**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное бюджетное**

**образовательное учреждение высшего образования**

**«Заполярный государственный университет им. Н.М Федоровского»**

**Кафедра Информационных систем и технологий**

УТВЕРЖДАЮ:

Зав.кафедрой

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, фамилия и инициалы)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту (работе) по дисциплине:

«Теория систем и системный анализ»

на тему:

Тестирование финансовых систем

Автор курсового проекта (работы)

Шалахов Алексей Сергеевич

                                                                                                    (подпись, дата, фамилия и инициалы)

Направление подготовки (специальности)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_09.03.03 «Прикладная информатика в экономике»\_\_\_\_

(код, наименование)

Обозначение курсового проекта (работы)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Группа\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель проекта (работы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

                                                        (подпись, дата)                                               (фамилия, инициалы)

Проект (работа) защищен(а) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Оценка \_\_\_\_\_\_\_\_\_\_\_\_

                                                                                                    (дата)

Члены комиссии : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

                                                        (подпись, дата)                                               ( фамилия, инициалы)

                                          \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

                                                        (подпись, дата)                                               ( фамилия, инициалы)

                                          \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

                                                        (подпись, дата)                                               ( фамилия, инициалы)

Норильск, 2025

Введение

Современные финансовые системы — это сложные технические и организационные комплексы, обеспечивающие функционирование банков, платёжных сервисов, криптовалютных платформ, электронных кошельков и других инструментов управления денежными потоками. В условиях стремительной цифровизации экономики такие системы становятся неотъемлемой частью жизни миллионов людей и бизнеса по всему миру. Их стабильность, надёжность и безопасность напрямую влияют на финансовое благополучие отдельных пользователей, корпораций и даже целых государств.

Актуальность исследования определяется высокой степенью зависимости финансовых процессов от информационных технологий, а также постоянным ростом числа кибератак и мошеннических схем, направленных на хищение средств и компрометацию данных. По данным Positive Technologies, в 2023 году на финансовый сектор было совершено 58% всех зафиксированных кибератак. Примеры таких инцидентов — массовый сбой системы быстрых платежей (СБП) в 2023 году, повлёкший задержку 1,5 млн транзакций из-за ошибки в API, и утечка данных в сервисе Revolut (2022), затронувшая 50 тысяч счетов клиентов. Подобные случаи приводят не только к прямым финансовым потерям и судебным искам, но и к потере доверия клиентов, что особенно критично для финансовых организаций.

Важным фактором является также ужесточение требований со стороны регуляторов. Финансовые системы должны соответствовать международным и национальным стандартам информационной безопасности:

- PCI DSS (Payment Card Industry Data Security Standard) — требования к защите данных платёжных карт;

- GDPR (General Data Protection Regulation) — регламент обработки персональных данных для ЕС;

- ГОСТ Р 57580 — российский стандарт защиты информации в финансовых организациях.

Нарушение этих стандартов влечёт за собой серьёзные штрафы (до 4% мирового оборота компании) и угрозу приостановки деятельности.

С экономической точки зрения профилактика и раннее выявление уязвимостей жизненно важны для бизнеса. Согласно исследованию IBM, исправление критических ошибок после запуска системы обходится в 30 раз дороже, чем их устранение на этапе проектирования. К примеру, ошибка в коде списания средств у PayPal в 2022 году привела к убыткам в $20 млн, что подчёркивает важность тщательного тестирования и анализа систем до вывода на рынок.

Технологический прогресс диктует новые вызовы:

- Open Banking и интеграция с внешними сервисами увеличивают количество точек входа для атакующих;

- Криптовалютные платформы и децентрализованные финтех-сервисы сложно тестировать из-за отсутствия единого контролирующего центра и специфики архитектуры.

В этих условиях традиционные методы тестирования — ручное или автоматизированное тестирование уже готового программного продукта — часто не позволяют своевременно выявить критические дефекты или угрозы безопасности. Необходим проактивный подход, основанный на моделировании процессов, архитектуры и сценариев работы системы ещё на этапе проектирования.

Моделирование бизнес-процессов с использованием формальных методов (EDEF0, DFD, IDEF3) позволяет:

- выявить логические ошибки и уязвимости до реализации системы;

- проанализировать потоки данных и сценарии сбоев;

- подготовить комплексные тест-кейсы и повысить эффективность тестирования.

Таким образом, тестирование финансовых систем на основе моделирования — это не просто “проверка кода”, а инструмент предотвращения кризисов, защиты интересов пользователей и сохранения деловой репутации компаний. В условиях цифровой трансформации экономики внедрение таких подходов становится не только актуальным, но и жизненно необходимым.

В последующих разделах работы будут рассмотрены теоретические основы тестирования финансовых систем, специфика применения моделей EDEF0, DFD, IDEF3, а также разработана и апробирована комплексная методика выявления уязвимостей на этапе проектирования и внедрения финтех-продуктов.

Цели и задачи исследования

Цель исследования

Основная цель данной курсовой работы — разработать и обосновать комплексную методику тестирования финансовых систем с использованием методов моделирования бизнес-процессов (EDEF0, DFD, IDEF3) для выявления уязвимостей и повышения надёжности на этапе проектирования, внедрения и эксплуатации.

В современных условиях цифровой трансформации финансовой сферы особую важность приобретает не только поиск и устранение ошибок в программном обеспечении, но и системная работа по предупреждению инцидентов, связанных с архитектурными просчётами, недостатками бизнес-логики и ошибками интеграции. Использование формальных моделей и системного анализа позволяет повысить качество тестирования и минимизировать риски для бизнеса и клиентов.

Задачи исследования

Для достижения поставленной цели в работе решаются следующие задачи:

1. Анализировать современные подходы к тестированию финансовых систем:

Провести обзор существующих методов, инструментов и стандартов тестирования в банковской сфере, платёжных сервисах и криптовалютах. Сравнить традиционные и инновационные техники выявления уязвимостей.

2. Классифицировать риски и угрозы для финансовых систем:

Выделить основные типы угроз (кибератаки, внутренние ошибки, сбои интеграций), а также оценить их влияние на надёжность, информационную безопасность и бизнес-процессы.

3. Описать возможности и преимущества использования моделирования бизнес-процессов:

Привести теоретическое обоснование применения моделей EDEF0, DFD, IDEF3 для исследования архитектуры, потоков данных и сценариев функционирования финансовых систем.

4. Разработать примеры построения моделей EDEF0, DFD, IDEF3 на основе реальных или типовых кейсов:

Реализовать текстовое и графическое описание моделей для конкретных сценариев: обработка транзакций, проверка безопасности передачи данных, анализ сбоев и "узких мест".

5. Провести анализ моделей для выявления уязвимостей и проблемных зон:

Использовать построенные схемы для поиска потенциальных дефектов в бизнес-логике, архитектуре, интеграционных интерфейсах.

6. Сформулировать практические рекомендации по внедрению моделирования в процесс тестирования:

Определить этапы интеграции моделей в жизненный цикл разработки и эксплуатации, дать советы по автоматизации и визуализации результатов.

7. Оценить экономическую и организационную эффективность применения комплексной методики:

Сравнить затраты на раннее моделирование и тестирование с потенциальными потерями от инцидентов и нештатных ситуаций в реальных финансовых системах.

Выполнение указанных задач позволит реализовать системный и проактивный подход к тестированию финансовых систем, обеспечив их устойчивость, соответствие стандартам безопасности и снижение рисков для всех участников финансового рынка.

Объект и предмет исследования

Объект исследования

Объектом исследования в данной курсовой работе выступают современные финансовые системы, реализуемые в виде программных комплексов для банков, платёжных сервисов, финансовых платформ, криптовалютных бирж и интеграционных решений на стыке финансов и информационных технологий (FinTech). Это могут быть как централизованные решения — например, внутренние банковские автоматизированные системы, так и децентрализованные платформы, такие как блокчейн-сети и криптобиржи.

Финансовые системы включают в себя:

- Банковские операционные платформы (core banking, интернет- и мобильные банки)

- Платёжные шлюзы и агрегаторы (например, системы быстрых платежей, PayPal, Stripe)

- Электронные кошельки и финансовые приложения

- Криптовалютные биржи, платформы обмена цифровых активов

- Интеграционные сервисы (Open Banking API, SWIFT, системы мгновенных переводов)

Эти системы характеризуются высоким уровнем интеграции с внешними сервисами, многоуровневой архитектурой, необходимостью строгого соблюдения требований безопасности и обработки большого объёма транзакций в реальном времени.

Предмет исследования

Предметом исследования являются методы, процессы и инструменты тестирования финансовых систем, а также способы выявления уязвимостей и обеспечение устойчивости систем посредством применения формальных моделей бизнес-процессов (EDEF0, DFD, IDEF3).

В данной работе предметом изучения становятся:

- Методики моделирования бизнес-процессов для формализации и анализа архитектуры финансовых систем

- Процессы тестирования, включая декомпозицию функций, анализ потоков данных и моделирование сценариев сбоев

- Инструменты и стандарты, регламентирующие тестирование, обеспечение безопасности и соответствие нормативным требованиям

- Типовые уязвимости и инциденты, связанные с ошибками проектирования, интеграции и эксплуатации в финансовых системах

Особое внимание уделяется интеграции методов системного анализа и моделирования в жизненный цикл разработки и тестирования, а также практическому применению моделей для предотвращения рисков, связанных с кибератаками, утечками данных, сбоями и нарушениями бизнес-логики.

Таким образом, объект исследования — это реальные финансовые ИТ-системы, а предмет — процессы и инструменты их тестирования, обеспечивающие безопасность, надёжность и соответствие современным требованиям рынка и регуляторов.

Методологическая основа исследования

В работе применяется системный подход, предполагающий рассмотрение финансовых систем как сложных, многоуровневых объектов, в которых тесно переплетаются технологические, организационные и экономические компоненты. Методология исследования базируется на принципах теории систем, системного анализа, а также на современных практиках моделирования бизнес-процессов и тестирования программного обеспечения.

Ключевой элемент методики — использование формальных моделей для описания, анализа и оптимизации процессов тестирования финансовых систем. Такой подход позволяет выявлять уязвимости, неочевидные на этапе традиционного тестирования, и формализует процесс поиска ошибок на ранних стадиях жизненного цикла разработки.

Основные методы исследования:

- Анализ и синтез — оценка существующих подходов, стандартов и инструментов тестирования, выделение их преимуществ и недостатков.

- Функциональное моделирование (EDEF0) — декомпозиция процессов тестирования на функциональные блоки, описание их взаимодействия, входов и выходов.

- Моделирование потоков данных (DFD) — выявление и визуализация потоков информации между компонентами системы, определение точек возможных утечек или атак.

- Сценарное моделирование (IDEF3) — описание типовых и аварийных сценариев функционирования системы, анализ последовательности событий и их последствий.

- Кейс-анализ — разбор реальных и гипотетических инцидентов в финансовых системах для демонстрации эффективности выбранного подхода.

Применение стандартов и нормативных документов

В процессе исследования применялись требования и рекомендации ведущих стандартов в области информационной безопасности и тестирования финансовых систем:

- PCI DSS — определяет обязательные меры по защите платёжных данных;

- GDPR — регламентирует обработку персональных данных;

- ГОСТ Р 57580 — устанавливает требования по защите информации для финансовых организаций в РФ;

- ISO/IEC 27001 — международный стандарт управления информационной безопасностью;

- Методические рекомендации Банка России — регламентируют обеспечение безопасности платёжных систем.

Этапы исследования

1. Сбор и анализ требований к безопасности и надёжности финансовых систем;

2. Построение формальных моделей процессов (EDEF0, DFD, IDEF3) для выбранных кейсов;

3. Анализ моделей на предмет уязвимостей, логических ошибок, неэффективных решений;

4. Разработка и апробация тест-кейсов на основе моделей;

5. Оценка эффективности предложенной методики через сопоставление с традиционными подходами и анализ экономических последствий выявленных инцидентов.

Обоснование выбранной методики

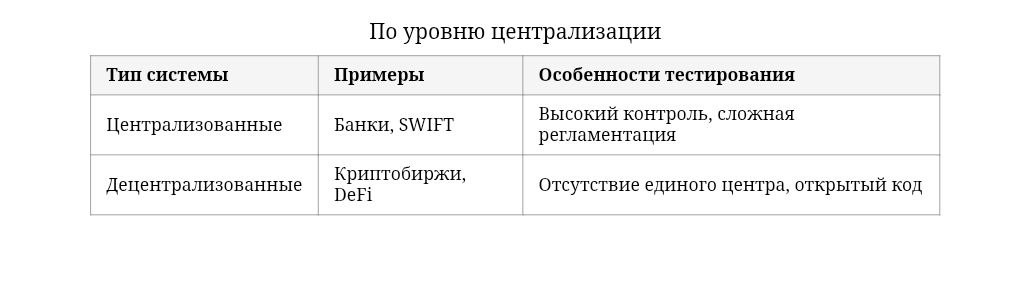
Использование формальных моделей позволяет перейти от интуитивного тестирования к системному анализу процессов, сделать тестирование прозрачным, воспроизводимым и легко масштабируемым. Это особенно важно для финансовых систем, где цена ошибки крайне высока, а требования к качеству и безопасности — максимальны.

В результате применение указанных методов и инструментов позволяет обеспечить комплексный, структурированный и проактивный подход к тестированию финансовых систем, интегрируя лучшие практики системного анализа и современные требования рынка.

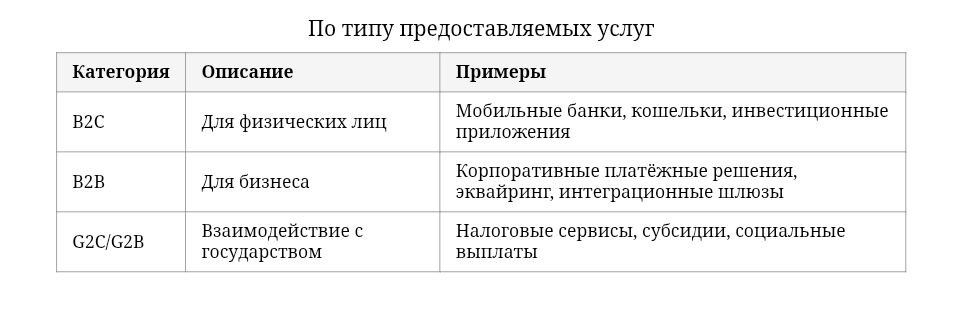
Теоретические основы тестирования финансовых систем

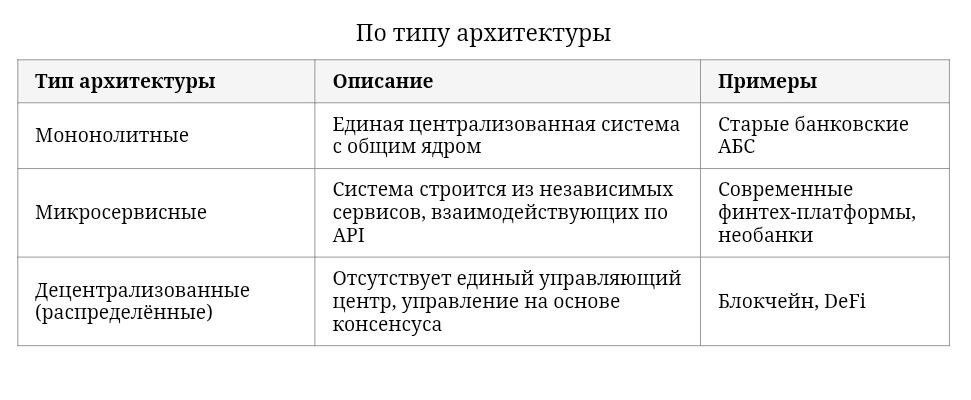
Классификация финансовых систем

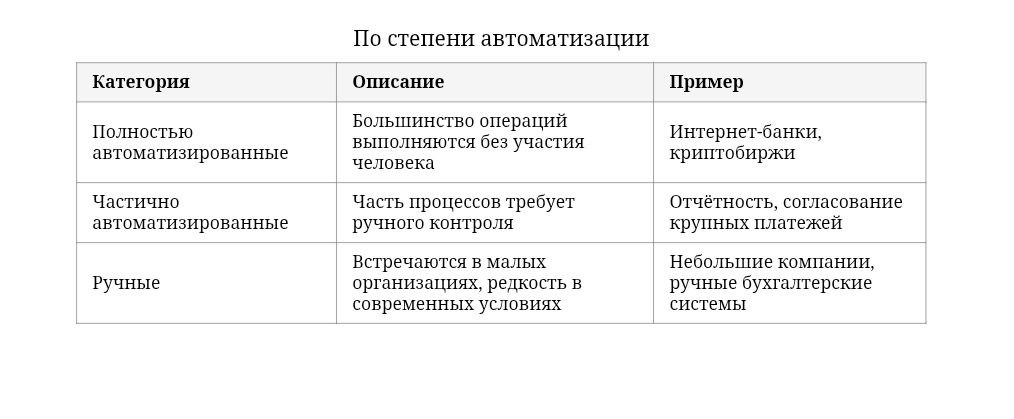
Финансовая система — это совокупность институтов, технологий, процессов и механизмов, обеспечивающих движение, учёт и контроль финансовых потоков в экономике. В современных условиях под финансовой системой всё чаще понимают сложные информационно-технологические комплексы, поддерживающие электронные операции, автоматизацию расчётов и интеграцию с внешними сервисами. Для целей системного анализа и тестирования такие системы классифицируются по ряду признаков: функциональному назначению, архитектуре, уровню централизации, типу услуг, интеграционной открытости и степени автоматизации.

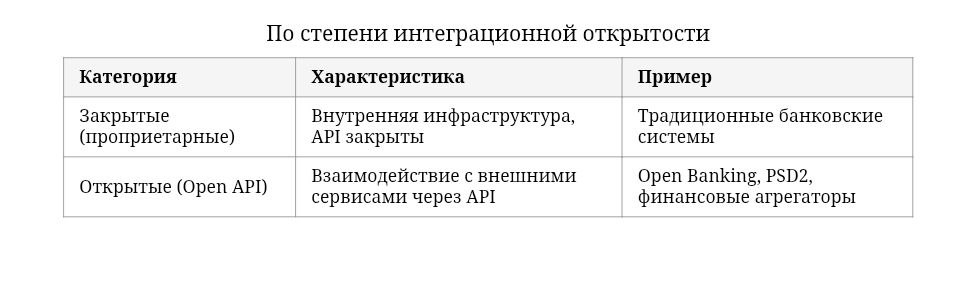




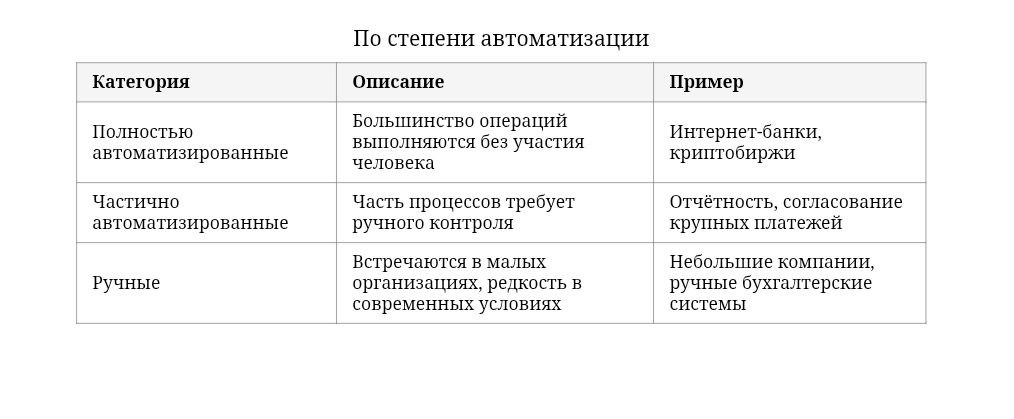












Вывод:

Классификация финансовых систем необходима для выбора правильных методик тестирования, оценки рисков и построения эффективных моделей бизнес-процессов. Каждый класс и тип системы предъявляет собственные требования к архитектуре, безопасности, интеграции и подходам к тестированию. Это требует внедрения гибких и адаптируемых методик, учитывающих специфику как централизованных банковских систем, так и децентрализованных финтех-платформ.

Особенности тестирования в FinTech

Тестирование финансовых систем (FinTech) имеет ряд принципиальных отличий от тестирования обычных информационных систем, что обусловлено высокой стоимостью ошибок, требованиями к безопасности, регуляторными нормами и спецификой бизнес-процессов.

1. Усиленные требования к безопасности и конфиденциальности

В финансовых приложениях обрабатываются чувствительные персональные данные, платёжные реквизиты, сведения о счетах и транзакциях. Ошибки в системе могут привести к утечке средств или информации, что требует:

- Комплексного тестирования на соответствие стандартам безопасности (PCI DSS, ГОСТ Р 57580, ISO 27001).

- Проведения тестов на проникновение (penetration testing), сканирования на уязвимости, анализа защищённости API и мобильных приложений.

- Проверки механизмов шифрования, управления сессиями, аутентификации и авторизации.

2. Строгое соответствие регуляторным и отраслевым стандартам

Любая ошибка, связанная с некорректной обработкой платёжных данных или несоблюдением регуляторных требований (GDPR, Закона о персональных данных, рекомендаций Центрального Банка), ведёт к штрафам и риску блокировки деятельности. Поэтому тестирование должно включать:

- Проверку корректности обработки персональных данных.

- Тестирование процедур логирования и аудита операций.

- Проверку соответствия бизнес-процессов стандартам и политике информационной безопасности.

3. Высокая надёжность и отказоустойчивость

Финансовые системы должны работать 24/7, часто с миллионами транзакций в день. Любой сбой может привести к финансовым потерям и массовому недовольству клиентов. Поэтому необходимы:

- Тестирование на отказоустойчивость (failover, disaster recovery).

- Проведение нагрузочного (stress, load) тестирования для оценки пределов производительности.

- Проверка работы систем резервного копирования и восстановления.

4. Тестирование интеграций и внешних интерфейсов

Финтех-системы интегрируются с большим количеством внешних сервисов: платёжные шлюзы, банки, государственные порталы, кредитные бюро, блокчейн-сети. Особенности:

- Тестирование API на предмет устойчивости к ошибкам, обработку некорректных ответов.

- Эмуляция внешних сервисов (mocking, stubbing).

- Проверка безопасности интеграционных каналов (шифрование, контроль доступа).

5. Работа с параллелизмом и транзакционностью

Многие процессы в финтехе требуют строгой атомарности и согласованности данных (например, перевод средств, списание и зачисление). Для этого необходимо:

- Тестирование корректности транзакций (ACID).

- Проверка сценариев “двойного списания”, блокировок, гонок данных.

- Использование специализированных инструментов для анализа параллельного выполнения.

6. Особые подходы к автоматизации

Автоматизация тестирования в FinTech строится с учётом сложных бизнес-правил, динамичных интерфейсов и частых изменений нормативных требований. Важные аспекты:

- Использование BDD (Behavior Driven Development) для формализации бизнес-сценариев.

- Интеграция автоматических тестов в CI/CD пайплайны.

- Внедрение регрессионного тестирования при любом изменении бизнес-логики.

7. Верификация расчётов и финансовой логики

Важнейшая особенность — необходимость автоматизированной проверки корректности расчётов, комиссий, процентных ставок и других финансовых формул:

- Разработка автотестов для проверки арифметики, округления, расчёта процентов по кредитам/депозитам.

- Сравнение результатов с эталонными значениями (reference data).

8. Примерная таблица:



Вывод:

Тестирование в FinTech — это комплексная, строго регламентированная деятельность с акцентом на безопасность, качество, надёжность и соответствие нормативам. Это требует применения специализированных методик, автоматизации и валидации бизнес-логики на всех этапах жизненного цикла системы.

Standards и регуляторные требования

Финансовые системы — одна из наиболее строго регулируемых областей информационных технологий. Их разработка и эксплуатация сопровождаются жёстким контролем со стороны национальных и международных регуляторов, а также отраслевых ассоциаций. Несоблюдение стандартов безопасности и нормативных актов может привести к крупным штрафам, отзыву лицензии и репутационным потерям.

1. Международные стандарты информационной безопасности

* PCI DSS (Payment Card Industry Data Security Standard)

Является обязательным для всех организаций, принимающих, обрабатывающих и передающих данные платёжных карт. Включает требования к шифрованию, сегментации сетей, мониторингу событий, тестированию на проникновение и регулярным аудитам.

\_Пример:\_ отсутствие шифрования или неправильная настройка брандмауэров — прямая угроза нарушения PCI DSS и последующего штрафа.

* ISO/IEC 27001

Международный стандарт по управлению информационной безопасностью. Регламентирует создание, внедрение, поддержание и совершенствование системы управления информационной безопасностью, а также проведение регулярных оценок рисков.

* SWIFT Customer Security Programme (CSP)

Обязателен для участников платёжной системы SWIFT. Содержит требования к защите инфраструктуры, контролю доступа, мониторингу и реагированию на инциденты.

2. Европейские и национальные регуляторные требования

* GDPR (General Data Protection Regulation)

Европейский регламент, регулирующий обработку и хранение персональных данных. Включает требования к прозрачности, ограничению доступа, хранению только необходимой информации (“data minimization”), возможности удаления (“право на забвение”) и обязательному уведомлению о нарушениях в течение 72 часов.

* PSD2 (Revised Payment Services Directive)

Директива о платёжных услугах, регулирующая открытие банковских API (Open Banking) и усиливающая требования к аутентификации клиентов.

- Российские стандарты (ГОСТ, ФЗ-152, 187-ФЗ, 161-ФЗ)

- ГОСТ Р 57580 — комплекс требований к защите информации в финансовых организациях, включая угрозы, архитектуру, процессы и документацию.

- Федеральный закон №152-ФЗ — регламентирует обработку и хранение персональных данных.

- Федеральный закон №161-ФЗ “О национальной платёжной системе” — определяет требования к безопасности платёжных сервисов, процессинговых центров и операторов платёжных систем.

- Федеральный закон №187-ФЗ “О безопасности критической информационной инфраструктуры Российской Федерации” — вводит дополнительные меры по защите критичных финансовых сервисов.

3. Отраслевые рекомендации и лучшие практики

* Методические указания Банка России

Регулярно обновляются, содержат рекомендации по анализу угроз, тестированию защищённости, управлению инцидентами и организации процессов безопасности.

* NIST SP 800-53

Руководство по обеспечению безопасности информационных систем, широко используется при построении и тестировании архитектуры финансовых приложений.

4. Практические требования к тестированию

- Внедрение автоматических средств контроля соответствия стандартам (compliance tools).

- Проведение регулярных аудитов, ревью кода, тестирования на проникновение (pen-testing).

- Подготовка и актуализация документации по безопасности процессов (политики, инструкции, планы реагирования на инциденты).

- Внедрение механизмов журналирования (логирования) и мониторинга событий.

- Реализация процедур управления рисками и инцидентами.

5. Ответственность за нарушения

- Нарушение требований GDPR — штрафы до 20 млн евро или 4% годового оборота, что выше.

- Неисполнение PCI DSS — штрафы от платёжных систем, возможное прекращение обслуживания.

- Нарушение российских законов — административная и уголовная ответственность, отключение от национальных платёжных сервисов, отзыв лицензии.

Вывод:

Тестирование финансовых систем неразрывно связано с необходимостью строгого соблюдения стандартов и нормативов. На этапе проектирования, внедрения и эксплуатации необходимо учитывать требования регуляторов, интегрировать автоматические проверки compliance, а также проводить регулярные аудиты и тесты на проникновение для предотвращения инцидентов и обеспечения высокой степени доверия со стороны клиентов, партнёров и государства.

Риски и угрозы в финансовых системах

Финансовые системы ежедневно подвергаются воздействию множества рисков и угроз, которые могут привести к финансовым потерям, нарушению работы, утечке данных и подрыву доверия клиентов. В современных условиях, когда цифровизация охватывает все сферы финансов, вопросы управления рисками и своевременного обнаружения угроз становятся ключевыми для успешной деятельности любой организации.

1. Классификация рисков и угроз

Риски в финансовых системах можно разделить на несколько основных категорий:

- Технические риски: связанные с ошибками в программном обеспечении, сбоями оборудования, уязвимостями в архитектуре и инфраструктуре.

- Информационные угрозы: кибератаки, вирусы, фишинг, несанкционированный доступ, перехват данных, атаки типа DdoS.

- Операционные риски: ошибки персонала, неправильные настройки, нарушение бизнес-процессов, некорректные транзакции.

- Мошенничество и злоупотребления: внутренние (действия сотрудников) и внешние (мошеннические транзакции, подделка документов, атаки на пользователей).

- Риски несоответствия нормативным требованиям: штрафы, санкции, блокировки со стороны регуляторов при нарушении стандартов (PCI DSS, ГОСТ, GDPR и др.).

- Риски репутации: утрата доверия клиентов и партнёров вследствие инцидентов, связанных с безопасностью или финансовыми потерями.

2. Основные типы угроз для финансовых систем

* Кибератаки и взломы:

Пример — использование уязвимостей в API или веб-интерфейсах для получения несанкционированного доступа к счетам пользователей. Масштабные DdoS-атаки могут привести к недоступности сервисов (например, атака на банки в РФ в 2022 году).

* Массовые фишинговые кампании:

Пользователям рассылаются поддельные письма или SMS, ведущие на ложные сайты, где похищаются данные карт и пароли.

* Внутренние угрозы (insider threats):

Несанкционированные действия сотрудников, сознательное или случайное раскрытие информации, манипуляции с транзакциями.

* Утечки и потеря данных:

Нарушение целостности или конфиденциальности клиентских данных из-за взлома, ошибок в настройках хранения или передачи информации.

* Ошибки в бизнес-логике:

Недостаточно протестированные сценарии могут привести, например, к двойному списанию средств или ошибкам расчёта процентов.

* Мошеннические транзакции:

Использование краденых карт, подделка документов, атаки через уязвимые интеграции с внешними сервисами.

* Сбои интеграционных процессов:

Нарушение связи между банком и процессингом, ошибочная маршрутизация платежей, сбои в обмене данными с государственными или международными системами (SWIFT, ЦБ, налоговые базы).

3. Специфические угрозы для FinTech и цифровых платформ

* Атаки на смарт-контракты и DeFi-протоколы:

Программы на блокчейне могут содержать ошибки, позволяющие злоумышленникам выводить средства или нарушать логику работы платформы (пример — взлом Poly Network в 2021 году на сумму более $600 млн).

* Эксплуатация уязвимостей мобильных приложений:

Использование небезопасных хранилищ, слабая аутентификация, подделка запросов.

* Манипуляции с курсом и ликвидностью на криптобиржах:

Атаки типа “pump and dump”, искусственная нагрузка на биржу для обрушения торгов.

4. Примеры последствий реализации угроз

- Прямая потеря денежных средств (у клиентов или самой организации).

- Нарушение работы сервисов (недоступность, задержки транзакций).

- Массовая утечка персональных данных с последующими мошенническими действиями.

- Юридические санкции, крупные штрафы, отзыв лицензий.

- Репутационный ущерб, отток клиентов, снижение рыночной стоимости компании.

5. Механизмы управления рисками и противодействия угрозам

- Построение многоуровневой системы защиты (периметр, сегментация, аутентификация, шифрование).

- Регулярное проведение тестирования на проникновение и аудита защищённости.

- Мониторинг и анализ событий безопасности (SIEM-системы).

- Обучение персонала и информирование клиентов о методах социальной инженерии.

- Внедрение процедур резервного копирования и планов восстановления после сбоев (disaster recovery).

- Интеграция риск-менеджмента в процессы проектирования, тестирования и эксплуатации.

Вывод:

В современных финансовых системах риск-менеджмент и обеспечение безопасности становятся неотъемлемой частью жизненного цикла продукта. Эффективное тестирование — один из ключевых инструментов для своевременного выявления уязвимостей, минимизации вероятности реализации угроз и снижения потенциального ущерба для бизнеса, клиентов и всей финансовой экосистемы.

Моделирование бизнес-процессов как этап тестирования: зачем моделировать процессы

Моделирование бизнес-процессов — это формальное описание того, как функционирует финансовая система на всех уровнях: от приёма и обработки платёжных поручений до взаимодействия с внешними сервисами и пользователями. В рамках тестирования финансовых систем моделирование становится важнейшим инструментом для повышения качества, надёжности и безопасности конечного продукта.

1. Выявление скрытых дефектов и уязвимостей на ранних этапах

Моделирование позволяет детально проанализировать логику работы системы ещё до начала программирования или автоматизированного тестирования. На этапе построения моделей выявляются:

- Логические противоречия, дублирующие или конфликтующие процессы;

- Неочевидные ветвления в сценариях, которые могут привести к ошибкам;

- “Узкие места” и потенциальные точки сбоя, недоступные при поверхностном тестировании;

- Возможные сценарии обхода контрольных механизмов, что особенно критично для финансовых систем (например, возможность двойного списания средств).

2. Формализация и стандартизация требований

Модель бизнес-процесса служит универсальным языком общения между разработчиками, тестировщиками, бизнес-аналитиками и заказчиком:

- Позволяет точно зафиксировать, что именно должна делать система (какие шаги, проверки, реакции на сбои);

- Помогает избежать разночтений и “человеческого фактора” при передаче требований;

- Упрощает составление спецификаций для автоматизированного тестирования.

3. Повышение эффективности тестирования и автоматизации

На основе моделей можно:

- Генерировать комплексные тест-кейсы, учитывающие все варианты поведения системы;

- Автоматизировать часть тестирования, используя сценарии и логику, прописанные в моделях;

- Сократить затраты времени и ресурсов на написание и обновление тестовой документации.

4. Анализ влияния изменений и управление рисками

Модели позволяют быстро оценить последствия любых изменений в бизнес-процессах:

- Видно, какие компоненты и функции затрагиваются в случае доработки или внедрения новых сервисов;

- Можно заранее спрогнозировать, где потенциально возникнут сбои, и подготовить соответствующие тесты;

- Упрощается управление архитектурными и операционными рисками.

5. Соответствие стандартам и подготовка к аудиту

Во многих стандартах (PCI DSS, ISO 27001, ГОСТ Р 57580) прямо или косвенно требуется документировать и регулярно пересматривать бизнес-процессы:

- Моделирование облегчает подготовку к внешним и внутренним аудитам;

- Позволяет наглядно продемонстрировать регуляторам и аудиторам, что процессы прозрачны и контролируемы.

6. Улучшение коммуникаций и обучение персонала

Графические и формальные модели бизнес-процессов легко воспринимаются новыми сотрудниками и внешними участниками проектов:

- Ускоряют адаптацию новых специалистов;

- Снижают вероятность ошибок, связанных с неправильным пониманием логики работы системы.

Вывод:

Моделирование бизнес-процессов — это не просто “рисование схем”, а мощный инструмент системного анализа, управления качеством и безопасностью в тестировании финансовых систем. Оно позволяет выявлять и устранять ошибки на самых ранних стадиях, формализовать требования, повысить прозрачность разработки и обеспечить соответствие стандартам индустрии.

Сравнение методов исследования в тестировании финансовых систем

В тестировании финансовых систем применяются различные методы исследования, каждый из которых имеет свои сильные и слабые стороны, а также области наилучшего применения. Сравнение этих методов позволяет выбрать наиболее эффективный подход для конкретных задач, учитывая специфику финансового сектора, требования к безопасности, надёжности и соответствию регуляторным нормам.

1. Функциональное моделирование (IDEF0)

Позволяет формализовать и визуализировать основные функции системы, их взаимосвязи, входы и выходы. Отлично подходит для декомпозиции сложных процессов на отдельные блоки и анализа архитектуры на верхнем уровне.

- Преимущества:

- Чёткая структура, простота восприятия;

- Позволяет выявить избыточные или дублирующие функции;

- Удобен для согласования требований между бизнесом и ИТ.

- Недостатки:

- Не отображает последовательность событий во времени;

- Ограничен в описании динамики и сценариев сбоев.

2. Моделирование потоков данных (DFD)

Ориентировано на отображение движения и преобразования информации внутри системы. Особенно полезно для поиска “узких мест” и потенциальных точек утечки данных.

- Преимущества:

- Детально показывает, как и куда передаются данные;

- Упрощает анализ безопасности и интеграций;

- Помогает выявлять “бутылочные горлышки” в обмене информацией.

- Недостатки:

- Меньше внимания уделяет бизнес-логике и ролям пользователей;

- Может быть громоздким для больших систем.

3. Сценарное моделирование (IDEF3)

Используется для описания последовательности событий и типовых сценариев функционирования системы, включая штатные и аварийные ситуации.

- Преимущества:

- Позволяет анализировать ветвления, параллельные процессы, исключения;

- Удобен для тестирования сложных бизнес-кейсов и автоматизации сценариев.

- Недостатки:

- Требует подробной проработки сценариев;

- Не всегда легко интегрируется с архитектурными моделями.

4. Кейс-анализ и эмпирические методы

Основаны на разборе реальных или гипотетических инцидентов, использовании статистики, обратной связи от пользователей.

- Преимущества:

- Позволяют учитывать реальные угрозы и ошибки;

- Помогают выявлять уязвимости, неочевидные на этапе моделирования.

- Недостатки:

- Неполнота охвата, фокус на уже известных проблемах;

- Зависят от качества и объёма исходных данных.

5. Автоматизированное тестирование и CI/CD

Регулярное, автоматическое выполнение тестов на всех этапах жизненного цикла системы.

- Преимущества:

- Высокая скорость и повторяемость;

- Быстрое обнаружение ошибок при изменениях в коде;

- Легко интегрируется с DevOps-процессами.

- Недостатки:

- Требует значительных затрат на настройку;

- Не всегда покрывает сложные бизнес-сценарии и “человеческий фактор”.

Сравнительная таблица методов исследования



Вывод:

Оптимальный подход к исследованию и тестированию финансовых систем — это комбинирование различных методов: формальных моделей для анализа архитектуры и процессов, сценарного моделирования для проверки бизнес-логики, кейс-анализа для учёта реальных угроз и автоматизации для постоянного контроля качества. Такой комплексный подход позволяет максимально снизить риски ошибок, повысить надёжность и соответствие нормативным требованиям.

Методология EDEF0: Теория

EDEF0 (часто встречается также как IDEF0 — Integration Definition for Function Modeling) — это методология функционального моделирования, предназначенная для формализации и графического описания бизнес-процессов, систем и организаций. Она является частью семейства стандартов IDEF, разработанных в 1970–1980-х годах по заказу Министерства обороны США для нужд системного анализа, управления проектами и автоматизации.

Ключевая идея EDEF0 — представить любую сложную систему в виде иерархии функций (действий, процессов), которые преобразуют входы в выходы под воздействием управляющих воздействий и с использованием определённых ресурсов (механизмов). Такой подход позволяет декомпозировать крупные процессы на более простые, описывать взаимосвязи между ними, а также выявлять дублирующие, лишние или неэффективные шаги в работе системы.

Основные элементы модели EDEF0:

- Функция (блок, процесс): Представляется прямоугольником и обозначает отдельное действие или группу действий в системе.

- Стрелки: Показывают тип и направление потоков между функциями. Всего выделяют четыре типа стрелок:

- Вход (Input): То, что преобразуется функцией (данные, сырьё, требования).

- Выход (Output): Результат выполнения функции (отчёты, готовый продукт, услуги).

- Управление (Control): Ограничения, правила, инструкции — то, что определяет, как должна выполняться функция (стандарты, регламенты, законы).

- Механизм (Mechanism): Ресурсы, инструменты, исполнители, необходимые для выполнения функции (оборудование, персонал, программные средства).

Структура моделей:

- Вся система описывается на верхнем уровне одной общей функцией (контекстная диаграмма).

- Далее эта функция декомпозируется на несколько подфункций, каждая из которых, при необходимости, может быть детализирована далее (иерархическое моделирование).

- Такой подход обеспечивает структурированность, позволяет анализировать процессы “сверху вниз” и выявлять проблемные места на любом уровне детализации.

Графическое отображение:

- На диаграмме каждый блок (функция) соединяется стрелками с другими блоками, указывая, что именно и каким образом передаётся между процессами.

- Для сложных систем могут строиться несколько связанных диаграмм, которые вместе образуют полную иерархию бизнес-процессов.

Преимущества EDEF0:

- Высокая наглядность и формализация процессов;

- Универсальность — подходит для любых предметных областей (производство, финансы, ИТ, логистика);

- Позволяет выявлять дублирующие и неэффективные процессы;

- Упрощает коммуникацию между бизнесом, ИТ, аналитиками и регуляторами;

- Легко интегрируется с другими методами моделирования и анализа.

Применение в тестировании финансовых систем:

- EDEF0 позволяет чётко структурировать и “разложить по полочкам” все этапы обработки финансовых операций, взаимодействие с внешними сервисами, процессы контроля и аудита.

- На основе моделей строятся тест-кейсы, выявляются неочевидные сценарии ошибок и уязвимостей.

- Формализованные модели EDEF0 часто используются для обоснования соответствия стандартам и подготовки к аудиту.

Вывод:

Методология EDEF0 — это мощный инструмент для описания, анализа и оптимизации функциональных процессов в финансовых системах. Она облегчает коммуникацию между участниками проекта, повышает прозрачность архитектуры и закладывает прочную основу для эффективного тестирования и управления рисками.

Пример построения EDEF0 для процесса тестирования

Построение модели EDEF0 позволяет структурировать процесс тестирования финансовой системы, выделить ключевые функции, их входы и выходы, а также управляющие воздействия и ресурсы. Ниже приведён пример построения EDEF0-диаграммы для процесса тестирования программного обеспечения в финансовой организации.

1. Контекстная диаграмма (A0): “Тестирование финансовой системы”

- Функция: Тестирование финансовой системы

- Входы:

— Программный продукт (ПО)

* Требования к ПО

— Тестовые данные

* Выходы:
* Отчёт о тестировании
* Обнаруженные дефекты

— Рекомендации по доработке

* Управление:
* Стандарты тестирования (например, ISO/IEC 29119, внутренние политики)
* Регуляторные требования (ГОСТ, PCI DSS)

— План тестирования

* Механизмы:
* Тестировщики
* Автоматизированные средства тестирования
* Инструменты для ведения баг-трекинга (JIRA, Bugzilla и др.)

2. Декомпозиция (A1, A2, A3…)

Рассмотрим детализацию процесса на уровне основных подфункций:

- A1. Подготовка к тестированию

- Входы: требования к ПО, план тестирования

- Выходы: тест-кейсы, сценарии, тестовые среды

- Управление: стандарты, методики

- Механизмы: аналитики, тестировщики, инструменты для проектирования тестов

- A2. Проведение тестирования

- Входы: программный продукт, тест-кейсы, тестовые данные

- Выходы: результаты тестов, обнаруженные дефекты

- Управление: сценарии тестирования, сроки

- Механизмы: тестировщики, автоматизированные средства

- A3. Анализ результатов и оформление отчётов

- Входы: результаты тестирования, обнаруженные дефекты

- Выходы: отчёт о тестировании, рекомендации

- Управление: требования к отчётности, критерии приёмки

- Механизмы: тест-менеджер, системы документооборота

3. Пример описания подфункции A2 “Проведение тестирования” в формате EDEF0

* Входы:
* Тестируемое ПО
* Тестовые данные

— Тест-кейсы

* Выходы:
* Лог тестирования

— Заявки на исправление дефектов

* Управления:
* Сценарии тестирования

— Временные ограничения

* Механизмы:
* Тестировщик
* Средства автоматизации тестирования

4. Визуализация (словесное описание)

- Программный продукт и тестовые данные поступают на вход функции “Проведение тестирования”.

- Управление осуществляется согласно заранее утверждённым сценариям тестирования и срокам.

- Механизмами служат тестировщики и средства автоматизации.

- На выходе формируются логи, протоколы и заявки на исправление дефектов, которые используются на следующем этапе анализа.

5. Практическое применение

Такая модель наглядно демонстрирует структуру процесса тестирования, позволяет определить ключевые точки контроля, декомпозировать процесс на более мелкие задачи и формализовать требования к каждому этапу. Это облегчает автоматизацию тестирования, распределение ответственности и подготовку к аудиту.

Вывод:

Использование EDEF0 для моделирования процесса тестирования финансовых систем помогает структурировать деятельность, повысить прозрачность и управляемость, а также обеспечивает соответствие стандартам и внутренним требованиям организации.

Анализ модели (Analysis model) в тестировании финансовых систем

Анализ модели — это этап в жизненном цикле тестирования, на котором проводится детальное изучение построенных бизнес-моделей (например, EDEF0, DFD, IDEF3) с целью выявления потенциальных проблем, уязвимостей, неэффективностей и рисков, а также определения приоритетных зон для тестирования.

1. Цели анализа модели

- Проверить полноту и корректность построенных моделей по отношению к бизнес-требованиям.

- Обнаружить логические ошибки, дублирование функций, “узкие места” и возможные точки отказа.

- Оценить соответствие процессов нормативным требованиям и стандартам безопасности.

- Определить критически важные сценарии, которые требуют первоочередного внимания при тестировании.

- Сформировать основу для составления комплексных и релевантных тест-кейсов.

2. Основные задачи анализа модели

- Проверка потоков данных: анализируется корректность передачи информации между функциями, поиск “бутылочных горлышек”, избыточных или неоптимальных маршрутов данных.

- Анализ ролей и ответственности: выявляются слабые места, где возможны злоупотребления или ошибки из-за недостаточного контроля со стороны персонала.

- Идентификация контрольных точек: определяются места, где необходимо внедрить дополнительные проверки, аудит или механизмы логирования.

- Оценка соответствия стандартам: модели сверяются с требованиями стандартов (например, PCI DSS, ГОСТ Р 57580, ISO/IEC 27001), чтобы убедиться в наличии необходимых процедур безопасности, резервирования, восстановления и т.д.

- Выявление сценариев отказа: анализируются варианты развития событий при сбоях, ошибках или атаках для формирования сценариев тестирования на устойчивость и безопасность.

3. Методы анализа моделей

- Визуальный анализ диаграмм и схем (ручная и коллективная экспертиза).

- Использование чек-листов и стандартных вопросов для оценки соответствия.

- Моделирование “что-если” (what-if scenarios) — проверка поведения системы при изменении входных данных, параметров или условий работы.

- Применение автоматизированных инструментов анализа (если такие предусмотрены для используемых нотаций).

4. Результаты анализа модели

- Список выявленных проблем, неэффективных участков или потенциальных уязвимостей.

- Перечень критических бизнес-сценариев и функций, требующих усиленного тестирования.

- Рекомендации по оптимизации процессов, устранению дублирования, введению новых контрольных точек.

- Документированная основа для написания тест-кейсов и сценариев тестирования.

5. Преимущества анализа модели

- Позволяет обнаружить ошибки и риски до начала или на ранних этапах тестирования, снижая затраты на исправление.

- Делает процессы тестирования и разработки более прозрачными и управляемыми.

- Обеспечивает соответствие нормативным требованиям и стандартам отрасли.

- Улучшает коммуникацию между бизнесом, ИТ и тестировщиками.

Вывод:

Анализ модели — это обязательный и стратегически важный этап тестирования финансовых систем. Он обеспечивает системный подход к выявлению и предотвращению ошибок, способствует построению эффективных и безопасных процессов, а также служит базой для дальнейшей автоматизации и оптимизации тестирования.

Диаграммы потоков данных (DFD): Теория

Диаграммы потоков данных (DFD, Data Flow Diagram) — это один из классических методов структурного анализа систем, широко используемый для моделирования и визуализации потоков информации внутри бизнес-процессов и информационных систем. DFD позволяют наглядно показать, как данные перемещаются между различными частями системы, какие процессы их обрабатывают и где происходят точки взаимодействия с внешними участниками.

Основные элементы DFD:

1. Процессы (Process):

Обозначают преобразование входных данных во внутренние или выходные. На диаграмме изображаются в виде круга или прямоугольника с закруглёнными углами. Каждый процесс должен иметь хотя бы один вход и один выход.

2. Потоки данных (Data Flow):

Отображают передачу информации между процессами, хранилищами данных и внешними сущностями. На диаграмме изображаются стрелками с указанием наименования потоков.

3. Хранилища данных (Data Store):

Представляют места хранения информации (базы данных, файлы и т.д.), которые используются или пополняются в ходе обработки. Изображаются в виде открытого прямоугольника (две параллельные линии).

4. Внешние сущности (External Entity):

Это внешние по отношению к системе участники процесса — клиенты, другие системы, организации, пользователи. Изображаются в виде прямоугольников.

Уровни DFD:

- Контекстная диаграмма (DFD Level 0):

Отображает всю систему как один процесс и показывает основные внешние взаимодействия (входы и выходы).

- Диаграммы детализированных уровней (Level 1, 2 и далее):

Декомпозируют центральный процесс на подпроцессы, показывают внутренние потоки и хранилища данных.

Правила построения DFD:

- Каждый процесс должен преобразовывать входные данные во что-то новое на выходе.

- Данные не могут передаваться напрямую между внешними сущностями и хранилищами без участия процесса.

- Потоки данных должны быть однозначно подписаны.

- На одном уровне детализации не должно быть избыточных или неиспользуемых элементов.

Преимущества DFD:

- Высокая наглядность: легко понять структуру потоков информации даже не специалистам.

- Универсальность: подходит для анализа бизнес-процессов, ИТ-систем, финансовых сервисов и многих других областей.

- Формализованность: упрощает постановку задач для разработчиков и тестировщиков, облегчает поиск ошибок в логике обработки данных.

Применение DFD в тестировании финансовых систем:

- Позволяет выявить “узкие места”, точки возможной потери или искажения данных.

- Упрощает формализацию тест-кейсов, особенно для интеграционных и сквозных сценариев.

- Помогает анализировать соответствие системы требованиям безопасности и регуляторным стандартам за счёт наглядного отображения всех потоков чувствительной информации.

Вывод:

DFD — это эффективный инструмент для структурного анализа и проектирования информационных систем, который помогает выявлять и устранять ошибки в обработке и передаче данных на ранних этапах разработки и тестирования. Диаграммы потоков данных способствуют прозрачности, взаимопониманию между участниками проекта и обеспечивают основу для построения качественных и надёжных финансовых решений.

Пример DFD (Диаграммы потоков данных) для передачи данных в финансовую систему

Рассмотрим типовую ситуацию: клиент отправляет платёжное поручение через веб-приложение банка, данные обрабатываются системой, проходят проверку, записываются в базу и передаются во внешний процессинговый центр.

Контекстная диаграмма (DFD Level 0):

На этом уровне система изображается как единый процесс “Обработка платёжного поручения”, отображаются основные внешние участники и основные потоки данных.

- Внешние сущности:

— Клиент

— Процессинговый центр

- Процесс:

— Обработка платёжного поручения

- Потоки данных:

— Поручение на платёж (от клиента к системе)

— Статус операции (от системы к клиенту)

— Информация о платеже (от системы к процессинговому центру)

— Подтверждение выполнения (от процессингового центра к системе)

Диаграмма первого уровня (DFD Level 1):

Декомпозиция процесса “Обработка платёжного поручения” на подпроцессы:

1. Приём данных от клиента

- Поток: Платёжное поручение

- Хранилище: Временное хранилище заявок

2. Валидация поручения

- Поток: Проверенные данные

- Хранилище: Хранилище ошибок (если поручение некорректно)

3. Запись данных в базу

- Поток: Сохранённые данные

- Хранилище: Основная база данных

4. Передача данных в процессинговый центр

- Поток: Информация о платеже

- Внешняя сущность: Процессинговый центр

5. Получение и обработка статуса операции

- Поток: Статус операции

- Внешняя сущность: Клиент

- Хранилище: Логи операций

Словесное описание взаимодействий:

- Клиент формирует платёжное поручение в системе.

- Оно поступает в процесс “Приём данных”, где сохраняется во временное хранилище.

- Далее поручение валидируется: если есть ошибки, они отправляются в хранилище ошибок, иначе — на запись в основную базу.

- После успешной записи поручение передаётся во внешний процессинговый центр.

- Полученное подтверждение статуса обрабатывается и отправляется обратно клиенту, а также фиксируется в логах операций.

Пояснение:

- DFD позволяет наглядно увидеть, как происходит движение данных на всех этапах — от ввода пользователем до финальной обработки и получения результата.

- Такая схема помогает выявить потенциальные “узкие места” (например, задержки на этапе валидации или передачи данных), а также точки контроля для тестирования корректности обработки и передачи информации.

Вывод:

DFD для передачи данных в финансовую систему обеспечивает прозрачность и формализованность бизнес-процесса, облегчает анализ архитектуры, поиск ошибок и подготовку качественных тест-кейсов.

Анализ уязвимостей в финансовых системах

Анализ уязвимостей — это процесс выявления, оценки и классификации слабых мест в программном обеспечении, инфраструктуре и бизнес-процессах финансовых систем, которые могут быть использованы злоумышленниками для нанесения ущерба. В финансовой сфере этот процесс критически важен из-за высокой стоимости ошибок и строгих требований к безопасности.

1. Основные этапы анализа уязвимостей:

1. Идентификация активов и границ системы

Определяются все компоненты системы, включая серверы, базы данных, приложения, сетевые устройства, точки интеграции с внешними сервисами.

2. Сбор информации

Использование автоматизированных сканеров (например, Nessus, Qualys, OpenVAS) и ручных методов для поиска известных уязвимостей, неправильных настроек, устаревших компонентов.

3. Анализ архитектуры и кода

Проводится ревью исходного кода, архитектурных схем, анализ логики бизнес-процессов, с целью поиска ошибок проектирования, небезопасного обращения с данными, ошибок авторизации и аутентификации.

4. Оценка уязвимостей

Каждая найденная уязвимость классифицируется по степени риска (например, по шкале CVSS), оценивается её влияние на бизнес и вероятность эксплуатации.

5. Проверка на практике (эксплуатация)

Иногда проводится контролируемое использование уязвимостей (penetration testing), чтобы убедиться в их реальной опасности и оценить возможные последствия.

6. Документирование и рекомендации

Формируется отчёт с перечнем уязвимостей, их описанием, степенью критичности и рекомендациями по устранению (патчи, изменение конфигураций, обновление ПО, усиление контроля доступов и т.д.).

2. Типовые уязвимости для финансовых систем:

- SQL-инъекции и другие виды инъекций

Позволяют злоумышленнику получить доступ к данным, изменять их или выполнять произвольные команды в системе.

- Недостаточная аутентификация и авторизация

Ошибки, позволяющие неавторизованным пользователям получить доступ к функциям или данным.

- Уязвимости веб-приложений

XSS, CSRF, небезопасная обработка сессий, отсутствие шифрования.

- Ошибки в интеграциях и API

Открытые или слабо защищённые API, уязвимости при обмене данными с внешними сервисами.

- Неправильное управление конфигурациями и правами доступа

Оставленные “дефолтные” пароли, доступ к административным панелям, избыточные привилегии.

- Использование устаревших библиотек и компонентов

Известные уязвимости в сторонних библиотеках, которые не были обновлены.

3. Особенности анализа уязвимостей в финансовых системах:

- Обязательное тестирование на соответствие стандартам (PCI DSS, ГОСТ, ISO 27001).

- Проверка защищённости всех точек интеграции с внешними сервисами.

- Анализ бизнес-логики: обнаружение уязвимостей, специфичных для финансовых процессов (например, двойное списание, обход лимитов, некорректные расчёты).

- Регулярное проведение penetration testing и security audit.

- Внедрение автоматических средств мониторинга (SIEM, IDS/IPS).

4. Пример процесса анализа уязвимостей:

1. Сканирование инфраструктуры с помощью автоматизированного инструмента (например, Nessus).

2. Ручная проверка критичных бизнес-процессов (например, обработка платежей, возвраты, начисление процентов).

3. Анализ логов и журналов безопасности.

4. Проведение тестирования на проникновение.

5. Подготовка отчёта с найденными уязвимостями, оценкой риска и планом их устранения.

5. Значение для бизнеса:

- Снижение риска финансовых потерь.

- Предотвращение штрафов и санкций за нарушение стандартов.

- Повышение доверия клиентов и партнёров.

- Своевременное выявление и устранение опасных уязвимостей.

Вывод:

Анализ уязвимостей — это неотъемлемая часть жизненного цикла финансовых систем, требующая комплексного и регулярного подхода с применением как автоматизированных, так и ручных методов. Только так можно обеспечить высокий уровень безопасности и устойчивости к современным киберугрозам.

IDEF3 (Process Description Capture Method): Теория

IDEF3 — это методология моделирования, предназначенная для описания и анализа последовательности действий (процессов) в бизнес-системах. В отличие от IDEF0, который фокусируется на функциональной декомпозиции (что делается), IDEF3 моделирует именно как и \*\*в какой последовательности\*\* выполняются действия, кто за них отвечает, какие события, условия и варианты развития процессов существуют.

Основные цели IDEF3:

- Фиксация сценариев поведения системы: описание реальных и предполагаемых последовательностей операций, событий, альтернатив и исключений.

- Документирование “живых” бизнес-процессов: не только “идеального” хода, но и всех возможных ветвлений, ошибок, возвратов, параллельных действий.

- Упрощение анализа, оптимизации и автоматизации процессов.

Основные элементы IDEF3:

1. Процессовые схемы (Process Description Schematics, PDS):

Графические диаграммы, отображающие последовательности событий, блоки обработки, точки принятия решений и параллелизма.

2. Объектные схемы (Object State Transition Networks, OSTN):

Описывают, как меняются состояния объектов системы под влиянием событий и процессов.

3. Юниты поведения (UOB — Unit of Behavior):

Базовые “кирпичики” модели — отдельные действия, события или процессы.

4. Связи (Links, Junctions):

Определяют последовательность, условия перехода, ветвления (OR, AND, XOR), начало и завершение процессов.

Как строится модель IDEF3?

1. Определение границ процесса: что моделируется, с какими внешними объектами/процессами взаимодействует.

2. Выделение ключевых событий и действий (UOB): что происходит на каждом шаге.

3. Построение последовательности действий: с помощью блоков и связей изображается “поток” процесса, включая альтернативные и параллельные ветвления.

4. Добавление условий и сценариев (ветвления, исключения, возвраты): что происходит при ошибках, какие есть варианты развития событий.

5. Документирование контекстов и описаний\*\*: для каждого блока указывается, кто его выполняет, какие требования предъявляются к входам/выходам.

Пример:

- Процесс “Обработка платежа” может выглядеть так:

1. Получение заявки → Проверка данных → [Ветвление: если ошибки — возврат на доработку, если всё верно — продолжение → Проведение платежа → Формирование отчёта.

2. Параллельно может идти отправка уведомления клиенту.

Преимущества IDEF3:

- Чётко отображает реальные сценарии, включая все исключения и альтернативы.

- Полезен для анализа сложных процессов, где важна последовательность и условия перехода.

- Является отличной основой для автоматизации, оптимизации и последующего тестирования процессов.

- Обеспечивает прозрачность и понятность даже для неподготовленных участников.

Применение в тестировании и бизнес-аналитике:

- На основе IDEF3 строятся сценарные тест-кейсы, учитывающие не только “прямой путь”, но и все альтернативные (ошибочные, возвратные) сценарии.

- Позволяет выявить “узкие места”, неочевидные ошибки, точки отказа и риски, которые часто пропускаются в процессах без детального сценарного анализа.

Вывод:

IDEF3 — это мощный инструмент для сценарного моделирования процессов, позволяющий глубоко анализировать, оптимизировать и тестировать бизнес-системы с учётом всех возможных вариантов хода событий, что особенно важно для сложных и критичных к сбоям отраслей, таких как финансы, производство, логистика.

Пример IDEF3 для моделирования процесса обработки сбоев в финансовой системе

Рассмотрим пример: в финансовой системе происходит обработка платёжных поручений, и требуется описать сценарий действий при возникновении сбоя (например, ошибка соединения с процессинговым центром или внутренний сбой компонента).

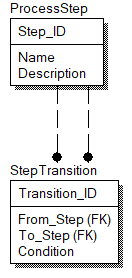
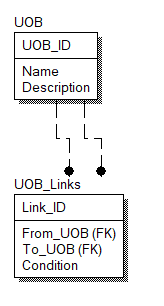
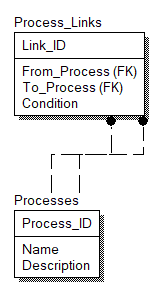


Таблица: ProcessStep (Этап процесса)

Графические элементы IDEF3:

Упрощенная схема IDEF3:

Примечания:

- В модели IDEF3 чётко фиксируются все варианты развития событий: штатный путь, обработка ошибок, повторные попытки, эскалация.

- Для каждого UOB (unit of behavior) можно добавить описание: кто исполняет (система/оператор), какие требования к входам/выходам, дополнительные условия.

- Такой сценарий позволяет формировать подробные тест-кейсы на сбои, прорабатывать автоматизацию мониторинга и эскалации, снижать риски задержек и потерь при инцидентах.

Вывод:

IDEF3 отлично подходит для описания и анализа реакций системы на сбои, поскольку позволяет явно моделировать как “прямые” (штатные), так и “альтернативные” (аварийные) сценарии с ветвлениями, возвратами и параллельными действиями.

Анализ возможных проблем в процессе тестирования финансовых систем

Анализ возможных проблем — это этап, на котором выявляются потенциальные трудности и риски, способные негативно повлиять на качество тестирования, безопасность и работоспособность финансовой системы. Такой анализ позволяет заранее подготовиться к сложным ситуациям, минимизировать влияние ошибок и повысить устойчивость процессов.

1. Типовые группы возможных проблем:

1.1. Организационные проблемы

- Неясные или противоречивые требования к системе.

- Отсутствие чёткой коммуникации между заказчиком, аналитиками, тестировщиками и разработчиками.

- Недостаточная квалификация или нехватка персонала.

- Нарушение сроков тестирования из-за изменений в проекте.

1.2. Процессные проблемы

- Неполное или недостаточно формализованное моделирование бизнес-процессов (например, пропущены исключения, альтернативные сценарии).

- Неучтённые интеграции с внешними сервисами.

- Отсутствие автоматизации рутинных проверок.

- Слабый контроль версий и тестовых данных.

- Недостаточное покрытие тестами (особенно негативных и граничных сценариев).

1.3. Технические проблемы

- Ошибки в архитектуре: “узкие места”, неэффективные алгоритмы, точки отказа.

- Использование устаревших или уязвимых компонентов.

- Сложности с настройкой тестовых сред, различия между тестовой и продуктивной инфраструктурой.

- Проблемы с производительностью и масштабируемостью.

1.4. Проблемы безопасности

- Недостаточная защита тестовых данных (особенно если используются реальные данные клиентов).

- Уязвимости, связанные с обработкой данных, авторизацией, аутентификацией.

- Отсутствие регулярного анализа и устранения уязвимостей.

1.5. Проблемы управления изменениями

- Неоперативное обновление тестовой документации при изменениях в системе.

- Нарушение согласованности между версиями компонентов.

- Сложности при откате изменений или воспроизведении багов.

2. Методы анализа возможных проблем:

- Мозговой штурм с командой (разработка «чек-листов» проблем).

- Анализ прошлых инцидентов (ретроспектива).

- Использование формальных методов моделирования (IDEF0, DFD, IDEF3) для поиска “узких мест”.

- Ревью требований и тест-кейсов с привлечением экспертов.

- SWOT-анализ\*\* (Strengths, Weaknesses, Opportunities, Threats).

- Ранжирование рисков по вероятности и влиянию.

3. Результаты анализа:

- Список потенциальных проблем с оценкой вероятности и критичности.

- Предложения по мерам снижения рисков (дополнительные тесты, автоматизация, обучение, резервирование).

- Рекомендации по улучшению процессов и коммуникаций.

- Основа для дальнейшего мониторинга и улучшения качества тестирования.

Вывод

Анализ возможных проблем — важная часть подготовки к тестированию финансовых систем. Он позволяет выявить риски до начала работ, обеспечить устойчивость процессов, повысить надёжность и безопасность конечного продукта, а также своевременно реагировать на возникающие трудности.

Разработка комплексной методики тестирования

Обоснование выбора инструментов

1. Критерии выбора инструментов для комплексного тестирования финансовых систем:

- Соответствие бизнес-требованиям и регуляторным стандартам. Инструменты должны поддерживать формализацию требований, автоматизацию аудита и протоколирования (соответствие PCI DSS, ГОСТ, ISO 27001 и др.).

- Поддержка различных видов тестирования. Для комплексного охвата нужны средства для функционального, интеграционного, нагрузочного, регрессионного, безопасности и автоматизированного тестирования.

- Интеграция с CI/CD и DevOps. Поддержка автоматического запуска тестов, сбора метрик и быстрого обратного отклика разработчикам.

- Масштабируемость и гибкость. Возможность расширения тестового покрытия и интеграции с внешними сервисами.

- Удобство анализа результатов и визуализации. Возможность построения отчетов, графиков, хранения истории тестов.

- Сообщество и поддержка. Активная поддержка инструментов, наличие документации, примеров, расширяемость.

2. Категории инструментов и обоснование выбора:

A. Инструменты для моделирования процессов

- IDEF0/DFD/IDEF3 (например, Cawemo, Bizagi Modeler, Visual Paradigm):

- Позволяют формализовать бизнес-процессы, выявить “узкие места”, определить контрольные точки для тестирования.

- Обеспечивают прозрачность и понятность процессов для всех участников (ИТ, бизнес, аудиторы).

- Упрощают построение тест-кейсов на основе моделей.

B. Автоматизированное тестирование (UI/API)

- Selenium, Playwright, Cypress (UI):

- Позволяют автоматизировать проверку пользовательских сценариев, сокращают время регрессионного тестирования.

- Легко интегрируются с CI/CD.

- Postman, SoapUI, Rest Assured (API):

- Автоматизируют тестирование интеграций, позволяют валидировать корректность передачи данных между внутренними и внешними сервисами.

- Поддерживают создание коллекций, сценариев и параметризацию.

C. Нагрузочное и стресс-тестирование

- Jmeter, Gatling, Locust:

- Необходимы для оценки производительности системы, поиска “узких мест” под высокой нагрузкой.

- Позволяют моделировать реалистичные сценарии массового использования.

D. Тестирование безопасности

- OWASP ZAP, Burp Suite, Nessus:

- Автоматически выявляют наиболее распространённые уязвимости (XSS, SQL-инъекции, CSRF и др.).

- Позволяют проводить регулярный аудит соответствия стандартам безопасности.

E. Управление тест-кейсами и результатами

- TestRail, Zephyr, Allure TestOps:

- Централизованное хранение тест-кейсов, истории запусков, автоматизация формирования отчетности.

- Упрощают анализ покрытия и выявление «белых пятен» в тестировании.

F. Интеграция и мониторинг

- Jenkins, GitLab CI/CD, TeamCity:

- Организация автоматического запуска тестов при каждом изменении кода.

- Сбор логов, метрик, быстрый фидбек для команды.

- Grafana, Kibana, Prometheus:

- Мониторинг метрик производительности, ошибок, инцидентов во время тестирования.

3. Примерная схема применения инструментов в комплексной методике:

1. Моделирование бизнес-процессов — формализация требований и построение тест-кейсов (Cawemo, Bizagi, Visual Paradigm).

2. Автоматизация тестов — реализация и запуск автоматизированных сценариев (Selenium, Postman).

3. Нагрузочное тестирование — проверка устойчивости системы (Jmeter, Gatling).

4. Тестирование безопасности — аудит уязвимостей (OWASP ZAP, Nessus).

5. Управление тестированием — планирование и анализ результатов (TestRail, Allure).

6. Интеграция с CI/CD — автоматизация процессов и мониторинг (Jenkins, Grafana).

4. Вывод:

Выбор инструментов для комплексной методики тестирования должен быть обусловлен спецификой финансовых систем, необходимостью соответствия стандартам, поддержкой автоматизации, интеграции и возможностью масштабирования. Использование набора проверенных инструментов позволяет существенно повысить качество тестирования, минимизировать риски и повысить доверие к разрабатываемым решениям.

Этапы построения моделей (на примере бизнес-процессов и информационных систем)

Построение моделей — важный этап анализа, проектирования и тестирования сложных систем (например, финансовых). Универсальная последовательность этапов подходит для большинства нотаций: IDEF0, DFD, IDEF3 и других.

1. Определение целей моделирования

- Для чего строится модель (автоматизация, оптимизация, тестирование, аудит, обучение).

- Какая проблема/область описывается (например, процесс обработки платежа).

2. Выделение границ моделируемой системы

- Определение входов и выходов системы.

- Определение внешних участников (клиенты, сервисы, партнёры).

3. Сбор и анализ информации

- Изучение документации, нормативов, интервью с экспертами.

- Анализ существующих процессов, схем, регламентов.

4. Идентификация ключевых функций, процессов и данных

- Определение основных действий, этапов или событий (например, "Проверка данных", "Передача в процессинговый центр").

- Определение потоков информации и их типов (данные, документы, команды).

5. Формализация требований и построение первичной (контекстной) диаграммы

- Создание схемы верхнего уровня (контекстной), показывающей всю систему одним блоком и её взаимодействие с внешними объектами.

6. Декомпозиция процессов

- Разбиение общих функций на более мелкие, до необходимого уровня детализации.

- Построение диаграмм следующих уровней (Level 1, 2…).

7. Определение связей, потоков и условий

- Уточнение маршрутов движения данных, условий переходов, сценариев ветвления и возврата.

- Указание хранилищ данных, точек контроля и ролей.

8. Проверка и валидация модели

- Совместная ревизия схем с ключевыми участниками процесса.

- Проверка на полноту, отсутствие противоречий, соответствие требованиям.

9. Документирование и описание элементов модели

- Добавление пояснений, описаний к блокам, уточнение ролей, условий, ограничений.

- Формирование справочной документации.

10. Актуализация и поддержка модели

- Регулярное обновление моделей при изменениях процессов или требований.

- Использование модели как основы для построения тест-кейсов, автоматизации, обучения персонала.

Вывод:

Этапы построения моделей позволяют системно и поэтапно формализовать даже сложные бизнес-процессы, выявить риски, упростить коммуникацию между участниками проекта и заложить основу для дальнейшей автоматизации, тестирования и оптимизации.

Интеграция модели в тестируемом процессе

Интеграция модели — это внедрение построенной аналитической или процессной модели (например, DFD, IDEF0, IDEF3) непосредственно в жизненный цикл тестирования для управления, оптимизации и контроля тестируемого процесса. Такой подход обеспечивает системность, прозрачность и полноту тестирования, особенно в сложных финансовых и ИТ-системах.

1. Цели интеграции модели:

- Формализация требований и сценариев: модель служит основой для создания корректных, полных и релевантных тест-кейсов.

- Контроль полноты тестирования: покрытие всех функций, сценариев, исключений и альтернатив.

- Выявление и устранение “слепых зон”: обнаружение процессов или ветвлений, которые могли быть упущены при ручном описании.

- Обеспечение соответствия регуляторным и внутренним стандартам.

2. Этапы интеграции модели в тестируемом процессе:

a. Построение и валидация модели

- Формализуются процессы, потоки данных, условия, роли — с помощью выбранной нотации (DFD, IDEF0, IDEF3 и др.).

- Модель согласовывается с бизнес- и ИТ-экспертами.

b. Связь модели с артефактами тестирования

- На каждый элемент модели (процесс, переход, сценарий, хранилище) формируется связанный набор тест-кейсов.

- В матрице трассировки фиксируется соответствие между элементами модели и тестами.

c. Использование модели при проектировании тестов

- Модель используется для генерации сценариев тестирования: как “прямых” (основных), так и “альтернативных” (исключения, сбои, возвраты).

- Модели с ветвлениями (IDEF3, DFD) позволяют выявлять и тестировать все возможные пути прохождения процесса.

d. Автоматизация тестирования на основе модели

- Интеграция с инструментами автоматизации: генерация тест-кейсов, сценариев тестирования, автотестов по структуре и логике модели.

- Использование моделей для контроля полноты покрытия автотестами.

e. Анализ результатов и обратная связь

- По итогам тестирования результаты соотносятся с элементами модели для оценки полноты и качества тестирования.

- Выявленные дефекты анализируются “по модели” — можно быстро локализовать проблемную зону в бизнес-процессе.

f. Обновление модели

- При изменениях в процессах или системе модель актуализируется, что позволяет синхронизировать тест-кейсы и поддерживать их релевантность.

3. Преимущества интеграции модели:

- Прозрачность: любой участник проекта может проследить, как требования реализованы в тестах и какие пути покрыты.

- Системность: все ветвления, условия, исключения и роли учтены.

- Гибкость: при изменении процессов можно быстро обновить модель и тесты.

- Контролируемость: легко контролировать полноту покрытия и управлять качеством тестирования.

4. Пример:

- Для процесса "Обработка платежа" моделируется DFD/IDEF3-схема, в которой каждая стрелка и блок соответствует отдельному тест-кейсу (например, “Проверка данных”, “Ошибочный ввод”, “Сбой передачи”, “Успешная обработка”).

- Все тест-кейсы формируются на основе модели. При изменении бизнес-процесса (например, добавлении новой проверки) модель сразу обновляется, и появляются новые тесты, которые автоматически связываются с соответствующими элементами.

Вывод:

Интеграция модели в тестируемом процессе — это современный и эффективный подход, позволяющий не только повысить качество и полноту тестирования, но и сделать процесс тестирования гибким, управляемым и прозрачным для всех участников проекта.

Практическая часть: Ключи обновления для финансовых систем (на примере местного банка или криптобиржи)

1. Что такое “ключи обновления” в финансовых системах?

В контексте тестирования и анализа финансовых систем “ключи обновления” — это триггеры, параметры или точки, при изменении которых система обязана корректно обработать, зафиксировать и синхронизировать свои данные. Они критичны, так как связаны с изменениями балансов, статусов транзакций, курсов, лимитов, условий обслуживания и т.д.

2. Примеры ключей обновления для местного банка:

- Изменение баланса счёта клиента

(поступления, списания, начисление процентов, комиссии, возвраты)

- Обновление статуса платёжного поручения

(отправлено, обработано, отклонено, выполнено, в очереди)

- Изменение лимитов на операции

(дневные/месячные лимиты, лимиты по картам)

- Обновление справочников

(курсы валют, актуальные тарифы, реквизиты банка)

- Изменения в профиле клиента

(смена контактных данных, статуса KYC, уровня доступа)

- Обновления расписаний платёжных систем

(график работы, технические окна)

3. Примеры ключей обновления для криптобиржи:

- Обновление баланса пользователя

(депозиты, выводы, трейдинг, списания комиссий)

- Изменение статуса ордера

(создан, частично исполнен, исполнен, отменён)

- Обновление курсов криптовалют

(тикеры, спотовые цены, исторические данные)

- Обновление лимитов аккаунта

(лимиты на ввод/вывод, торговые лимиты)

- Смена статуса верификации пользователя

- Изменение параметров API-ключей

(создание, отзыв, изменение прав)

- Обновления по листингу/делистингу монет

- Технические обновления (release/maintenance)

4. Практическое применение (тест-кейсы):

Пример 1. Проверка обработки обновления баланса (местный банк):

- Провести платёж, убедиться, что баланс обновился мгновенно и корректно.

- Проверить, что в истории операций отражено последнее обновление.

- Изменить баланс принудительно (например, через админ-панель) — проверить корректность отображения на клиентских интерфейсах и в отчетности.

Пример 2. Проверка обновления статуса ордера (криптобиржа):

- Создать ордер, инициировать его исполнение частично (matched), затем полностью.

- Проверить, что изменения статуса моментально видны в UI, API и истории пользователя.

- Попробовать отменить ордер на каждом этапе — убедиться, что статусы обновляются корректно, а остатки возвращаются на баланс.

Пример 3. Проверка обновления лимитов:

- Изменить лимит на вывод средств для клиента.

- Сделать попытку вывода — убедиться, что новый лимит применяется немедленно и корректно.

5. Важные аспекты при тестировании обновлений:

- Согласованность: обновления должны быть мгновенно отражены во всех связанных подсистемах (балансы, отчёты, уведомления).

- Логирование: каждое обновление должно регистрироваться в логах для аудита.

- Реакция на конкурирующие изменения: система должна корректно отрабатывать параллельные обновления (например, два одновременных списания).

- Восстановление после сбоев: при нештатных ситуациях (обрыв соединения, crash) обновления не должны теряться или дублироваться.

- Безопасность: обновления ключей (особенно лимитов, API-ключей) требуют строгой аутентификации и авторизации.

Вывод:

Ключи обновления — важнейшие точки контроля в тестировании финансовых систем. Их правильная обработка обеспечивает целостность, прозрачность и безопасность операций как в банках, так и на криптобиржах. В тест-кейсах ключи обновления должны быть покрыты максимально подробно, с учётом всех сценариев изменений и возможных сбоев.

Построение моделей: IDEF0, DFD, IDEF3

1. IDEF0 (Functional Modeling)

IDEF0 — методика для функционального моделирования сложных систем. Позволяет описать, КАКИЕ функции выполняются в системе и КАК они связаны между собой.

Основные элементы:

- Функция (блок): изображается прямоугольником.

- Стрелки:

- Input (вход): слева

- Output (выход): справа

- Control (управление): сверху

- Mechanism (механизм, исполнители): снизу

Пошаговое построение:

1. Определить главную функцию/процесс (например, “Обработка заявки на платёж”).

2. Нарисовать контекстную диаграмму — один блок, стрелки входа/выхода/контроля/механизмов.

3. Декомпозировать функцию — разбить основной процесс на подфункции (например, “Проверка данных”, “Проведение платежа”, “Формирование отчёта”).

4. Связать блоки стрелками согласно логике процесса.

5. Добавить описание стрелок и блоков.

2. DFD (Data Flow Diagram)

DFD — методика описания потоков данных между процессами, хранилищами и внешними сущностями.

Основные элементы:

- Процессы: круги или прямоугольники с закруглёнными углами.

- Потоки данных: стрелки.

- Хранилища данных: открытые прямоугольники (две линии).

- Внешние сущности: прямоугольники.

Пошаговое построение:

1. Построить контекстную диаграмму (Level 0): система как единый процесс, отображены только внешние связи.

2. Декомпозировать процессы (Level 1, 2…): каждый процесс разбивается на подпроцессы, показываются внутренние потоки данных и хранилища.

3. Пронумеровать процессы и потоки для однозначности.

4. Добавить описание каждого элемента схемы (что за поток, какие данные, кто участник).

3. IDEF3 (Process Description Capture)

IDEF3 — методика сценарного моделирования, акцент на последовательность, условия, альтернативы и варианты развития событий.

Основные элементы:

- UOB (Unit of Behavior): прямоугольник — шаг/действие процесса.

- Стрелки: показывают последовательность.

- Junctions (развилки): ромбы — условия (AND/OR/XOR).

- Объектные схемы: изменение состояния объектов под воздействием событий.

Пошаговое построение:

1. Определить границы процесса и ключевые события/шаги.

2. Построить линейную последовательность UOB.

3. Добавить развилки (Junctions) для альтернативных сценариев, ошибок, возвратов.

4. Описать условия переходов и результаты.

5. (Дополнительно) Построить объектные схемы изменения состояния.

Пример для процесса “Обработка платежа”

IDEF0:

- Главная функция: “Обработка платежа”

- Вход: Платёжное поручение

- Контроль: Регламенты, лимиты

- Механизм: Оператор/Система

- Выход: Статус, отчёт

DFD:

- Внешние сущности: Клиент, Процессинг

- Процессы: Приём заявки, Валидация, Запись в БД, Передача в процессинг, Получение статуса

- Потоки: Поручение, Статус, Ошибка

- Хранилища: База заявок, Логи

IDEF3:

- UOB: Получить заявку → Проверить данные → [Ветвление: ошибка/успех] → Передать в процессинг → Получить статус → Завершить

- Развилка: при ошибке — возврат клиенту, при успехе — завершение.

Вывод:

Для построения моделей IDEF0, DFD и IDEF3 важно понимать цели моделирования, выделять ключевые шаги и связи, четко придерживаться формальных правил построения диаграмм. Это обеспечивает прозрачность процессов, выявляет риски, облегчает тестирование и коммуникацию между командами.

Анализ выявленных уязвимостей

Анализ выявленных уязвимостей — это этап, на котором найденные в ходе тестирования или аудита уязвимости оцениваются по степени риска, влияния на бизнес и вероятности эксплуатации. Такой анализ помогает приоритизировать усилия по устранению проблем и минимизировать угрозы для финансовой системы (или любой другой критически важной ИТ-системы).

1. Классификация уязвимостей

- По типу:

- Ошибки авторизации и аутентификации

- Инъекции (SQL, Command Injection, и др.)

- XSS, CSRF, IDOR

- Уязвимости в API

- Ошибки конфигурации

- Использование устаревших или небезопасных компонентов

- По уровню риска:

- Критические (приводят к несанкционированному доступу, потере денег или данных)

- Высокие

- Средние

- Низкие

2. Оценка риска (примерная схема):

- Вероятность эксплуатации:

Насколько просто злоумышленнику воспользоваться уязвимостью? (Публично известная, требует специальных знаний, требует внутреннего доступа)

- Влияние на систему:

Какие последствия для бизнеса? (Потеря данных, финансовые убытки, простои, нарушение регуляторных требований)

- Возможные сценарии атаки:

Какие действия может предпринять злоумышленник, используя уязвимость?

- Связанные уязвимости:

Может ли комбинация уязвимостей привести к более серьёзным последствиям?

3. Документирование и приоритизация:

- Включить в отчёт:

- Описание уязвимости (в том числе с примерами эксплуатации)

- Степень критичности (например, по CVSS)

- Влияние на бизнес-процессы

- Рекомендации по устранению (технические и административные)

- Сроки устранения (SLA)

- Пример записи в анализе:

- Уязвимость: SQL-инъекция в модуле “Платежи”

- Критичность: Критическая

- Возможные последствия: Неавторизованный доступ к данным клиентов, возможность проведения несанкционированных операций

- Рекомендации: Внедрить параметризированные запросы, провести ревью кода, добавить автоматические проверки на инъекции

- Срок устранения: 2 дня

4. Контроль устранения и повторная проверка:

- Проверить, что уязвимость устранена (повторное тестирование).

- Убедиться, что не появилось новых ошибок (регистрировать регрессионные баги).

- Вести журнал устранённых уязвимостей и отслеживать сроки.

5. Использование результатов анализа:

- Формирование плана по усилению безопасности.

- Обучение команды (разработчики, тестировщики, администраторы).

- Улучшение процессов разработки и тестирования (внедрение автоматизированных сканеров, ревью кода, контроль исходных данных).

Вывод:

Анализ выявленных уязвимостей позволяет не только своевременно закрывать критические “дыры” в безопасности, но и формировать культуру безопасной разработки и эксплуатации систем. Это важный элемент обеспечения устойчивости и доверия к информационной системе, особенно в финансовой сфере.

Рекомендации по обеспечению безопасности и качества в финансовых системах

1. Технические рекомендации

- Внедряйте регулярный анализ уязвимостей:

Используйте автоматизированные сканеры (Nessus, OpenVAS, OWASP ZAP) и ручное тестирование для поиска слабых мест, особенно после обновлений и релизов.

- Проводите пентесты и моделирование атак:

Регулярно имитируйте действия злоумышленников, чтобы выявлять реальные векторы угроз.

- Используйте современные средства мониторинга:

Внедряйте системы SIEM, мониторинг журналов событий, настройте оповещения о подозрительной активности.

- Автоматизируйте тестирование:

Развивайте автоматизированные проверки (UI, API, нагрузочные тесты) для сокращения “ручной” работы и повышения покрытия.

- Обновляйте компоненты и библиотеки:

Отслеживайте устаревшие или уязвимые зависимости, своевременно устанавливайте патчи и обновления.

2. Процессные рекомендации

- Формализуйте процессы тестирования:

Используйте методики моделирования (IDEF0, DFD, IDEF3) для построения прозрачных схем бизнес-процессов и контроля полноты тестов.

- Ведите документацию и матрицу трассировки:

Соотносите требования, сценарии, тест-кейсы и результаты, чтобы не упустить критичные сценарии.

- Проводите ревью требований и кода:

Включайте экспертов по безопасности и бизнес-аналитиков в ревью.

- Обеспечьте обучение и повышение квалификации сотрудников:

Регулярные тренинги по новым угрозам, инструментам, стандартам (PCI DSS, ISO 27001).

3. Организационные рекомендации

- Разделяйте доступы по принципу минимальных привилегий:

Не выдавайте лишних прав ни для пользователей, ни для сервисных аккаунтов.

- Настройте резервное копирование и план реагирования на инциденты:

Регулярно тестируйте восстановление из резервных копий, имейте план действий при атаках или сбоях.

- Контролируйте работу с персональными и финансовыми данными:

Обеспечьте шифрование при передаче и хранении, ведите аудит доступа.

- Внедряйте двухфакторную аутентификацию и дополнительные меры защиты для критичных операций.

4. Рекомендации по совершенствованию тестирования

- Используйте модели процессов как основу для тест-кейсов:

Это позволяет выявлять не только штатные, но и альтернативные (аварийные) сценарии.

- Проводите анализ инцидентов и ретроспективу тестирования:

Изучайте причины ошибок для предотвращения повторения в будущем.

- Внедряйте автоматические отчёты и дашборды:

Для прозрачного мониторинга прогресса и покрытия тестами.

- Следите за актуальностью тестовой документации:

Своевременно обновляйте её при изменениях бизнес-процессов или требований.

Вывод:

Комплексный подход, сочетающий технические, процессные и организационные меры, позволяет существенно повысить безопасность, устойчивость и надёжность финансовых систем, минимизировать риски и ускорить развитие бизнеса.

Заключение

В ходе работы была рассмотрена комплексная методика тестирования финансовых систем с использованием современных методов моделирования (IDEF0, DFD, IDEF3), анализа рисков, построения и интеграции моделей, а также анализа уязвимостей. Проведен разбор этапов построения моделей, интеграции их в тестируемые процессы, рассмотрены примеры ключевых точек обновления (на примере банков и криптобирж) и даны рекомендации по обеспечению качества и безопасности.

Основные выводы:

- Использование формальных методов моделирования (IDEF0, DFD, IDEF3) позволяет структурировать бизнес-процессы, выявить “узкие места”, повысить прозрачность и полноту тестирования.

- Интеграция моделей в тестируемый процесс обеспечивает контроль полноты покрытия тест-кейсами, облегчает выявление и локализацию проблем, способствует быстрой актуализации тестовой документации при изменениях в бизнес-процессах.

- Анализ уязвимостей и внедрение регулярного мониторинга, автоматизации тестирования, а также обучение персонала существенно повышают уровень безопасности и доверия к системе.

- Выбор инструментов и построение тестовой инфраструктуры должны основываться на специфике финансовой сферы, требованиях регуляторов, а также поддерживать масштабируемость, гибкость и интеграцию с CI/CD.

- Практическая часть показывает важность контроля ключевых точек обновления, тщательной проверки всех сценариев изменений и мгновенной реакции системы на сбои и инциденты.

- Комплексный подход — сочетание технических, процессных и организационных мер — является ключом к успешному обеспечению качества и безопасности в современных финансовых системах.

Таким образом, внедрение комплексной методики тестирования, основанной на моделировании и анализе, позволяет не только минимизировать риски, но и значительно повысить надежность, прозрачность и конкурентоспособность финансовых продуктов.

Перспективы развития

Современные финансовые системы и процессы тестирования для них находятся в постоянной динамике, что определяется как технологическими инновациями, так и возрастающими требованиями к безопасности, скорости внедрения и качеству сервисов. Перспективы развития в данной области включают:

1. Углубление автоматизации тестирования

- Интеллектуальные автотесты:

Внедрение искусственного интеллекта и машинного обучения для генерации тест-кейсов, анализа результатов и обнаружения аномалий.

- Self-healing тесты:

Автоматическое адаптирование тестов при изменениях интерфейса или логики приложения, что снижает затраты на поддержку тестовой инфраструктуры.

- Интеграция с DevOps:

Развитие практик CI/CD, расширение автоматизации проверки безопасности (DevSecOps) на всех этапах жизненного цикла разработки.

2. Моделирование и симуляция сложных сценариев

- Digital Twin (цифровой двойник):

Создание виртуальных моделей финансовых процессов для тестирования в условиях, максимально приближенных к реальным, без рисков для бизнеса.

- Сценарии стресс-тестирования:

Автоматизация комплексных стресс- и катастрофических сценариев с моделированием массовых сбоев, атак и пиковых нагрузок.

3. Усиление безопасности

- Постоянный мониторинг уязвимостей:

Использование автоматизированных сканеров, интеграция с внешними базами угроз и оперативное реагирование.

- Shift-left security:

Перенос процессов обеспечения безопасности на более ранние этапы разработки и тестирования.

- Zero Trust и контроль доступа:

Внедрение концепций нулевого доверия для минимизации внутренних и внешних рисков.

4. Развитие моделирования процессов

- Автоматизация построения моделей:

Использование специализированных инструментов для автоматического построения и актуализации моделей IDEF0, DFD, IDEF3 на основе анализа кода, логов и метаданных.

- Связь моделей с метриками качества:

Автоматическое сопоставление элементов моделей с показателями покрытия, стабильности и времени отклика.

5. Расширение аналитики и отчетности

- Интерактивные дашборды:

Визуализация результатов тестирования, покрытия, уязвимостей и бизнес-метрик в реальном времени.

- Применение Big Data и AI для анализа инцидентов:

Глубокий анализ больших объемов данных для выявления скрытых тенденций, прогнозирования риска и оптимизации процессов тестирования.

6. Гибридные и облачные архитектуры

- Тестирование в облаке:

Использование облачных платформ для масштабирования тестовой среды, эмуляции распределенных систем и ускорения тестового цикла.

- Гибридные решения:

Комбинирование локальных и облачных ресурсов для оптимизации затрат и повышения гибкости.

Вывод:

Перспективы развития сфокусированы на автоматизации, интеллектуализации тестирования, усилении безопасности и гибкости, что позволяет не только минимизировать риски и издержки, но и быстрее внедрять инновации, поддерживать высокий уровень доверия со стороны клиентов и регуляторов.

Список литературы

1. Киселёв А.А. Моделирование бизнес-процессов. Практическое руководство. — М.: ДМК Пресс, 2020. ISBN: 978-5-97060-842-8

2. Шаронов В.А. Информационная безопасность финансовых организаций. — М.: КНОРУС, 2019. ISBN: 978-5-406-07174-7

3. Гостев А.Е. Информационная безопасность: Учебник для вузов. — М.: Академия, 2022. ISBN: 978-5-4468-1841-6

4. Yourdon E., DeMarco T. Structured Analysis and System Specification. — Prentice Hall, 1979. ISBN: 978-0138543801

5. Mayer, R.J., Menzel, C.P., Painter, M.K., et al. Information Integration for Concurrent Engineering (IDEF Family of Methods). — Knowledge Based Systems, Inc., 1995.

ISBN: 978-1883553033

6. Sommerville, I. Software Engineering (10th Edition). — Pearson, 2015. ISBN: 978-0133943033

7. Beizer B. Software Testing Techniques. — Dreamtech Press, 2002. ISBN: 978-8177222608

8. Канер К., Фолк Дж., Нгуен Х. Тестирование программного обеспечения. Основы профессионального подхода. — СПб.: Питер, 2016. ISBN: 978-5-496-01813-6

9. Ammann P., Offutt J. Introduction to Software Testing. — Cambridge University Press, 2016. ISBN: 978-1107172012

10. Куликов Д.Ю., Нечаев А.С. Безопасность банковских систем: учебное пособие. — М.: Финансы и статистика, 2019. ISBN: 978-5-279-03813-9