УДК 004.03

АНАЛИЗ СИСТЕМ ТЕСТИРОВАНИЯ КОМПОНЕНТОВ АРХИТЕКТУРЫ ПРИЛОЖЕНИЙ В УСЛОВИЯХ НЕОПРЕДЕЛЁННОСТИ

Мамедгулиев Р.И. (СПбГЭТУ «ЛЭТИ»)
Научный руководитель – доктор экономических наук, профессор Цуканова О.А.
(СПбГУ)

Введение. Тестирование компонентов архитектуры приложений (модулей, сервисов, адаптеров, интеграционных контрактов и инфраструктурных конфигураций) в современных условиях сталкивается с высокой степенью неопределённости. Данная неопределённость выражается в неполноте и изменчивости требований от бизнес-заказчика, вариативности эксплуатационных нагрузок, динамике облачной инфраструктуры и обеспечении частичной наблюдаемости внутренних состояний распределённых систем. В результате, тестирование перестало являться исключительно процедурой верификации и валидации реализованной логики и трансформировалось в инструмент уточнения гипотез о поведении системы, в том числе и для оценки рисков. Особенно это актуально для микросервисных приложений, где даже корректные по отдельности компоненты могут демонстрировать непредсказуемый ход поведения при интеграции целостной системы.

Основная часть. Неопределённость на уровне архитектурных компонентов связана с несколькими факторами: изменчивостью требований к атрибутам качества (безопасностью, производительностью и отказоустойчивостью), нестабильностью профилей нагрузки, а также зависимостью поведения системы от конфигураций среды выполнения. Например, поведение сервиса при пятидесяти запросах в секунду может существенно различаться от поведения того же сервиса при реализации пятисот запросов, а иное размещение контейнеров или изменение сетевых политик способно влиять на латентность и частоту отказов. В этих условиях обычный набор модульных тестов оказывается недостаточным, и в архитектурное тестирование важны для включения инструменты, ориентированные на интеграцию, взаимодействие и реализацию поведения в динамическом режиме [1]. Поэтому анализ систем тестирования в условиях роста неопределённости должен учитывать не только полноту покрытия, но и способность работы инструментов с вариативностью, случайностью и различной инфраструктурной динамикой.

При тестировании интерфейса API и контрактов взаимодействия широко применяются инструменты, такие как Postman, Insomnia, SoapUI и Swagger. Они позволяют формировать коллекции запросов, моделировать различные сценарии обмена данными, проверять корректность ответов и автоматизировать регрессионные проверки в CI/CD [2]. В условиях неопределённости такие инструменты ценны тем, что дают возможность быстро варьировать входные параметры, заголовки, форматы данных и проверять устойчивость сервиса к нетиповым комбинациям входов. Однако их эффективность ограничена, если не учитывать реальные профили нагрузки и инфраструктурные факторы.

Для нагрузочного и стресс-тестирования архитектурных компонентов используются решения, такие как Apache JMeter, Gatling и k6. Эти инструменты позволяют моделировать вариативные сценарии пользовательского поведения, управлять интенсивностью и распределением запросов, анализировать показатели времени отклика и отказов. В условиях неопределённости особое значение приобретает не только среднее время отклика, но и анализ «хвостов» распределения, деградации при пиковых нагрузках и поведения системы при конкуренции за ресурсы. Архитектурная ценность таких тестов заключается в выявлении узких мест взаимодействия компонентов и неочевидных эффектов масштабирования.

Динамический анализ и фаззинг-тестирование применяются, когда спецификация поведения системы неполна или трудно формализуема. Инструменты вроде AFL и libFuzzer ориентированы на автоматическую генерацию входных данных с целью обнаружения аварийных состояний, некорректной обработки исключений и уязвимостей. В контексте микросервисной архитектуры фаззинг может быть направлен не только на отдельный сервис,

но и на API-шлюзы, сериализацию сообщений, механизмы аутентификации и взаимодействие через брокеры сообщений [3]. Подход помогает выявлять дефекты, возникающие в редких и трудно прогнозируемых комбинациях параметров, снижая неопределённость относительно корректности поведения компонентов в реальной эксплуатации.

Отдельное направление связано с проверкой устойчивости архитектуры к отказам и инфраструктурным сбоям. Здесь применяются инструменты хаос-инжиниринга, такие как Chaos Monkey и LitmusChaos, встроенные механизмы оркестрации в Kubernetes. Эти средства позволяют искусственно инициировать сбои (перезапуск контейнеров, ограничение ресурсов, сетевые задержки), анализировать реакцию системы. В условиях неопределённости подобные эксперименты особенно важны, поскольку сценарии отказов часто невозможно полностью предсказать на этапе проектирования. Хаос-тестирование оценивает корректность фолбек-механизмов, стратегий ретраев и изоляции компонентов, проверяет соответствие фактической устойчивости заявленным архитектурным характеристикам [4].

Таким образом, инструменты API-тестирования позволяют варьировать входные параметры и проверять контракты, а нагрузочные средства выявляют предельные режимы и узкие места, фаззинг обнаруживает дефекты в непредсказуемых комбинациях входов, а хаос-инжиниринг проверяет устойчивость к отказам среды. Их комплексное применение формирует более реалистичную картину поведения архитектуры в условиях вариативной и частично неизвестной эксплуатации.

Выводы. В условиях неопределённости тестирование архитектурных компонентов должно рассматриваться как систематический процесс снижения риска и уточнения знаний о поведении приложения. Невозможно полностью устранить неопределённость, однако её можно контролировать через сочетание различных инструментов: проверки контрактов и API, нагрузочного моделирования, динамического анализа и управляемых экспериментов с отказами. Практически это означает необходимость перехода от изолированных тестовых задач к интегрированной и непрерывной системе архитектурного тестирования, встроенной в жизненный цикл разработки и эксплуатации систем. При данном подходе тестирование становится механизмом повышения надёжности и предсказуемости сложных ИТ-систем.

Список использованных источников:

1. Кулямин В.В. Обзор методов динамического анализа программного обеспечения // Труды ИСП РАН. – 2023. – Т. 35. – № 4. – С. 7-44.
2. Чумакова А.А. Стратегия подготовки тестовых данных через Postman Flows при тестировании API // Universum: технические науки. – 2024. – № 8 (125). – С. 53-55.
3. Юрьев А.С. Фаззинг полиморфных систем в структурах микросервисов // Труды ИСП РАН. – 2024. – Т. 36. – № 1. – С. 45-60.
4. Бачурин Д.Ю. Использование хаос-инжиниринга в системе оркестрации контейнеров Kubernetes // Вестник науки. – 2025. – № 5 (86). – С. 556-561.