развитие нетиповых умений.

Заключение

Представленный комплекс разрабатывался в 2001/2002 уч. г., экспериментальная отработка осуществлялась в 2002/2003 уч. г., с 2003/2004 уч. г. он введен в основной учебный процесс института информационных технологий «СЕГРИС», а с текущего года он внедряется на факультете СПО СПбГУ ИТМО. В результате его применения повысилась наглядность предоставляемого материала, за счет раскрепощенности действий усилилась мотивация, что положительно сказалось на успеваемости учащихся. Более 40% учащихся могут выполнять алгоритм нетиповой деятельности, что соответствует третьему уровню усвоения материала согласно государственному образовательному стандарту.

О ПРОЕКТЕ ИНТЕРНЕТ-ПОРТАЛА В ОБЛАСТИ ФОТОНИКИ, ОПТОИНФОРМАТИКИ И НАНООПТИКИ

Ю.В. Шлюжайте
Научный руководитель – доктор физико-математических наук, профессор Ю.Л. Колесников

Актуальной задачей на сегодняшний день является создание мощного Интернет-ресурса в области фотоники, оптоинформатики и нанооптики с целью удовлетворения профессиональных нужд специалистов и исследователей всех уровней. В качестве решения этой актуальной задачи рассматривается реализация проекта Интернет-портала в области фотоники, оптоинфоратики и нанооптики.

Введение

В информационном обществе актуальной задачей является получение студентом, молодым специалистом, ученым необходимой для его работы или исследований информации, которая к тому же соответствовала бы современному развитию того направления, в котором он работает. Особенно это относится к студентам, которые только начинают свою исследовательскую и трудовую деятельность: они пока не освоились в огромной массе информации и порой не знают, как найти именно то, что им нужно. Кроме того, современные студенты – начинающие специалисты и исследователи – по большей части привыкли черпать информацию из глобальной сети Интернет, и тут возникает другая проблема. Несмотря на огромное количество разнообразных Интернет-ресурсов, требуется потратить много времени и посетить порой не один десяток ресурсов, чтобы выйти на ту информацию, которая представляет действительную ценность на данный момент. А иногда так и не удается найти необходимую информацию во всемирной паутине.

Положение в области фотоники, оптоинформатики и нанооптики

Что касается таких научно-технических направлений, как фотоника, оптоинформатика и нанооптика, развитие которых на современном этапе происходит просто семимильными шагами, то наиболее оптимальным способом «обнародования» последних результатов исследований и достижений в данной отрасли является использование Интернет, а именно публикация этих результатов в рамках различных Интернет-ресурсов. Такое представление информации научному сообществу является наиболее обоснованным по той причине, что уже сразу после опубликования в Интернет она становится доступной любому заинтересованному пользователю из любой точки мира, конечно, при условии, что он знает, где именно искать эту информацию. В то же время на издание и распространение информации традиционными способами уходит не один день и даже не неделя, по прошествии которых информация может утратить свою актуальность.

На сегодняшний день в Интернет существует достаточно большое количество ресурсов, которые так или иначе связаны с фотоникой, оптоинформатикой и нанооптикой. Пользователям российского Интернет наиболее знакомы следующие: Optics.org, являющийся первым Интернет-ресурсом в области оптики, Photonics.com, принадлежащий американской компании Laurin Publications, которая издает специализированные журналы в области фотоники и Nanotechweb.org, освещающий вопросы нанотехнологий. Но большинство из них принадлежит коммерческим компаниям, занимающимся разработками и производством в данных научно-технических направлениях. В соответствии с этим основной функцией упомянутых Интернет-ресурсов является продвижение продукции и услуг тех компаний, которым они принадлежат. Но, кроме этого,

данные Интернет-ресурсы выполняют и другие функции, предоставляя заинтересованным пользователям доступ к:

- результатам исследований и разработок;
- научно-технической документации и публикациям специалистов;
- своим базам данных;
- вычислительным ресурсам и инструментам обработки научных данных;

и многому другому [1].

Таким образом, можно сделать вывод, что в Интернет существует много ресурсов, где как опытные, так и начинающие специалисты в области фотоники, оптоинформатики и нанооптики могут черпать научно-техническую информацию. Но при этом нерешенным вопросом остается вопрос о поиске необходимой информации: ведь, как уже упоминалось выше, пользователи могут потратить немало времени прежде, чем выйти на Интернет-ресурс, содержащий нужную им информацию. В целом этот вопрос решают путем создания тематических Интернет-порталов, включающих в себя рубрикатор со множеством ссылок на другие Интернет-ресурсы, которые непосредственно могут предоставить необходимые информацию и сервисы. В российском сегменте Интернет существуют такие порталы, разработанные в рамках Федеральной целевой программы «Развитие единой образовательной информационной среды (2001–2005 годы)» [2]. В систему федеральных образовательных порталов входят Интернет-порталы, посвященные различным областям знания – естественнонаучные дисциплины (математика, физика, химия), информационные технологии, гуманитарные науки и т.п. К сожалению, несмотря на то, что развитие этих научно-технических направлений происходит интенсивно, а пользователям просто необходим ориентир в информационных ресурсах, посвященных данной области знания, портал в области фотоники, оптоинформатики и нанооптики на настоящий момент времени не существует. Таким образом, актуальность создания Интернет-портала в области фотоники, оптоинформатики и нанооптики несомненна. Поэтому было принято решение о восполнении данного пробела в информационной образовательной среде посредством разработки методики построения Интернет портала в области фотоники, оптоинформатики и нанооптики, а затем непосредственной реализации данного ресурса в сети Интернет.

Проект реализации нового Интернет-портала

Процесс реализации данного проекта был разбит на несколько этапов. Первый этап направлен, как это принято, на определение как основных, так и второстепенных целей и задач Интернет-портала в области фотоники, оптоинформатики и нанооптики. Второй этап предусматривает обзор существующих Интернет-порталов с целью выявления их методики построения с дальнейшим анализом их плюсов и минусов, которые необходимо учитывать при разработке методики построения нового Интернет-портала в соответствии с целями и задачами портала. На третьем этапе в соответствии с целями и задачами формируются научно-методические требования к наполнению, сервисам и защищенности Интернет-портала, а также разрабатывается структура портала и алгоритм его построения. На четвертом этапе происходит непосредственное наполнение основных разделов портала, а на пятом – его размещение в глобальной сети Интернет и оценка его использования заинтересованными пользователями и степень их удовлетворенности ресурсом.

На этапе определения основных целей и задач нового проекта с небольшой корректировкой обозначены требования, предъявляемые к любому информационно-образовательному Интернет-порталу, а именно:

- обеспечение доступа пользователей к хранимой на портале информации различного типа (новости, события, публикации, интерактивные демонстрации и многое другое);
- обеспечение доступа к различным общим и специализированным (средства автоматизации обработки результатов, расчетов и т.п.) сервисам;
- организация интерактивного взаимодействия пользователей друг с другом и с разработчиками (служба поддержки пользователей);
- обеспечение тренингов и образовательных возможностей в виртуальной среде;
- реализация контроля различной деятельности пользователей в рамках Интернет-портала.

При проведении обзора системы федеральных образовательных порталов и разработки методики построения нового Интернет-портала в качестве прототипа, как наиболее удачный с точки зрения реализации и наиболее подходящий к целям и задачам нового проекта, был принят Портал по информационно-коммуникационным технологиям.

В соответствии с обозначенными основными целями и задачами нового проекта разработана предварительная структурная схема Интернет-портала в области фотоники, оптоинформатики и нанооптики, которая представлена на рис. 1.

Рис. 1. Структурная схема Интернет-портала

В качестве пилотного проекта создан Интернет-ресурс «Фотоника, оптоинформатика и нанооптика» (см. рис. 2), в рамках которого представлен рубрикатор ресурсов по соответствующим научно-техническим направлениям. Кроме того, Интернет-ресурс содержит новостную ленту событий относительно данной тематики, представлены некоторые публикации, а также осуществлена обратная связь с пользователями. Целью данного проекта является выяснение реальных интересов и требований потенциальных пользователей, которые они могут предъявить к Интернет-порталу в данной области.

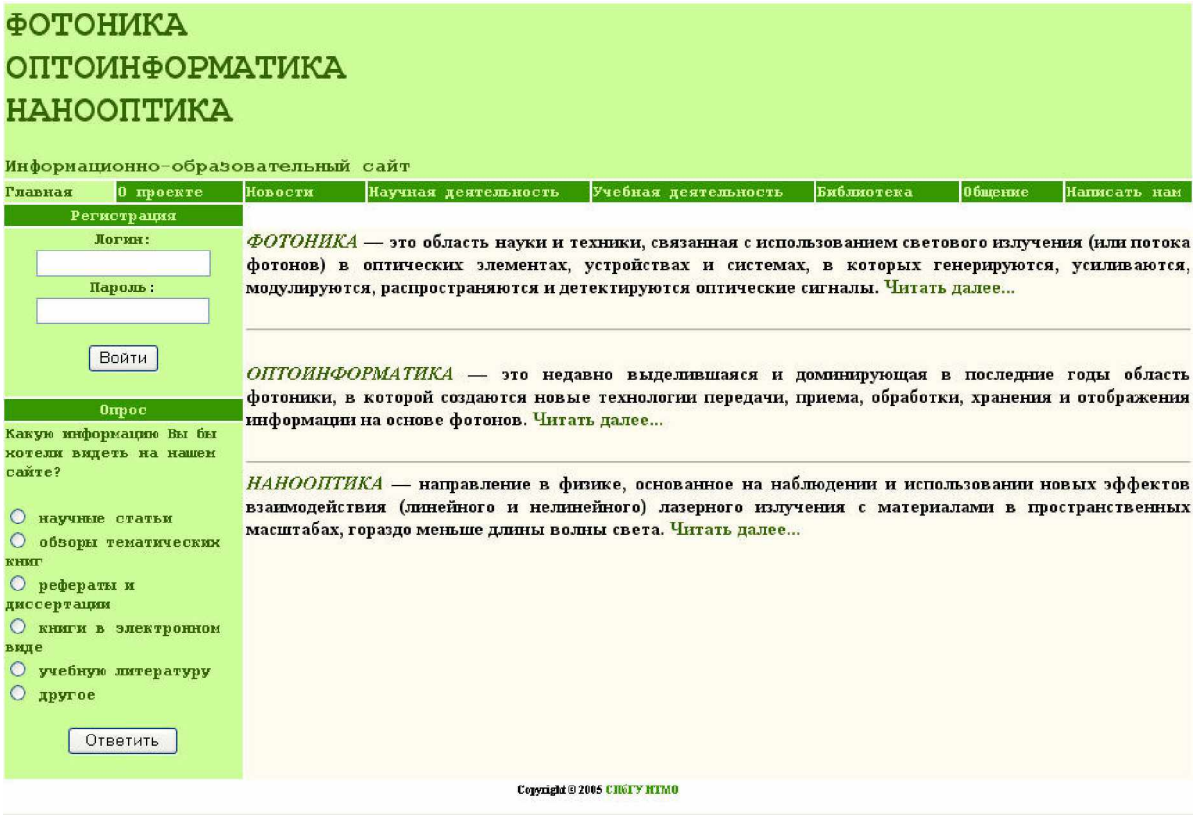


Рис. 2. Стартовая страница пилотного сайта

Заключение

Создание информационно-образовательных Интернет-ресурсов, поддерживаемых различными образовательными и исследовательскими учреждениями, выполняет важную роль как в информатизации образовательного и исследовательского процессов, так и в поддержании уровня образования на высоте. Реализация тематических порталов в различных областях знания в первую очередь помогает заинтересованным пользователям экономить время на поиск необходимой информации и научно-технической документации, что немаловажно в стремительно развивающемся мире высоких технологий.

Литература

1. Волков Д.И., Волков П.Д. Концепция проекта корпоративного научно-образовательного портала. / В сб. научных статей «Интернет-порталы: содержание и технологии». Вып. 2. / Редкол.: А.Н. Тихонов (пред.) и др.; ГНИИ ИТТ «Информика». М.: Просвещение, 2004. С. 94–130.
2. Тихонов А.Н., Иванников А.Д., Гридина Е.Г., Куракина Н.И., Симонов А.В., Чиннова И.И. Комплексный анализ системы федеральных образовательных порталов. / В сб. научных статей «Интернет-порталы: содержание и технологии». Вып. 2. / Редкол.: А.Н. Тихонов (пред.) и др.; ГНИИ ИТТ «Информика». М.: Просвещение, 2004. С. 192–226.

ВИРТУАЛЬНЫЕ ЛАБОРАТОРНЫЕ РАБОТЫ В СРЕДЕ СИСТЕМЫ ДИСТАНЦИОННОГО ОБУЧЕНИЯ

А.А. Волкова

Научный руководитель – кандидат технических наук, доцент А.В. Лямин

Статья содержит описание метода организации виртуальных лабораторных работ, предусматривающего возможность автоматизированной проверки результата выполнения.

Введение

Основной целью работы является разработка и внедрение технологии организации доступа к лабораторной установке, создание комплекса виртуальных лабораторных работ в системе дистанционного обучения Санкт-Петербургского государственного университета информационных технологий, механики и оптики по различным дисциплинам. Выполнение лабораторной работы должно заканчиваться представлением отчета, который может быть проверен автоматически. По результатам выполнения лабораторной работы в базу данных системы ДО заносится оценка.

При изучении рынка виртуальных лабораторных работ было выявлено, что большинство предлагаемых вариантов лабораторий ориентированы на проведение эксперимента, например, лабораторная работа по полупроводникам (http://www.pilab.ru/csi/AUK/Microelectr/lab.html). Главной особенностью нашей технологии является обеспечение автоматической проверки отчетов.

Виртуальная лаборатория является модулем системы дистанционного обучения. В общем случае виртуальная лаборатория представляет собой некую информационную среду, позволяющую проводить эксперименты без непосредственного доступа к объекту исследования. При этом эксперименты могут проводиться как с использованием математических моделей, так и с использованием удаленного доступа к изучаемому объекту [1].

Виртуальная лаборатория предоставляет возможность произвольной оценки производимых учеником действий, а также построения автором собственного алгоритма анализа результатов. Результатом выполнения работы может быть любая структура (схема, программа, набор чисел), которая в виде строки передается на сервер дистанционного обучения, а затем на сервер удаленного доступа, где автор размещает свой анализатор результата. Полученный отчет проверяется в автоматическом режиме. В частном случае результатом выполнения лабораторной работы может быть формальное описание какой-либо системы, которая оценивается по реакциям на эталонные воздействия [2].

Организация доступа к удаленной лабораторной установке

Была разработана и реализована схема организации доступа к лабораторной установке, изображенная на рис. 1. Мысленно ее можно разделить на три основных части – клиентское программное обеспечение, проверяющий сервер и необходимое оборудование кафедры, а также аппаратное и программное обеспечение центра дистанционного обеспечения, обеспечивающее корректную работу и взаимосвязь первых двух компонент. К ним относятся:

- программный интерфейс лабораторной работы, выполненный в виде *Java*-апплета;
- апплет-менеджер, загруженный через обозреватель на компьютер пользователя системы (обучающегося) и осуществляющий связь пользовательского интерфейса лабораторной работы с сервером дистанционного обучения;
- *Web*-сервер, обрабатывающий запросы пользователя на доступ к ресурсам системы;

- система управления базой данных (СУБД), которая обеспечивает функционирование базы данных системы;
- *Java*-приложение, управляющее лабораторной установкой;
- *Java*-сервлет, формирующий по запросу пользователя WWW-страницы на основе информации, поступающей от СУБД и Java-приложения, а также фиксирующий результаты работы пользователя с системой.

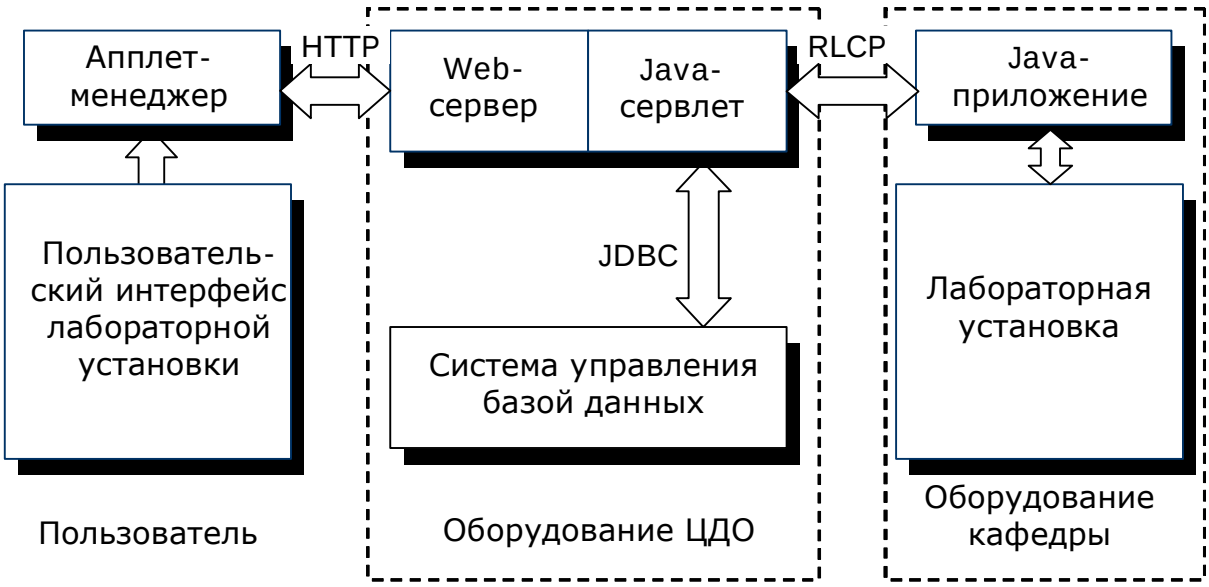


Рис. 1. Организация доступа к удаленной лабораторной установке

Пакет виртуального лабораторного комплекса

Пакет виртуального лабораторного комплекса должен содержать в себе zip-архив с откомпилированными class-файлами java-апплета виртуального стенда и проверяющего сервера, а также описания сценария лабораторной работы, заданий к ней, и описания лабораторного стенда [3].

Основным форматом для представления виртуальной лаборатории является язык XML. Были разработаны правила оформления XML-файлов, которые задаются при помощи Document Type Definition или DTD-описаний. При погружении в систему информации программа-анализатор автоматически проверяет правильность вводимой информации, основываясь на DTD-описаниях. В поставляемый пакет лабораторного комплекса входят три вида XML-файлов: сценарий лабораторной работы, описание заданий лабораторной работы и описание виртуальной лабораторной установки. Иерархия отношений этих описаний представлена на рис. 2.

Сценарий лабораторной работы включает в себя ряд заданий для выполнения, шкалу для оценивания результатов, которая будет использоваться при выставлении оценки, признак режима аттестации.

Задания лабораторной работы содержат группы тест-наборов для проверки результатов с указанием входных и ожидаемых выходных данных, указатель на виртуальную лабораторную установку и адрес сервера удаленного доступа, на котором расположен интерпретатор, анализирующий результат выполнения лабораторной работы.

Описание виртуальной лабораторной установки состоит из набора атрибутов для отображения, таких как высота и ширина апплета, а так же содержит адрес архива установки.

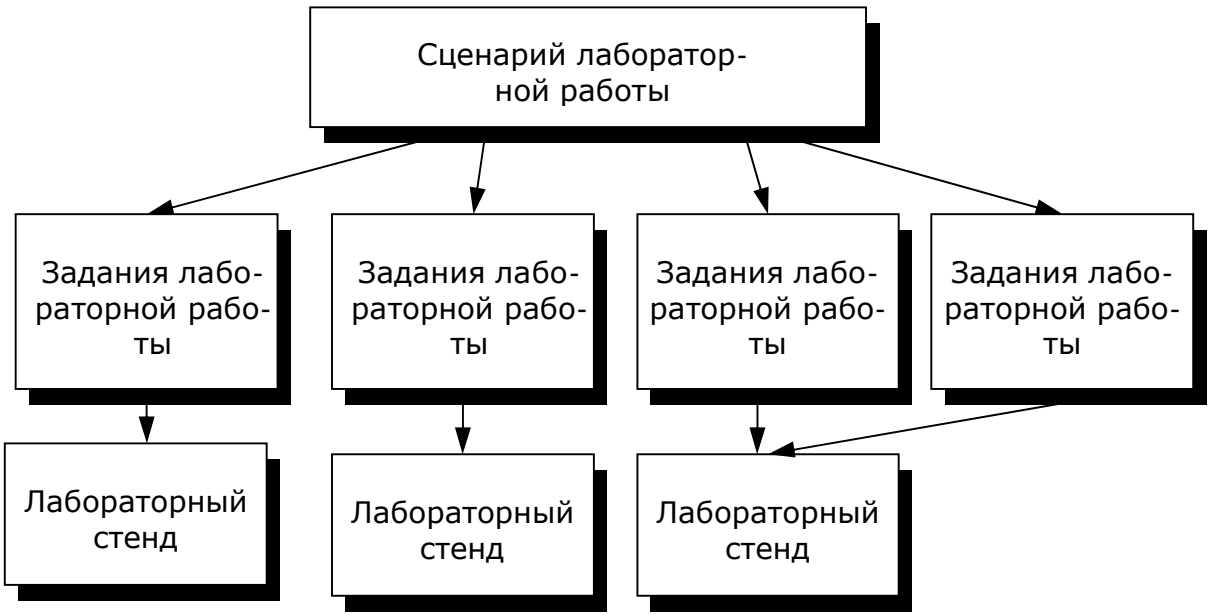


Рис. 2. Структура лабораторной работы

Кроме того, в приведенных XML-документах автор лабораторной работы задает ряд настраиваемых параметров, которые в дальнейшем будут учитываться в учебном процессе. К ним относятся:

- правила оценивания результатов выполнения лабораторной работы;
- ограничение по времени на выполнение задания студентом;
- режим выполнения;
- группы тест-наборов для проверки результатов;
- коэффициент сложности каждого тест-набора;
- число тест-наборов из группы, используемых для проверки;
- ограничение по времени на проверку тест-набора.

Необходимо отметить, что на разных уровнях структуры лабораторной работы настраиваемые параметры могут повторяться. Для избежания конфликта были сформулированы следующие положения: для числовых значений используются величины, указанные на нижних уровнях иерархии, а ограничивающие правила имеют больший приоритет на верхних уровнях.

В состав виртуальной лаборатории могут быть включены GIF-анимации и интерактивные FLASH-анимации. Авторский стиль лабораторной работы оформляется в соответствии со стандартом CSS. Для повышения уровня интерактивности виртуальной лаборатории допускается использование скриптов, написанных на языке JavaScript.

При разработке программного инструментария автор должен руководствоваться рядом правил, которые были выбраны для организации совместимости виртуального стенда и среды дистанционного обучения. Например, математическая модель лабораторной установки выполняется на языке Java 2 в режиме апплета. Базовым для классов, определяющих специфику той или иной лабораторной работы, должен быть класс `Base`. Он расширяет класс `Applet`. В классе `Base` должен быть объявлен один абстрактный метод `getResults()`, который необходимо переопределить в каждом классе-потомке класса `Base`. Этот метод возвращает данные лабораторной работы в виде строки. Формат строк произвольный, выбирается автором. При обработке результата эта строка будет инкапсулирована в протокол RLCP.

Каждая виртуальная лабораторная работа представляет собой обучающий комплекс, содержащий несколько компонентов:

- краткое описание и анализ теоретических аспектов изучаемого объекта, явления или процесса;
- описание приборов и оборудования, используемых для проведения исследований, их характеристики и порядок применения;
- виртуальный лабораторный стенд, который позволяет проводить исследование изучаемого объекта, явления или процесса по индивидуальной программе, обработку результатов и представление отчета.

Все компоненты виртуальной лаборатории размещаются в рамках одной страницы. На рис. 3 изображен пример лабораторной работы. В нижней части экрана расположен так называемый апплет-менеджер. Его роль состоит в осуществлении связи пользовательского интерфейса лабораторной работы с сервером дистанционного обучения. Визуально он состоит из полосы оставшегося времени и кнопки «Ответ готов». После нажатия на эту кнопку апплет-менеджер обращается к методу `getResults()`, забирает ответ студента, оформленный в виде строки, и передает на дальнейшую обработку.

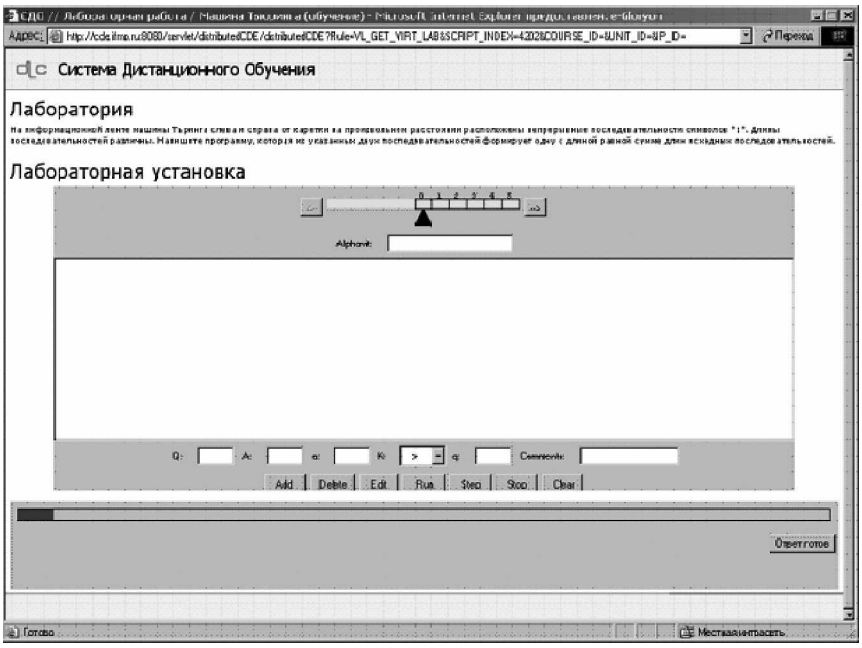


Рис. 3. Интерфейс лабораторной работы «Машина Поста»

Протокол управления лабораторией

Протокол управления лабораторией RLCP базируется на протоколе TCP. Взаимодействие происходит по следующей схеме: сначала устанавливается TCP-соединение с сервером, потом передается запрос клиента, далее сервер проверки возвращает ответ, после чего происходит разрыв соединения.

Запрос клиента генерируется процедурой базы данных и содержит следующие компоненты:

- строка состояния,
- поля заголовка,
- пустая строка,
- тело запроса.

Строка состояния является обязательной, в ней указывается метод проверки ответа на проверяющем сервере. В системе предусмотрено два метода: вычислить и проверить.

Поля заголовка представляют собой совокупность строк, каждая из которых имеет следующий формат:

Имя_поля: *значение*

Поля заголовка могут перечисляться в произвольном порядке, но должны использоваться не более одного раза. Названия полей заголовка не чувствительны к регистру. Имена полей приведены в табл. 1.

Название	Значение
`url`	RLCP URL проверяющего сервера
`content-length`	Размер тела запроса в байтах

Таблица 1. Имена полей в запросе клиента

Поля заголовка `content-length` и `url` являются обязательными для запроса. Формат RLCP URL:

rlcp://[*Логин***:***Пароль***@**](*IP-адрес_сервера*|*Доменное_имя_сервера*)**:***Порт*.

Пустая строка и тело запроса являются обязательными. Тело запроса оформляется в соответствии с разработанным DTD-определением и содержит тестовые наборы для проверки результата и строку ответа. Пример запроса клиента можно увидеть на рис. 4.

```
check
url:rlcp://vera:Rt5612@127.0.0.1:1732
content-length:537

<?xml version="1.0" encoding="Windows-1251"?>
<!DOCTYPE Request SYSTEM "http://de.ifmo.ru/--DTD/Request.dtd">
<Request>
    <Conditions>
            <ConditionForChecking id="1" Time="5">
                    <Input><!--5:11111011--></Input>
                <Output><!--Null:111--></Output>
            </ConditionForChecking>
    </Conditions>
    <Instructions>
<!—1 > 2 //2 ^ 3 //3 ! 3 //-->
    </Instructions>
</Request>
```

Рис. 4. Пример запроса клиента

Необходимо отметить, что входных и ожидаемых выходных наборов может быть несколько. Их формат задается автором.

Ответ сервера состоит из:

- строки состояния,
- поля заголовка,
- пустой строки,
- тела ответа.

В строке состояния указывается трехзначный код ответа, несущий информацию об успешной обработке запроса или возникших ошибках. Примеры кода ответа приведены в табл. 2.

Код	Расшифровка
100	Сервер продолжает обрабатывать запрос
200	Запрос успешно обработан, и в ответе передаются данные, указанные в запросе
300	Затребованный ресурс временно изменил свой адрес
400	Некорректный XML-файл
401	Некорректный набор входных данных
402	Несуществующая комбинация логина и пароля
403	Неподдерживаемый метод в запросе
404	Отсутствуют строка состояния или обязательные поля заголовка
405	Отсутствует значение поля заголовка
500	Затребованный ресурс временно недоступен
501	Указанный метод поддерживается протоколом, но данным сервером нет

Таблица 2. Код ответа сервера

Поля заголовка содержат информацию о размере тела ответа (в случае успешной обработки запроса) или указывают на реальное расположение поверяющего сервера для перенаправления запроса. Тело запроса оформляется в соответствии с DTD-определением и содержит тестовые наборы, направленные на проверку результата, а также результат обработки каждого тестового набора. На основании этих данных рассчитывается оценка за выполнение в соответствии с правилами оценивания сценария, которая заносится в базу данных. Пример ответа сервера можно увидеть на рис. 5.

```
200
content-length:277

<?xml version="1.0" encoding="Windows-1251"?>
<!DOCTYPE Response SYSTEM "http://de.ifmo.ru/--DTD/Response.dtd">
<Response>
        <CheckingResult id="1" Time="2" Result="1">
                <!--111-->
        </CheckingResult>
</Response>
```

Рис. 5. Пример ответа сервера

Алгоритм проверки результатов

В основе алгоритма проверки результатов лежит теория системного анализа. Схема проверки представлена на рис. 6. Здесь система 1 – эталонная система, на вход которой подаются проверяющие наборы u, а на выходе получаются всегда верные наборы y. Система 2 – программа-ответ студента. На ее вход подаются те же самые проверяющие наборы u, что и для системы 1, и выходные значения y* сравниваются с эталонными. На основании анализа отклонения результата от эталона и принятой автором системы оценивания в базе данных фиксируется полученная оценка. Такой подход позволяет использовать задания, результатом выполнения которых может быть бесконечный дискретный набор данных, например, написание программ, сборка схем.

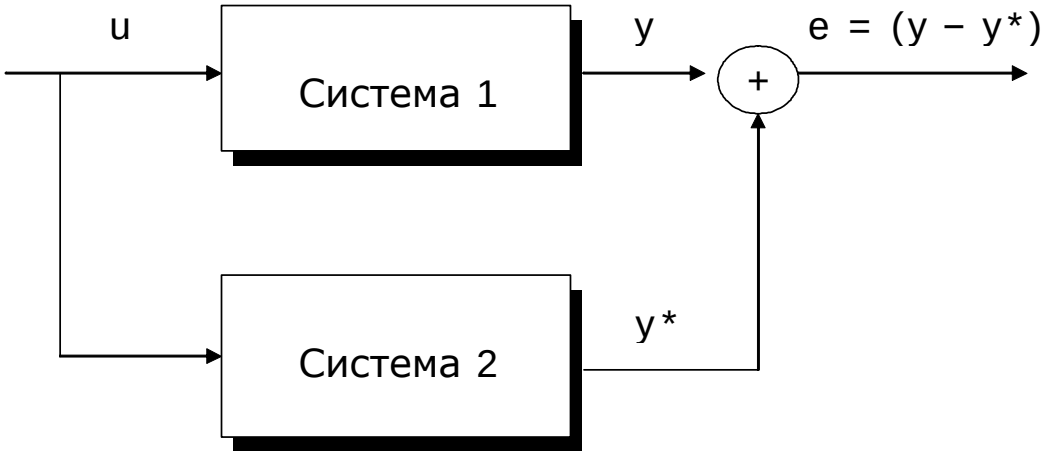


Рис. 6. Схема проверки результата

Заключение

Описываемая технология построения лабораторных работ была реализована и внедрена в систему дистанционного обучения СПбГУ ИТМО. Автоматическая проверка результата позволяет значительно снизить трудозатраты на проверку отчетов о выполнении лабораторной работы, объективно оценить правильность выполнения задания, снизить уровень затрат на приобретение дорогостоящего оборудования для лабораторных установок. Технология позволяет студентам удаленно работать с лабораторной установкой или ее математической моделью как в обучающем, так и в аттестующем режиме, оперативно получать оценку за выполнение задания. Преподаватели освобождаются от длительной проверки результатов и сосредотачивают свое внимание на объяснение теоретических аспектов.

Основные положения и результаты работы использованы и прошли апробацию в учебном процессе на дисциплинах «Информатика», «Электроника и микропроцессорная техника», «Теоретическая механика». Одновременно в учебном процессе участвовало порядка двадцати человек. Эксперимент показал, что такая нагрузка не критична для системы.

Дальнейшее развитие разработки заключается в поиске новых перспективных алгоритмов проверки результатов выполнения виртуальной лабораторной работы.

Литература

1. Лямин А.В., Чежин М.С. Модульное построение электронных учебно-методических комплексов для системы дистанционного обучения. // Труды XII Всероссийской научно-методической конференции «Телематика'2005». Санкт-Петербург, 2005. Т. 2. С. 511–512.
2. Волкова А.А. Технология построения виртуальных лабораторных работ. / Сборник материалов Всероссийского конкурса инновационных проектов аспирантов и студентов по приоритетному направлению развития науки и техники «Информационно-телекоммуникационные системы» / Под. ред. А.О. Сергеева. М.: ГНИИ ИТТ «Информика», 2005. 132 с.
3. Техническое руководство по разработке учебно-методического комплекса для системы дистанционного обучения / ЦДО СПбГУ ИТМО, 2005. 135 с.

СОДЕРЖАНИЕ

Научно-технический вестник СПбГУ ИТМО. Выпуск 25. Исследования в области информационных технологий. / Главный редактор д.т.н., профессор В.Н. Васильев – СПб: СПбГУ ИТМО, 2006. 214 с.

НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК СПбГУ ИТМО
Выпуск 25

ИССЛЕДОВАНИЯ В ОБЛАСТИ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Главный редактор
д.т.н., профессор В.Н. Васильев
Научный редактор выпуска
д.т.н., профессор В.Л. Ткалич
Дизайн обложки В.А. Петров
Редакционно-издательский отдел СПбГУ ИТМО
Зав. РИО Н.Ф. Гусарова
Лицензия ИД № 00408 от 05.11.99.
Подписано в печать 25.06.2006
Заказ 955. Тираж 100 экз.