**Общие стандарты**

* **Какие международные стандарты (ISO, IEEE, ГОСТ) регулируют разработку программного обеспечения?**

ISO/IEC 12207

Основной стандарт, описывающий процессы жизненного цикла ПО. Он охватывает все этапы: от заказа и разработки до эксплуатации и сопровождения. Включает российский аналог ГОСТ Р ИСО/МЭК 12207—2010 26.

Актуальные редакции:

ISO/IEC 12207:2008 — базовая версия с процессами для ПО 2.

ISO/IEC/IEEE 12207:2017 — обновленная версия, интегрирующая системную инженерию и совместимая с ISO/IEC/IEEE 15288 (жизненный цикл систем) 811.

ISO/IEC/IEEE 15288

Стандарт для жизненного цикла систем, включая программные компоненты. Часто используется совместно с ISO/IEC 12207 для комплексного управления системами, где ПО является частью более крупной структуры 811.

ГОСТ 34 (российский стандарт)

Регулирует разработку автоматизированных систем, включая ПО. Например

ГОСТ 34.601-90 — этапы создания автоматизированных систем.

ГОСТ 34.003-90 — терминология и процессы 6.

ISO/IEC 15504 (SPICE)

Оценка процессов разработки ПО. Используется для аудита и улучшения процессов, описанных в ISO/IEC 12207 11.

ISO/IEC TR 15271 и ISO/IEC TR 16326

Руководства по применению ISO/IEC 12207, включая управление проектами и адаптацию процессов

* **Объясни принципы стандарта ISO/IEC 12207 (Жизненный цикл ПО).**

1. Гибкость и адаптивность

Стандарт не предписывает жестких этапов или методов, а предлагает набор процессов, которые можно адаптировать под конкретный проект. Например:

* Организация выбирает только необходимые процессы (например, разработку, сопровождение) 16.
* Допускает использование различных моделей жизненного цикла (каскадной, спиральной) 911.

2. Структура процессов

Процессы делятся на три категории:

* Основные:
  + Приобретение, поставка, разработка, эксплуатация, сопровождение 11.
  + Например, процесс разработки включает анализ требований, проектирование архитектуры, тестирование 26.
* Вспомогательные:
  + Управление конфигурацией, верификация, обеспечение качества, решение проблем 11.
* Организационные:
  + Управление проектом, обучение персонала, совершенствование процессов 11.

3. Взаимодействие сторон

* Регулирует отношения между заказчиком, поставщиком и разработчиком, даже если они принадлежат одной организации 16.
* Например, процесс поставки определяет обязанности поставщика, а процесс приобретения — требования заказчика 11.

4. Интеграция с системной инженерией

* Совместим со стандартом ISO/IEC 15288, что позволяет синхронизировать процессы разработки ПО и систем 811.
* Учитывает, что ПО редко существует изолированно — оно часть более крупной системы 8.

5. Документирование и контроль

* Акцент на управление документацией, конфигурацией и рисками 11.
* Например, процесс аудита проверяет соответствие продуктов и процессов установленным требованиям 11.

Пример применения ISO/IEC 12207

При разработке ПО для банковской системы:

1. Приобретение: Заказчик формирует требования к безопасности и функционалу 1.
2. Разработка: Команда создает архитектуру, проводит тестирование (квалификационное, интеграционное) 2.
3. Сопровождение: Управление обновлениями и исправление ошибок после внедрения

* **В чем разница между IEEE 829 и ISO/IEC/IEEE 29119 в тестировании ПО?**

IEEE 829 (Standard for Software Test Documentation) и ISO/IEC/IEEE 29119 (серия стандартов по тестированию ПО) различаются по охвату, структуре и подходу к тестированию:

* Сфера применения
  + IEEE 829 фокусируется исключительно на документации тестирования, включая шаблоны для тест-планов, тест-кейсов и отчетов 15.
  + ISO/IEC/IEEE 29119 охватывает все аспекты тестирования: процессы, документацию, техники тест-дизайна и ключевые слова для автоматизации. Это комплексный стандарт, заменяющий IEEE 829 и другие (например, BS 7925-2) 59.
* Структура документации
  + В IEEE 829 используется понятие «Master Test Plan» и «Level Test Plan», а также жесткие шаблоны для тест-кейсов и отчетов 1.
  + ISO/IEC/IEEE 29119-3 вводит более гибкие шаблоны, включая «Политику тестирования», «Стратегию тестирования», разделение на проектный и релизный тест-планы, а также отчеты о средах тестирования 19.
* Подход к процессам
  + IEEE 829 не регулирует процессы тестирования, только документирование.
  + ISO/IEC/IEEE 29119-2 определяет процессы управления тестированием на организационном, проектном и операционном уровнях, включая риск-ориентированный подход 59.
* Критика
  + ISO 29119 критикуют за избыточную документацию, несовместимость с Agile и игнорирование контекстно-ориентированного тестирования. Эксперты вроде Джеймса Баха отмечают, что стандарт не отражает практики сообщества тестировщиков
* **Как стандарт ISO/IEC 25010 определяет качество программного обеспечения?**

Стандарт ISO/IEC 25010 (модель SQuaRE) определяет качество ПО через 8 характеристик и их подхарактеристики:

1. Функциональная пригодность
   * Полнота, корректность, соответствие стандартам.
2. Надежность
   * Отказоустойчивость, восстановление после сбоев.
3. Производительность
   * Время отклика, использование ресурсов.
4. Совместимость
   * Взаимодействие с другими системами.
5. Удобство использования
   * Интуитивность интерфейса, доступность.
6. Безопасность
   * Конфиденциальность, целостность данных.
7. Удобство сопровождения
   * Модульность, тестируемость.
8. Портативность
   * Адаптируемость к разным средам 311.

Особенности стандарта 11:

* Расширяет предыдущую версию ISO 9126, добавляя характеристики безопасности и совместимости.
* Разделяет качество на качество продукта (внутренние атрибуты) и качество в использовании (опыт пользователя).
* Интегрируется с ISO 12207 и ISO 15288 для управления жизненным циклом.
* **Какие стандарты регламентируют документирование ПО?**

Международные стандарты:

* ISO/IEC/IEEE 29119-3  
  Определяет шаблоны тестовой документации (тест-планы, отчеты) 913.
* IEEE 1063  
  Требования к пользовательской документации (структура, содержание) 4.
* ISO/IEC 26514  
  Руководство по созданию пользовательской документации, включая чек-листы и примеры 4.

Российские стандарты (ГОСТ):

* ГОСТ 19.хх (ЕСПД)  
  Регламентирует виды документов (техническое задание, пояснительная записка, руководство оператора) и этапы их разработки 12.
* ГОСТ Р ИСО/МЭК 15910-2002  
  Описывает процесс создания пользовательской документации, включая стиль и структуру 48.
* ГОСТ Р ИСО/МЭК 12207—2010  
  Управление документацией в рамках процессов жизненного цикла ПО 8.

Другие ключевые стандарты:

* ISO/IEC 18019  
  Рекомендации по проектированию документации для прикладного ПО 4.
* BS 7925-1  
  Терминология в тестировании, частично включенная в ISO 29119 1.

Проблемы:

* Некоторые стандарты (например, ISO 26514) платные, что ограничивает их доступность 4.
* ГОСТ 19.хх устарел, но до сих пор применяется в РФ из-за отсутствия современных аналогов

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

**Архитектура и проектирование**

* **Какие стандарты применяются для проектирования архитектуры ПО?**

Основные стандарты и методологии для проектирования архитектуры ПО:

1. IEEE 1471 / ISO/IEC 42010 : Регламентирует описание архитектуры систем через представления, точки зрения и обоснования.
2. TOGAF (The Open Group Architecture Framework) : Методология разработки корпоративных архитектур, не стандарт, но широко используется.
3. Zachman Framework : Структурирует описание архитектуры через таблицу «5W+1H» (Who, What, When, Where, Why, How).
4. FEA (Federal Enterprise Architecture) : Используется в государственных системах США.
5. UML (Unified Modeling Language) : Стандарт моделирования ПО (см. ответ на вопрос 3).
6. SysML (Systems Modeling Language) : Расширение UML для системной инженерии.

* **Как IEEE 1471 (ISO/IEC 42010) описывает архитектурное описание систем?**

IEEE 1471 (переименован в ISO/IEC 42010) фокусируется на описании архитектуры , а не на её проектировании. Основные элементы:

1. Архитектурное описание (Architecture Description, AD) : Документ или набор документов, содержащих информацию об архитектуре.
2. Представления (Views) : Разные перспективы системы (например, логическая, физическая, процессная).
3. Точки зрения (Viewpoints) : Рамки для создания представлений, учитывающие потребности стейкхолдеров (например, безопасность, производительность).
4. Модели (Models) : Формализованные описания элементов системы (диаграммы, схемы).
5. Связь с заинтересованными сторонами (Stakeholders) : Учет их требований и ограничений.

Пример: Для разработчиков создаётся техническое представление с диаграммами компонентов, для менеджеров — высокоуровневая бизнес-архитектура.

* **Какие стандарты регламентируют UML-моделирование?**

UML (Unified Modeling Language) :

Стандарт OMG (Object Management Group) , последняя версия — UML 2.5.1 (2022).

Определяет нотацию и семантику для 14 типов диаграмм (например, диаграммы классов, последовательности, состояний).

SysML (Systems Modeling Language) :

Расширение UML для системной инженерии, также стандартизировано OMG.

MOF (Meta-Object Facility) : Метамодель для определения UML и других языков моделирования.

CWM (Common Warehouse Metamodel) : Для моделирования хранилищ данных.

* В **чем суть стандарта IEEE 1016 для описания проектных решений?**

IEEE 1016 (полный заголовок: IEEE Standard for Information Technology — Systems Design Description ) регламентирует документирование архитектурных и проектных решений . Основные аспекты:

* Структура документа :
  + Идентификация элементов системы.
  + Описание компонентов, их интерфейсов и взаимодействия.
  + Обоснование принятых решений.
  + Ссылки на требования и стандарты.

Цель : Обеспечить ясность и воспроизводимость проектных решений для разработчиков, тестировщиков и технических писателей.

Применение : Используется в критических системах (например, авиация, медицина), где важна документированность архитектуры.

* **Какие существуют стандарты для проектирования API (REST, GraphQL, gRPC)?**

REST (Representational State Transfer) :

Не имеет формального стандарта, но регулируется RFC 7231 (HTTP/1.1 Semantics and Content).

OpenAPI Specification (OAS) : Стандарт описания REST API (ранее Swagger).

GraphQL :

Спецификация от Facebook, поддерживается GraphQL Foundation .

Используется для гибкого запроса данных.

gRPC :

Базируется на Protocol Buffers (Protobuf) , стандартизирован Google.

Использует HTTP/2 для транспорта (стандарт RFC 7540 ).

JSON API : Стандарт обмена данными через JSON (jsonapi.org).

AsyncAPI : Для описания асинхронных API (например, мессенджеров).

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Безопасность и защита данных**

* **Как стандарт OWASP влияет на разработку безопасных приложений?**

OWASP (Open Web Application Security Project) — это некоммерческая организация, которая разрабатывает рекомендации и инструменты для повышения безопасности приложений. Основные направления влияния:

1. OWASP Top 10 : Список самых критичных уязвимостей веб-приложений (например, внедрение кода, утечка данных, CSRF). Разработчики используют его для приоритезации мер безопасности.
2. Руководства и методологии :
   * OWASP ASVS (Application Security Verification Standard) : Стандарт проверки безопасности приложений. Определяет уровни проверки (например, базовый, продвинутый) и требования к тестированию.
   * OWASP SAMM (Software Assurance Maturity Model) : Методология оценки зрелости процессов безопасности в SDLC (жизненном цикле разработки ПО).
3. Инструменты :
   * OWASP ZAP : Открытый инструмент для автоматического тестирования на проникновение.
   * Dependency-Check : Анализ уязвимостей в зависимостях.
4. Обучение : Курсы и документация для разработчиков, тестировщиков и архитекторов.

Пример влияния : При разработке веб-приложения команды следуют OWASP Top 10, чтобы избежать SQL-инъекций, внедряя параметризованные запросы и валидацию входных данных.

* **Какие требования к безопасности ПО предъявляет ISO/IEC 27001?**

ISO/IEC 27001 — международный стандарт по управлению информационной безопасностью (ISMS). Хотя он не специфичен для ПО, его требования касаются разработки:

* Управление рисками :
  + Проведение оценки рисков (risk assessment) для информационных систем.
  + Реализация мер для снижения рисков до допустимого уровня.
* Контрольные меры (Annex A) :
  + A.14. Управление безопасностью систем и приложений :
    - Защита от злонамеренного кода (антивирусы, анализ кода).
    - Безопасная разработка (проверка кода, тестирование на проникновение).
    - Управление изменениями (ревью кода, версионный контроль).
    - Конфигурационное управление (отслеживание изменений в системе).
* Политики и процедуры :
  + Документирование политик безопасности, процедур резервного копирования, обработки инцидентов.
* Сертификация : Организации, разрабатывающие ПО, могут получить сертификат ISO 27001, доказывая соответствие требованиям.

Пример : При разработке корпоративного ПО команда проводит регулярное тестирование на уязвимости (например, с помощью SAST/DAST) и внедряет шифрование данных в соответствии с политиками ISMS.

* **Как GDPR влияет на разработку приложений с персональными данными?**

GDPR (General Data Protection Regulation) — европейский регламент по защите персональных данных. Основные требования для разработки ПО:

* Принципы обработки данных :
  + Законность, справедливость, прозрачность : Пользователи должны знать, как собираются и используются их данные.
  + Целевое ограничение : Сбор только необходимых данных.
  + Минимизация данных : Хранение минимального объема информации.
  + Срок хранения : Удаление данных после достижения цели («право на забвение»).
* Технические меры :
  + Шифрование и псевдонимизация данных.
  + Анонимизация данных для аналитики.
  + Контроль доступа (RBAC, MFA).
* Обязанности разработчиков :
  + DPIA (Data Protection Impact Assessment) : Оценка рисков для конфиденциальности при высоком риске (например, обработка медицинских данных).
  + Уведомление о нарушении : Информирование регулятора в течение 72 часов при утечке данных.
  + Поддержка прав субъектов данных : Реализация функций для удаления, экспорта или исправления данных.
* Штрафы : До 4% годового оборота компании или €20 млн за нарушения.

Пример : Приложение должно запрашивать явное согласие пользователя на сбор данных и предоставлять интерфейс для удаления аккаунта с очисткой всех персональных записей.

* **Какие стандарты регулируют криптографическую защиту данных в ПО?**

Основные стандарты и организации:

* NIST (National Institute of Standards and Technology) :
  + FIPS 140-2/3 : Требования к криптографическим модулям (например, для аппаратных и программных решений).
  + SP 800-56A/B/C : Рекомендации по обмену ключами (Diffie-Hellman, ECDH).
  + AES (Advanced Encryption Standard) : Алгоритм шифрования (Rijndael), стандартизированный FIPS PUB 197.
  + SHA-2/SHA-3 : Алгоритмы хэширования.
* ISO/IEC 18033 : Стандарты шифрования (например, AES, RSA, ECDSA).
* PKCS (Public-Key Cryptography Standards) : Серия стандартов для работы с асимметричным шифрованием (например, PKCS #7 для подписи данных).
* TLS/SSL :
  + RFC 8446 (TLS 1.3) : Протокол шифрования для безопасных соединений.
  + X.509 : Стандарт для цифровых сертификатов.
* Российские стандарты :
  + ГОСТ 34.10-2012/34.11-2012 : Алгоритмы электронной подписи и хэширования.
  + КриптоПро : Реализация ГОСТ в коммерческих продуктах.

Пример : При разработке платежного приложения используется TLS 1.3 для шифрования трафика и AES-256 для хранения данных на сервере.

* **Как PCI DSS влияет на разработку платежных систем?**

PCI DSS (Payment Card Industry Data Security Standard) — стандарт безопасности для организаций, обрабатывающих данные платежных карт (Visa, MasterCard и др.). Основные требования:

* Защита данных карт :
  + Не хранить данные в открытом виде (например, PAN — Primary Account Number).
  + Использовать шифрование (AES, TLS) при передаче данных.
* Контроль доступа :
  + Разделение ролей (RBAC).
  + Аутентификация с двумя факторами (MFA) для администраторов.
* Мониторинг и тестирование :
  + Регулярное сканирование на уязвимости (например, с помощью Nessus).
  + Проведение пентестов не реже раза в год.
* Политики и процедуры :
  + Документирование политик безопасности.
  + Обучение сотрудников.
* Сертификация : Компании должны пройти аудит у Qualified Security Assessor (QSA) для соответствия уровню 1–4 (в зависимости от объема транзакций).

Пример : При разработке интернет-магазина необходимо использовать токенизацию для замены номеров карт на уникальные токены и внедрить 3D Secure для аутентификации пользователей.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Тестирование и качество**

* **Какие стандарты описывают процесс тестирования ПО?**

Основные стандарты, регламентирующие процесс тестирования ПО:

* IEEE 829-2008 (IEEE Standard for Software and System Test Documentation) :
  + Описывает структуру документации для тестирования, включая план тестирования, спецификации тестов, отчёты о результатах.
  + Пример: Документ *Test Plan* включает цели, стратегию, ресурсы, критерии выхода.
* IEEE 1008-1987 (IEEE Standard for Software Unit Testing) :
  + Фокусируется на unit-тестировании: методы покрытия кода (например, покрытие ветвлений), критерии завершения тестирования.
* ISO/IEC/IEEE 29119 (Software Testing Standards) :
  + Серия стандартов, охватывающих весь цикл тестирования: планирование, проектирование, выполнение, управление.
  + Включает рекомендации по автоматизации и тестированию Agile.
* ISTQB (International Software Testing Qualifications Board) :
  + Не формальный стандарт, но глобальный сертификационный фреймворк, описывающий лучшие практики тестирования (например, уровни тестирования, техники проектирования тест-кейсов)
* **Как TMMi (Test Maturity Model integrated) улучшает процесс тестирования?**

TMMi — это модель зрелости процессов тестирования, аналогичная CMMI, но сфокусированная на тестировании. Основные аспекты:

* Уровни зрелости (от 1 до 5):
  + Уровень 1 (Initial) : Процессы неформализованы, хаотичны.
  + Уровень 2 (Managed) : Тестирование планируется, отслеживается и завершается по критериям.
  + Уровень 3 (Defined) : Стандартизированные процессы (например, шаблоны тест-кейсов, методологии тестирования).
  + Уровень 4 (Measured) : Количественные метрики (например, покрытие требований, количество найденных багов).
  + Уровень 5 (Optimized) : Постоянное улучшение через анализ данных и внедрение инноваций.
* Ключевые практики :
  + Тестовое планирование : Определение стратегии, приоритетов и рисков.
  + Проектирование тестов : Использование методов вроде граничных значений, эквивалентных классов.
  + Управление дефектами : Систематизация и трекинг багов через инструменты (Jira, ALM).
  + Автоматизация : Интеграция автоматизированных тестов в CI/CD.
* Преимущества :
  + Снижение рисков ошибок в продакшене.
  + Повышение эффективности команды за счёт стандартизации.
* **Какие метрики качества ПО регламентирует ISO/IEC 9126?**

ISO/IEC 9126 (заменён на ISO/IEC 25010 ) описывает характеристики качества ПО и метрики для их оценки. Основные характеристики:

* Функциональность (Functionality) :
  + Подхарактеристики: функциональная полнота, корректность, совместимость.
  + Метрики: процент выполненных требований, количество ошибок в расчетах.
* Надёжность (Reliability) :
  + Подхарактеристики: готовность, устойчивость к сбоям, восстановление после сбоев.
  + Метрики: MTBF (среднее время безотказной работы), MTTR (время восстановления).
* Удобство использования (Usability) :
  + Подхарактеристики: понятность, обучаемость, привлекательность.
  + Метрики: время выполнения задачи, количество ошибок пользователей.
* Производительность (Performance) :
  + Подхарактеристики: время отклика, пропускная способность.
  + Метрики: задержки под нагрузкой, использование ресурсов CPU/памяти.
* Поддерживаемость (Maintainability) :
  + Подхарактеристики: модульность, повторное использование, простота анализа.
  + Метрики: сложность кода (Cyclomatic Complexity), процент комментариев.
* Переносимость (Portability) :
  + Подхарактеристики: адаптивность, совместимость с платформами.
  + Метрики: количество изменений для переноса на новую ОС.

Пример : Для веб-приложения измеряют MTBF (например, 1000 часов без сбоев) и время отклика (не более 2 секунд при нагрузке 1000 запросов/сек).

* **В чем разница между unit-тестированием и интеграционным тестированием по стандартам?**

|  |  |  |
| --- | --- | --- |
| **Цель** | **Проверка корректности отдельных модулей/классов.** | **Проверка взаимодействия между модулями/компонентами.** |
| **Уровень абстракции** | **Низкий (код, функции).** | **Средний/высокий (интерфейсы, API, базы данных).** |
| **Инструменты** | **JUnit (Java), NUnit (.NET), PyTest (Python).** | **Postman (API), Selenium (UI), SoapUI.** |
| **Стандарты** | **IEEE 1008 (покрытие кода, критерии завершения).** | **IEEE 829 (план тестирования, отчёты).** |
| **Типы ошибок** | **Логические ошибки в коде, некорректные вычисления.** | **Ошибки в интерфейсах, передаче данных, обработке исключений.** |
| **Автоматизация** | **Часто автоматизируется.** | **Автоматизируется в рамках CI/CD.** |

Пример : Unit-тест для функции sum(a, b) проверяет, что sum(2, 2) == 4. Интеграционный тест проверяет, что API /api/calculate возвращает правильный JSON при вызове POST /sum.

* **Как стандарты описывают автоматизацию тестирования?**

Стандарты, регламентирующие автоматизацию тестирования:

* IEEE 829-2008 :
  + Упоминает автоматизированные тесты в разделе *Test Design Specification* : описание скриптов, данных, ожидаемых результатов.
* ISO/IEC/IEEE 29119-6 (Test Automation Standards) :
  + Рекомендации по выбору инструментов, проектированию фреймворков (например, Page Object Model).
  + Управление тестовыми данными и отчётностью.
* ISTQB Advanced Level – Test Automation Engineer :
  + Рекомендации по этапам внедрения автоматизации: анализ требований, выбор инструментов, создание фреймворка, поддержка.
* Best Practices :
  + Выбор инструментов : Selenium (UI), JMeter (нагрузочное), REST Assured (API).
  + CI/CD интеграция : Запуск тестов в Jenkins, GitLab CI.
  + Метрики : Покрытие автоматизацией, время выполнения тестов, количество найденных регрессионных багов.

Пример : Внедрение автоматизированных UI-тестов с использованием Selenium WebDriver и отчётов Allure, интегрированных в Jenkins.

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

**DevOps и CI/CD**

* **Какие стандарты регулируют процессы непрерывной интеграции и доставки (CI/CD)?**

CI/CD (Continuous Integration/Continuous Delivery) не имеет единого формального стандарта, но ряд документов и практик регламентируют процессы:

* ISO/IEC 29110 :
  + Серия стандартов для малых и средних организаций (SME), включая процессы автоматизации сборки, тестирования и развертывания.
  + Модуль *Lifecycle Profiles for Very Small Entities* описывает базовые практики CI/CD.
* DevOps Maturity Model (DMM) :
  + Не стандарт, но популярный фреймворк (развитие TMMi), описывающий уровни зрелости CI/CD: от начального (ручные процессы) до оптимизированного (полная автоматизация и мониторинг).
* IEEE 829-2008 и ISO/IEC 29119 :
  + Описывают документирование тестирования, включая автоматизированные тесты в CI/CD.
* DevOps Institute (DOI) :
  + Обучает по программам, включая сертификации (например, Certified DevOps Engineer ), основываясь на практиках CI/CD.
* Cloud Native Computing Foundation (CNCF) :
  + Рекомендации для CI/CD в облачных средах (например, использование GitOps и ArgoCD).

Пример : Внедрение CI/CD-пайплайна с Jenkins или GitLab CI, где каждый коммит запускает автоматическую сборку, тестирование (unit и интеграционные тесты) и развертывание в staging-среду.

* **Как ITIL и DevOps взаимодействуют в управлении разработкой?**

ITIL (Information Technology Infrastructure Library) и DevOps дополняют друг друга, объединяя процессы эксплуатации и разработки:

* Общие цели :
  + Ускорение выпуска изменений с минимальными рисками.
  + Обеспечение стабильности и качества сервисов.
* Ключевые точки взаимодействия :
  + Change Management (ITIL) ↔ CI/CD (DevOps) :
    - ITIL регламентирует одобрение изменений, DevOps автоматизирует их внедрение через пайплайны.
  + Incident Management (ITIL) ↔ Monitoring/Alerting (DevOps) :
    - Инциденты из ITIL обрабатываются через автоматизированные алерты (например, Prometheus + Grafana).
  + Service Catalog (ITIL) ↔ Infrastructure as Code (IaC) :
    - Описание сервисов в ITIL реализуется через Terraform или Ansible.
* Преимущества интеграции :
  + Снижение времени восстановления (MTTR) за счёт автоматизации.
  + Соблюдение регуляторных требований (например, PCI DSS) через аудит изменений.

Пример : При запуске нового микросервиса через CI/CD, команда регистрирует его в ITIL-каталоге сервисов и настраивает мониторинг через DevOps-инструменты.

* **Какие стандарты описывают контейнеризацию (Docker, Kubernetes)?**

Контейнеризация не имеет формальных стандартов, но регулируется инициативами и спецификациями:

* Open Container Initiative (OCI) :
  + Image Format Specification : Формат образов контейнеров (OCI Image), поддерживается Docker, containerd, CRI-O.
  + Runtime Specification : Стандарт выполнения контейнеров (например, runc, Kata Containers).
* Cloud Native Computing Foundation (CNCF) :
  + Kubernetes (K8s) :
    - Де-факто стандарт оркестрации контейнеров. API Kubernetes стандартизирован CNCF.
    - Инструменты экосистемы: Helm (шаблоны развертывания), Istio (Service Mesh).
* Docker и Moby Project :
  + Docker использует спецификации OCI, но расширяет их (например, Dockerfile, Compose).
* Open Policy Agent (OPA) :
  + Регулирует политики безопасности контейнеров (например, ограничения на запуск привилегированных контейнеров).

Пример : Развертывание

* **Как ISO/IEC 27017 влияет на безопасность облачных приложений?**

ISO/IEC 27017 — это дополнение к ISO/IEC 27001, специфичное для облачных сервисов. Основные требования:

* Управление идентичностью и доступом :
  + Реализация MFA (многофакторной аутентификации) для облачных аккаунтов.
  + Разделение ролей (RBAC) в IaaS/PaaS/SaaS.
* Шифрование данных :
  + Шифрование данных в покое (AES-256) и в движении (TLS 1.2+).
  + Управление ключами (например, с помощью AWS KMS или HashiCorp Vault).
* Мониторинг и аудит :
  + Логирование всех действий в облаке (например, AWS CloudTrail).
  + Регулярные проверки конфигураций (например, с помощью AWS Config).
* Соответствие SLA :
  + Гарантия доступности сервисов (например, 99.9% uptime).
  + Резервное копирование и восстановление после инцидентов.
* Контроль изменений :
  + Автоматизация через Infrastructure as Code (IaC) с версионным контролем.
  + Проверка безопасности изменений (например, через SAST/DAST).

Пример : Приложение в AWS должно использовать IAM-роли для временных учетных данных и шифровать данные в S3-бакетах с помощью KMS.

* **Какие стандарты регламентируют управление конфигурациями (Ansible, Terraform)?**

Управление конфигурациями (Configuration Management) регулируется стандартами и фреймворками:

* ITIL 4 :
  + Процесс Service Configuration Management :
    - Управление CI/CD (Configuration Items) и их взаимосвязями.
    - Интеграция с DevOps через автоматизацию.
* COBIT (Control Objectives for Information and Related Technologies) :
  + Домен DSS05 (Manage IT Assets) : Управление конфигурациями как часть ИТ-активов.
* ISO/IEC 12207 :
  + Стандарт жизненного цикла ПО, включающий управление версиями и конфигурациями.
* Open Group TOGAF :
  + Архитектурное управление конфигурациями в рамках Enterprise Architecture.
* Infrastructure as Code (IaC) Best Practices :
  + Использование Terraform, Ansible, Puppet для описания инфраструктуры в коде.
  + Версионный контроль (Git), тестирование (InSpec, Kitchen), CI/CD-интеграция.

Пример : Развертывание инфраструктуры в AWS через Terraform с хранением состояния в S3-бакете и проверкой через AWS Config.

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

**Методологии**

* **Как Scrum, Kanban и SAFe соответствуют стандартам разработки?**

Эти методологии не являются формальными стандартами, но их практики могут соответствовать или дополнять существующие стандарты. Основные точки взаимодействия:

* Scrum :
  + ISO/IEC 29110 :
    - Серия стандартов для малых организаций, включает процессы планирования, тестирования и управления качеством, совместимые с итеративным подходом Scrum.
  + IEEE 12207 :
    - Стандарт жизненного цикла ПО. Scrum может быть адаптирован под его процессы (например, управление требованиями через Product Backlog).
* Kanban :
  + ISO/IEC 33000 (Process Assessment) :
    - Оценивает зрелость процессов. Kanban-практики (визуализация потока, ограничение WIP) соответствуют требованиям к управлению и оптимизации.
  + ITIL 4 :
    - Интеграция Kanban в управление изменениями и Incident Management для ускорения исправления ошибок.
* SAFe (Scaled Agile Framework) :
  + ISO/IEC 21827 (Information Security Framework) :
    - SAFe включает безопасность в DevOps-процессы (Secure DevOps), что соответствует требованиям к защите данных.
  + CMMI (Capability Maturity Model Integration) :
    - SAFe поддерживает уровни зрелости CMMI через регулярные ретроспективы, метрики и улучшение процессов.

Пример : Команда использует Scrum для итеративной разработки, а требования документируются в формате IEEE 830-1998 (спецификации Use Cases), адаптированные под Agile.

* **Какие стандарты поддерживают гибкую разработку (Agile)?**

Agile не регламентируется единым стандартом, но ряд документов и фреймворков способствуют его внедрению:

* Agile Manifesto (2001) :
  + Не стандарт, но основа для методологий (Scrum, XP, Lean). Принципы: итеративность, сотрудничество, гибкость.
* ISO/IEC 29110 :
  + Серия стандартов для малых команд, включает Agile-процессы: планирование спринтов, управление рисками, тестирование.
* IEEE 12207 :
  + Стандарт жизненного цикла ПО. Agile-процессы (например, CI/CD) могут быть встроены в его структуру (например, управление конфигурацией через Git).
* ISO/IEC 25010 :
  + Метрики качества ПО. Agile-команды используют их для оценки характеристик (например, надежности) в каждом спринте.
* DevOps Maturity Model (DMM) :
  + Фреймворк для оценки зрелости DevOps, включая Agile-практики (например, автоматизация тестирования).

Пример : Agile-команда использует ISO/IEC 29110 для структурирования процессов разработки, включая планирование спринтов и управление дефектами.

* **В чем разница между Agile и Waterfall с точки зрения стандартов?**

|  |  |  |
| --- | --- | --- |
| * Подход к процессам | Итеративный, итеративно-инкрементальный. | Линейный, последовательный. |
| Стандарты | ISO/IEC 29110, IEEE 12207 (адаптированный). | IEEE 12207, ISO/IEC 15504 (SPICE). |
| Управление требованиями | Требования эволюционируют (User Stories, Product Backlog). | Требования фиксируются на ранних этапах (IEEE 830-1998). |
| Тестирование | Тестирование в каждом спринте (CI/CD). | Тестирование после реализации (IEEE 829-2008). |
| Документация | Минимальная, создается по мере необходимости. | Подробная, документируется на каждом этапе. |
| Гибкость | Быстрая адаптация к изменениям. | Сложность внесения изменений после этапа проектирования. |
| Пример стандартов | Scrum Guide, SAFe. | IEEE 12207, ISO/IEC 12119. |

Пример : В Waterfall требования документируются в IEEE 830-1998 до начала разработки, а в Agile они корректируются в Product Backlog на основе обратной связи клиентов.

* **Как ISO/IEC 26515 регулирует документацию в Agile?**

ISO/IEC 26515 — стандарт для документации пользовательского контента (руководства, справка, онлайн-помощь). Его применение в Agile:

* Итеративное создание документации :
  + Документация разрабатывается параллельно с функциональностью (например, обновление руководства пользователя в каждом спринте).
* Минимизация избыточности :
  + Акцент на актуальность и удобство использования, а не на объем.
* Интеграция с CI/CD :
  + Автоматическая генерация документации (например, через Swagger для API) в пайплайнах.
* Учет пользовательского опыта :
  + Проверка документации через юзабилити-тестирование в конце спринта.

Пример : При разработке мобильного приложения в Agile-команде, руководство пользователя обновляется после каждого спринта, а сложные разделы проверяются с помощью A/B-тестирования.

* **Какие стандарты описывают управление требованиями в Agile?**

Управление требованиями в Agile регулируется как Agile-фреймворками, так и общими стандартами:

* Agile-фреймворки :
  + Scrum :
    - Требования хранятся в Product Backlog и уточняются на Sprint Planning.
  + SAFe :
    - Использует Agile Release Train (ART) для согласования требований между командами.
  + XP (Extreme Programming) :
    - User Stories и Acceptance Tests как основа для требований.
* Общие стандарты :
  + IEEE 830-1998 (Recommended Practice for Software Requirements Specifications) :
    - Может быть адаптирован для Agile: User Stories как спецификации требований.
  + ISO/IEC/IEEE 29148 :
    - Стандарт по анализу и управлению требованиями. В Agile требования уточняются через регулярные встречи с заказчиком.
  + INCOSE Systems Engineering Handbook :
    - Управление требованиями в сложных системах, совместимое с Agile через итеративное уточнение.
* Инструменты :
  + Jira, Azure DevOps, Trello: для управления требованиями в формате User Stories и Epics.

Пример : В Agile-проекте требования описываются в виде User Stories в Jira, а Acceptance Criteria проверяются через автоматизированные тесты (например, Cucumber).

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Специфичные технологии**

* **Какие стандарты применяются для разработки мобильных приложений?**

Мобильная разработка регулируется как общими стандартами для ПО, так и специфичными для мобильных платформ:

* Общие стандарты качества :
  + ISO/IEC 25010 : Характеристики качества ПО (функциональность, надёжность, удобство использования, производительность).
  + IEEE 830-1998 : Управление требованиями (например, User Stories для мобильных функций).
* Безопасность :
  + OWASP Mobile Top 10 : Список уязвимостей (например, небезопасное хранение данных, слабая аутентификация).
  + OWASP MASVS (Mobile Application Security Verification Standard) : Требования к тестированию мобильных приложений.
* Соответствие законодательству :
  + GDPR : Защита персональных данных (например, шифрование локального хранилища).
  + COPPA (Child Online Privacy Protection Act) : Ограничения на сбор данных у несовершеннолетних.
* Платформенные стандарты :
  + Android Design Guidelines : Рекомендации Google по интерфейсам (Material Design).
  + Apple Human Interface Guidelines : Требования к UX для iOS (например, поддержка Dynamic Type).
* Тестирование :
  + ISO/IEC 29119 : Автоматизация тестирования (например, Espresso для Android, XCTest для iOS).

Пример : При разработке банковского приложения реализуются OWASP MASVS (шифрование данных в localStorage) и соблюдается GDPR (анонимизация пользовательской активности).

* **Как стандарты регулируют веб-разработку (W3C, WCAG)?**

Веб-разработка основывается на стандартах, утверждённых W3C (World Wide Web Consortium) и другими организациями:

* Стандарты языков разметки и стилей :
  + HTML5 : Стандарт структуры веб-страниц (W3C Recommendation).
  + CSS3 : Стандарт оформления (W3C Candidate Recommendation).
  + ECMAScript (ECMA-262) : Стандарт JavaScript (поддерживается TC39).
* Доступность :
  + WCAG 2.1/2.2 (Web Content Accessibility Guidelines) : Требования для людей с ограниченными возможностями (например, контраст текста, поддержка screen reader).
  + WAI-ARIA (Accessible Rich Internet Applications) : Стандарт описания интерфейсных элементов для assistive technologies.
* Производительность и безопасность :
  + Lighthouse : Инструмент Google, реализующий рекомендации W3C по оптимизации (например, сжатие изображений, минификация JS).
  + CSP (Content Security Policy) : Защита от XSS (стандарт W3C).
  + CORS (Cross-Origin Resource Sharing) : Контроль доступа между доменами (RFC 6454).
* SEO и семантическая вёрстка :
  + Schema.org : Стандарт структурированных данных для поисковых систем.
  + Open Graph Protocol : Описание метаданных для социальных сетей.

Пример : Веб-сайт соответствует WCAG 2.1 (контраст текста ≥ 4.5:1) и использует Schema.org для улучшения отображения в результатах поиска.

* **Какие стандарты описывают разработку встроенных систем (Embedded)?**

Встроенные системы (Embedded Systems) регулируются стандартами, учитывающими надёжность, безопасность и функциональность:

* Функциональная безопасность :
  + IEC 61508 : Базовый стандарт для электронных систем управления (например, в промышленности). Определяет уровни SIL (Safety Integrity Level).
  + ISO 26262 : Адаптация IEC 61508 для автомобильной промышленности (например, управление рисками в системах ABS).
  + IEC 60730 : Требования к автоматическим контроллерам бытовой электроники.
* Стандарты программирования :
  + MISRA C/C++ : Правила безопасного кодирования для встроенных систем (например, запрет на использование указателей без проверки).
  + AUTOSAR (AUTomotive Open System ARchitecture) : Стандарт архитектуры ПО для автомобильной промышленности.
* Тестирование и верификация :
  + DO-178C : Стандарт сертификации авиационного ПО (например, бортовые компьютеры).
  + IEC 61513 : Требования к инструментам тестирования ядерных систем.
* Управление жизненным циклом :
  + IEEE 12207 : Процессы разработки, включая управление конфигурацией и версионность.

Пример : Разработка ПО для автомобиля по ISO 26262 включает анализ рисков (Hazard Analysis) и тестирование по DO-178C (авиационный аналог).

* **Как IEC 62304 регулирует ПО для медицинских устройств?**

IEC 62304 — ключевой стандарт для разработки медицинского ПО, обеспечивающий безопасность и соответствие требованиям:

* Жизненный цикл разработки :
  + Требует формализованных процессов: планирование, анализ требований, проектирование, реализация, тестирование, сопровождение.
  + Включает управление конфигурацией и версионность (например, Git с политикой Code Review).
* Классификация рисков :
  + Уровни классификации: A (нет риска повреждения пациента), B (возможны легкие повреждения), C (возможны тяжелые повреждения или смерть).
* Тестирование и верификация :
  + Unit- и интеграционное тестирование с покрытием кода (например, покрытие ветвлений ≥ 90%).
  + Тестирование на отказоустойчивость (например, обработка ошибок датчиков).
* Документирование :
  + Требуется подробная документация: план тестирования, отчёты о дефектах, руководства пользователя.
  + Используется IEEE 830-1998 для описания требований.
* Сертификация :
  + Соответствие стандарту требуется для получения разрешений FDA (США), CE Mark (ЕС).

Пример : Разработка ПО для кардиостимулятора по IEC 62304 включает тестирование по уровню классификации C, с анализом всех сценариев отказа.

* **Какие стандарты применяются в разработке игр?**

Игровая индустрия использует общие стандарты ПО, а также специфичные для игровой среды:

* Технические стандарты :
  + ISO/IEC 25010 : Качество ПО (например, производительность в реальном времени, надёжность).
  + OpenGL/DirectX : Стандарты графического API для рендеринга.
* Безопасность и защита данных :
  + GDPR/CCPA : Обработка данных игроков (например, сохранение прогресса в облаке).
  + Anti-Cheat-системы : Стандарты защиты от читов (например, Easy Anti-Cheat, BattlEye).
* Доступность и локализация :
  + WCAG 2.1 : Для доступности интерфейсов (например, поддержка цветовой слепоты).
  + Unicode (ISO/IEC 10646) : Поддержка множества языков.
* Юридические стандарты :
  + ESRB/PEGI : Классификация возрастных ограничений (например, «18+» для насилия).
  + IARC (International Age Rating Coalition) : Автоматическая оценка контента для цифровых платформ.
* Инструменты и движки :
  + Unity/HDRP : Стандарты использования движка (например, оптимизация под PS5/Xbox Series X).
  + Unreal Engine : Стандарты создания шейдеров и физики (PhysX).

Пример : Разработка игры для консолей требует соответствия HDRP-стандартам Unity и сертификации ESRB (например, проверка контента на насилие).