# Самостоятельная работа по теме: Анализ спецификаций, верификация и аттестация программного обеспечения

**Выполнил студент 319 группы: Меркушин Евгений**

**Задания:**

### **1. Теоретическая часть:**

**Спецификация требований программного обеспечения** — структурированный набор требований/запросов (функциональность, производительность, конструктивные ограничения и атрибуты) к программному обеспечению и его внешним интерфейсам. Предназначен для того, чтобы установить базу для соглашения между заказчиком и разработчиком (или подрядчиками) о том, как должен функционировать программный продукт.

**Верификация** — **процесс проверки и подтверждения достоверности чего-либо с использованием различных методов**.

**Аттестация** — это периодическая проверка сотрудника на соответствие занимаемой должности.

**Спецификация** — это описание разрабатываемой/разработанной части системы. Документ создают на этапе проработки задач, чтобы адаптировать требования бизнеса под системный язык и передать команде разработки.

**Моделирование в процессе анализа спецификаций** — это **способ познания процессов разработки программного обеспечения с использованием объекта-модели**.

Верификация фокусируется на «*Делаем ли мы продукт правильно?* Она включает в себя следующие действия:

* Анализ требований
* Использование статических методов анализа кода
* Контроль проекта и процессов разработки
* Проведение код-ревью

Пример верификации:  
Проверка того, что требования к программному обеспечению ясны, полны и не противоречивы.

Все средства измерений с целью обеспечения единства измерений подвергаются оценке соответствия требованиям, установленным при утверждении их типа, если они используются в сфере действия государственного метрологического контроля и надзора. ПО, как правило, не аттестуется отдельно, а проходит аттестацию в составе СИ, т.е. в процессе аттестации не учитываются характеристики надежности и погрешности самого ПО.

Учитывая все возрастающую надежность механических, электрических и электронных компонентов средств измерений можно сделать вывод, что качество измерений все в большей степени определяется правильной работой программного обеспечения.

### **2. Практическая часть:**

Для улучшения требований можно сделать их более конкретными и измеримыми, а также учесть дополнительные аспекты, которые могут быть важны при реализации системы.

Вот как можно улучшить требования:

# Требование 1: «Система должна обеспечивать безопасность пользовательских данных.»

* **Что добавить:** уточнить, какие именно меры безопасности должны быть реализованы (например, шифрование данных, аутентификация пользователей, защита от несанкционированного доступа).
* **Изменить:** сделать требование более конкретным и детализированным.

**Пример улучшенного требования:** «Система должна обеспечить защиту пользовательских данных с использованием SSL/TLS для передачи данных по сети, шифрования хранимых данных с помощью алгоритма AES-256 и двухфакторной аутентификации для входа в систему.»

# Требование 2: «Приложение должно загружаться быстро.»

* **Что добавить:** указать конкретные временные рамки или критерии скорости загрузки.
* **Изменить:** добавить количественные показатели для оценки выполнения этого требования.

**Пример улучшенного требования:** «Время загрузки приложения не должно превышать 5 секунд при подключении к интернету со скоростью 10 Мбит/с.»

# Требование 3: «Пользователь должен иметь возможность восстановить доступ к учетной записи через электронную почту.»

* **Что добавить:** описать процесс восстановления доступа, включая шаги, которые должен пройти пользователь.
* **Изменить:** разделить требование на несколько подтребований для большей ясности.

**Примеры улучшенных требований:**

* «При утере пароля пользователь должен иметь возможность запросить восстановление доступа через ссылку, отправленную на зарегистрированный адрес электронной почты.»
* «Ссылка для восстановления пароля должна быть действительна в течение 24 часов после отправки.»
* «После успешного ввода нового пароля старый пароль должен стать недействительным.»

# Процедура верификации функции "Восстановление пароля" в веб-приложении

# Этапы процедуры верификации:

1. **Запрос на восстановление пароля:**
   * Пользователь вводит свой email-адрес в форму восстановления пароля.
   * Проверяется наличие введенного email-адреса в базе данных пользователей.

**Возможные ошибки:**

* + Email-адрес может быть введен неверно (опечатки, неправильный формат).
  + Адрес может отсутствовать в базе данных.

**Как избежать ошибок:**

* + Реализовать проверку формата email-адреса на стороне клиента (валидацию) до отправки запроса на сервер.
  + Если email отсутствует в базе, выводить сообщение об ошибке без указания конкретной причины для защиты от перебора возможных адресов злоумышленниками.

1. **Отправка ссылки для восстановления пароля:**
   * На указанный email отправляется письмо с уникальной ссылкой для сброса пароля, которая действует ограниченное время (например, 30 минут).
   * Ссылка содержит токен, который связывается с пользователем в базе данных и имеет срок действия.

**Возможные ошибки:**

* + Письмо может попасть в спам.
  + Токен может быть скомпрометирован.

**Как избежать ошибок:**

* + Использовать надежные сервисы рассылки писем, чтобы минимизировать вероятность попадания письма в спам-папку.
  + Генерировать токены с использованием криптографически стойких алгоритмов (например, JWT с подписью).
  + Ограничивать количество попыток использования одной ссылки для предотвращения атак методом подбора.

1. **Переход по ссылке и ввод нового пароля:**
   * После перехода по ссылке пользователь попадает на страницу ввода нового пароля.
   * Новый пароль должен соответствовать требованиям сложности (минимальная длина, использование специальных символов и т.д.).

**Возможные ошибки:**

* + Пароль может не соответствовать установленным правилам.
  + Возможно повторное использование старого пароля.

**Как избежать ошибок:**

* + Ввести строгие правила проверки паролей на стороне сервера (длина, сложность, отсутствие старых паролей).
  + Показывать подсказки о том, какой пароль считается безопасным.

1. **Сохранение нового пароля:**
   * Старый пароль пользователя заменяется новым в базе данных, новый пароль хранится в зашифрованном виде (с использованием хеширования с солью).
   * Пользователю выводится уведомление о успешном восстановлении пароля и предлагается войти под новыми данными.

**Возможные ошибки:**

* + Проблемы с сохранением нового пароля в базе.
  + Некорректная обработка ошибок при сохранении.

**Как избежать ошибок:**

* + Обеспечить надежное шифрование паролей (используя современные алгоритмы, такие как bcrypt или Argon2).
  + Логирование всех операций с базой данных для отслеживания возможных проблем.

1. **Тестирование логина с новым паролем:**
   * Провести тестовый вход с новым паролем, чтобы убедиться, что процедура завершена успешно.

**Ошибки:**

* + Возможны проблемы с авторизацией после смены пароля (некорректное хранение или проверка пароля).

**Избежать ошибок:**

* + Убедиться, что система корректно сравнивает хэш нового пароля с тем, что хранится в базе, используя тот же алгоритм хэширования.

# Оценка требований к аттестации медицинского приложения

# Основные требования к безопасности и надежности:

1. **Конфиденциальность данных:**
   * Данные пациентов должны быть защищены от несанкционированного доступа.
   * Использование шифрования данных как при передаче (SSL/TLS), так и при хранении (AES-256 или аналогичные стандарты).
   * Аутентификация пользователей с использованием многофакторной аутентификации (MFA).
2. **Целостность данных:**
   * Все изменения данных должны фиксироваться в журнале аудита с указанием времени, автора изменений и характера изменений.
   * Защита от несанкционированных модификаций данных (цифровые подписи, контроль версий).
3. **Доступность данных:**
   * Система должна быть доступна пользователям в любое время, с минимальными простоями.
   * Резервное копирование данных и механизмы быстрого восстановления после сбоев.
4. **Соответствие нормативным актам:**
   * Соблюдение стандартов обработки персональных данных (например, GDPR в Европе, ФЗ-152 в России).
   * Соответствие медицинским стандартам (HIPAA в США, Приказ Минздрава РФ №344н в России и др.).
5. **Защита от внешних угроз:**
   * Регулярное обновление программного обеспечения и патчей безопасности.
   * Проактивная защита от DDoS-атак, SQL-инъекций, XSS-атак и других видов кибератак.
   * Наличие межсетевых экранов и систем обнаружения вторжений (IDS/IPS).
6. **Обучение персонала:**
   * Персонал компании должен проходить регулярные тренинги по информационной безопасности и защите данных.
   * Политики и процедуры управления инцидентами должны быть документированы и доступны всем сотрудникам.
7. **Аудит и мониторинг:**
   * Постоянный аудит безопасности системы, включая тестирование на проникновение (penetration testing).
   * Мониторинг активности пользователей и журналов событий для раннего выявления подозрительных действий.

# Документация и данные для успешной аттестации:

1. **Техническая документация:**
   * Описание архитектуры системы и используемых технологий.
   * Руководства по эксплуатации и обслуживанию системы.
   * Инструкции по восстановлению данных и управлению инцидентами.
2. **Политика безопасности:**
   * Внутренние политики компании по обработке и защите персональных данных.
   * Правила доступа к системе и разграничения прав пользователей.
3. **Результаты тестирования:**
   * Отчеты о проведении тестов на проникновение и нагрузочных тестов.
   * Результаты аудитов безопасности и анализа уязвимостей.
4. **Лицензии и сертификаты:**
   * Лицензии на используемое программное обеспечение.
   * Сертификаты соответствия международным и национальным стандартам (ISO 27001, HIPAA и др.)
5. **Журналы аудита:**
   * Журналы событий и изменений данных за определенный период.
   * Записи о проведенных обновлениях и исправлениях.
6. **Контракты с поставщиками услуг:**
   * Договоры с облачными провайдерами, хостерами и другими внешними сервисами.
   * Соглашения о конфиденциальности и неразглашении информации (NDA).
7. **Планы аварийного восстановления:**
   * Подробные планы действий в случае чрезвычайных ситуаций (аварии, утечки данных и т.п.).
   * Механизмы резервного копирования и восстановления данных.

Эти документы и данные помогут продемонстрировать соответствие системы требованиям безопасности и надежности, необходимым для успешного прохождения аттестации.

### **3. Аналитическая часть:**

# Разница между анализом спецификаций и верификацией

**Анализ спецификаций** – это процесс проверки требований к программному обеспечению на предмет полноты, непротиворечивости и точности. Он проводится на ранних этапах разработки, еще до написания кода. Цель анализа спецификаций – убедиться, что все требования четко определены и понятны команде разработчиков, а также выявить любые потенциальные пробелы или противоречия в документации.

**Верификация**, напротив, выполняется уже после создания программного продукта и направлена на то, чтобы проверить, соответствует ли разработанное ПО исходным требованиям и спецификациям. Это включает в себя различные виды тестирования (функциональное, интеграционное, регрессионное и т.д.), целью которых является подтверждение правильности работы программы.

# Ключевые различия:

* **Этап выполнения**: Анализ спецификаций происходит до разработки, тогда как верификация – после.
* **Цель**: Анализ направлен на улучшение качества требований, а верификация – на проверку соответствия готового продукта этим требованиям.
* **Методы**: Анализ требует тщательного изучения документов и обсуждения с заинтересованными сторонами, тогда как верификация включает в себя практические тесты и эксперименты с программой.

# Этап жизненного цикла ПО и важность разделения процессов

**Этапы выполнения:**

* **Анализ спецификаций**: выполняется на стадии планирования и проектирования, до начала кодирования.
* **Верификация**: начинается после завершения разработки отдельных модулей или всего продукта, обычно во время фазы тестирования.

**Почему важно разделять:**

* Разделение этих процессов позволяет сосредоточиться на разных аспектах качества ПО. Анализ помогает предотвратить ошибки на ранней стадии, а верификация подтверждает правильность реализации.
* Раннее выявление недостатков в спецификациях экономит ресурсы и время, поскольку внесение изменений на поздних стадиях разработки обходится дороже.
* Четкое разделение ответственности между аналитиками и тестировщиками способствует повышению эффективности каждого процесса.

# Риски недостаточной аттестации

Недостаточная аттестация может привести к серьезным последствиям, особенно в критичных областях, таких как медицина и авиация, где надежность и безопасность имеют первостепенное значение.

**Примеры рисков:**

* В медицинском приложении недостаточная аттестация может привести к ошибкам в диагностике или лечении пациентов, что ставит под угрозу их здоровье и жизнь.
* В авиационном ПО неполная аттестация может спровоцировать отказы оборудования или неправильное управление самолетом, что чревато катастрофическими последствиями.

**Последствия для пользователей:**

* Потеря доверия к продукту и компании-разработчику.
* Финансовые убытки из-за необходимости устранения дефектов и компенсаций пострадавшим.
* Юридические последствия, включая штрафы и судебные иски.

# Влияние на затраты

Регулярное выполнение верификации и аттестации может существенно снизить общие затраты на разработку ПО благодаря предотвращению дорогостоящих ошибок на поздних этапах проекта.

**Как влияет на затраты:**

* При своевременной верификации дефекты выявляются раньше, что снижает стоимость их исправления.
* Аттестация помогает удостовериться в соответствии продукта стандартам и требованиям, что уменьшает риск возврата продукта на доработку после его выпуска.

**Пример увеличения затрат при пренебрежении:**

* Если ошибка обнаруживается только после запуска продукта, ее устранение потребует значительных ресурсов (переписывание кода, повторное тестирование, обновление документации).
* Дополнительные расходы на поддержку и исправление багов после релиза могут значительно превысить первоначальные затраты на качественную верификацию и аттестацию.

# Анализ времени на тестирование

Грамотное планирование времени на анализ спецификаций и верификацию может существенно сократить сроки разработки, так как позволяет заранее устранить многие проблемы.

**Причины сокращения сроков:**

* Тщательный анализ спецификаций предотвращает недоразумения и недопонимания между командой, что ускоряет процесс разработки.
* Качественное тестирование на ранних этапах позволяет своевременно обнаружить и исправить баги, избегая длительных циклов доработок.

**Важность учета времени:**

* Закладывание достаточного времени на эти процессы с самого начала проекта обеспечивает более плавную работу команды и снижает стресс от сжатых сроков.
* Игнорирование важности анализа и верификации часто приводит к тому, что разработчики вынуждены работать в авральном режиме, что негативно сказывается на качестве конечного продукта.

# Роль внешних аудитов

Внешние аудиты и независимая аттестация играют важную роль в повышении доверия к программному продукту, особенно если он предназначен для использования в критических областях.

**Преимущества привлечения сторонних компаний:**

* Независимость и объективность оценок.
* Глубокий опыт и знания в области сертификации и стандартизации.
* Возможность получить рекомендации по улучшению продукта и процессов разработки.

**Минусы:**

* Увеличенные финансовые затраты на привлечение внешних экспертов.
* Необходимость адаптации внутренних процессов к требованиям внешнего аудита.

Таким образом, проведение анализа спецификаций, верификации и независимой аттестации являются важными этапами жизненного цикла ПО, которые помогают гарантировать качество продукта и минимизировать риски, связанные с его эксплуатацией.

### **4. Заключение:**

Анализ спецификаций, верификация и аттестация являются ключевыми элементами в процессе создания качественного и безопасного программного обеспечения. Эти этапы позволяют обеспечить соответствие продукта заявленным требованиям, предотвратить появление ошибок на ранних стадиях разработки и подтвердить готовность продукта к использованию в реальных условиях.

Анализ спецификаций помогает уточнить и согласовать требования к ПО, устраняя неопределенности и противоречия, что сокращает риски возникновения ошибок на последующих этапах разработки. Верификация, проводимая после создания продукта, гарантирует, что программа функционирует согласно заданным спецификациям и стандартам. Аттестация, особенно важная в критических областях, таких как медицина и авиация, подтверждает соответствие продукта нормативным требованиям и стандартам безопасности.

Игнорирование этих этапов может привести к значительным финансовым потерям, юридическим последствиям и снижению доверия к продукту. Регулярное выполнение анализа спецификаций и верификации на протяжении всего жизненного цикла разработки помогает оптимизировать затраты и сократить сроки разработки, обеспечивая высокое качество конечного продукта.

Привлечение независимых внешних аудиторских компаний повышает уровень доверия к программному обеспечению, предоставляя объективную оценку его соответствия требованиям и нормам. Таким образом, комплексное применение анализа спецификаций, верификации и аттестации является залогом создания надежного, безопасного и конкурентоспособного программного продукта.