

Введение в разработку программного обеспечения (ИС и ЦД)

Технологии разработки ПО. Тестирование ПО. Документирование

План лекции:

- тестирование ПО: основные понятия и определения;
- классификация тестирования;
- виды тестирования;
- цели, задачи и принципы тестирования;
- ручное тестирование;
- примеры.
- проектная документация;
- состав и назначение документации;
- стандарты документирования.

1. Тестирование программного обеспечения



Эдсгер Вибе Дейкстра (11 мая 1930, — 6 августа 2002, Нидерланды) — нидерландский ученый.

Один из тех людей, с именем которых связано превращение программирования из шаманства в науку; один из разработчиков концепции структурного программирования, исследователь формальной верификации и распределенных вычислений. Тьюринговский лауреат (1972).

<https://habr.com/ru/post/303712/>

«Тестирование программ можно использовать для того, чтобы показать наличие ошибок, и никогда — для того чтобы показать их отсутствие!»

2. Тестирование программного обеспечения – основные понятия и определения

Тестирование программного обеспечения (Software Testing) –

проверка соответствия между реальным и ожидаемым поведением программы, осуществляемая на конечном наборе тестов, выбранном определенным образом.
[IEEE Guide to Software Engineering Body of Knowledge, SWEBOOK, 2004]

Верификация (Verification) – это процесс оценки системы или её компонентов с целью определения удовлетворяют ли результаты текущего этапа разработки условиям, сформированным в начале этого этапа *[IEEE]*. Т.е. выполняются ли наши цели, сроки, задачи по разработке проекта, определенные в начале текущей фазы.

Валидация (Validation) – это определение соответствия разрабатываемого ПО ожиданиям и потребностям пользователя, требованиям к системе [BS7925-1].

План Тестирования (Test Plan) – это документ, описывающий весь объем работ по тестированию, начиная с описания объекта, стратегии, расписания, критериев начала и окончания тестирования, до необходимого в процессе работы оборудования, специальных знаний, а также оценки рисков с вариантами их разрешения.

Тест дизайн (Test Design) – это этап процесса тестирования ПО, на котором проектируются и создаются тестовые случаи (тест кейсы), в соответствии с определёнными ранее критериями качества и целями тестирования.

Тестовый случай (Test Case) – это артефакт (побочный продукт, созданный в процессе тестирования ПО), описывающий совокупность шагов, конкретных условий и параметров, необходимых для проверки реализации тестируемой функции или её части.

Баг/Дефект Репорт (Bug Report) – это документ, описывающий ситуацию или последовательность действий, приведшую к некорректной работе объекта тестирования, с указанием причин и ожидаемого результата.

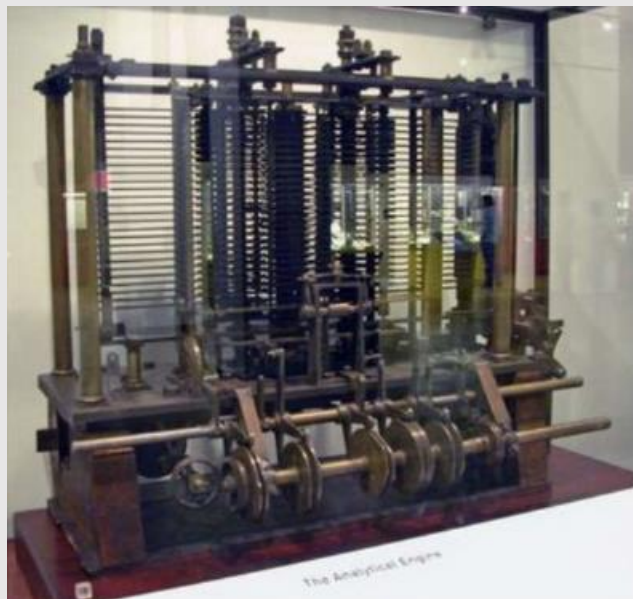
Тестовое Покрытие (Test Coverage) – это одна из метрик оценки качества тестирования, представляющая из себя плотность покрытия тестами требований либо исполняемого кода.

Детализация Тест Кейсов (Test Case Specification) – это уровень детализации описания тестовых шагов и требуемого результата, при котором обеспечивается разумное соотношение времени прохождения к тестовому покрытию

Время Прохождения Тест Кейса (Test Case Pass Time) – это время от начала прохождения шагов тест кейса до получения результата теста.

Понятие дефекта

Error, defect, failures, bug	
1843 г. – первое упоминание <i>ошибки</i> в аналитическом движении Чарльза Баббиджа.	1878 г. – Том Эдисон, первое слово « <i>bug</i> » в письме.



Menlo Park Mch 3rd 78
 Wm Oulton Esq
 Dear Sir
 You were partly correct, I did find a "bug" in my apparatus, but it was not in the telephone proper. It was of the genus "callbellum". The insect appears to find conditions for its existence in all call apparatus of telephones. Another delay was the sickness of Adam's wife. I intend to present you with a first class phonograph for your home, for reproducing music etc. this apparatus will run with a clockwork train. I will also place one in the room called "experimental room" if you will be so kind as to inform me where that is. I wish you could find time some afternoon to come down and see my experimental room (no desks manned with mathematicians) and hear some good phonograph singing and talking.
 Yours Truly
 J. A. Edison

Error (ошибка) – логическая или другая ошибка, которая может привести к возникновению дефекта.

Defect (дефект) – различие между ожидаемым и фактическим результатом (программа или система не делает то, что должна).

Failure (отказ) – сбой, к которому может привести дефект.

Bug (баг) – ошибка в программе или системе, которая выдает неожиданный или неправильный результат.

Пример:



Основные функции:

- вскипятить воду;
- налить воду в чашку;
- вместимость воды;
- перемещение воды в пространстве.

Тестирование свойства: *вскипятить воду*

Шаги проверки:




- налить воду в чайник;

Ожидаемый результат:

- вода вскипела;



OK

<ul style="list-style-type: none"> - поставить на подставку; - включить; - подождать; - визуально оценить целостность чайника. 	<ul style="list-style-type: none"> - чайник целый. 	
Тестирование свойства: <i>налить воду в чашку</i>		
<i>Шаги проверки:</i> <ul style="list-style-type: none"> - налить воду в чайник; - налить воду в чашку. 	<i>Ожидаемый результат:</i> <ul style="list-style-type: none"> - чайник целый: - количество воды в чайнике уменьшилось. 	 OK
Тестирование свойства: <i>вместимость воды</i>		
<i>Шаги проверки:</i> <ul style="list-style-type: none"> - налить воду в чайник; - визуально оценить целостность чайника. 	<i>Ожидаемый результат:</i> <ul style="list-style-type: none"> - чайник целый: - количество воды в чайнике не изменилось. 	 OK
Тестирование свойства: <i>перемещение воды в пространстве</i>		
<i>Шаги проверки:</i> <ul style="list-style-type: none"> - налить воду в чайник; - поднять чайник; - перенести чайник; - визуально оценить целостность чайника. 	<i>Ожидаемый результат:</i> <ul style="list-style-type: none"> - чайник целый; - количество воды в чайнике не изменилось. 	 OK

3.Разработка требований и тестовые артефакты

Артефакты — это некоторые побочные продукты, возникающие в процессе проектирования, разработки, тестирования программного обеспечения.

<i>Разработка требований</i>	<i>Требования</i> – совокупность утверждений относительно атрибутов, свойств или качеств разрабатываемого программного обеспечения.
	<i>Спецификация</i> – законченное описание поведение программы, которую требуется разработать.

	Функциональные требования – требуемые характеристики системы (функциональность).
	Нефункциональные требования – требования, которые не влияют на основную функциональность системы.
Тестовые артефакты	Тестовый случай (тест, test case) – совокупность шагов, конкретных условий и параметров, необходимых для проверки реализации тестируемой программы, функции.
	Ошибка (дефект, bug) – отклонение фактического результата от ожидаемого.
	Отчет об ошибке (bug report) – это документ, описывающий ситуацию, которая привела к обнаружению ошибки, фактический и ожидаемый результат.

Что такое тестирование? Что такое ПО?

Тестирование ПО	ПО – совокупность программ системы обработки информации, соответствующая документация и данные, относящиеся к функционированию системы.
	Тестирование ПО – проверка и оценка соответствия между реальным и ожидаемым поведением программы.

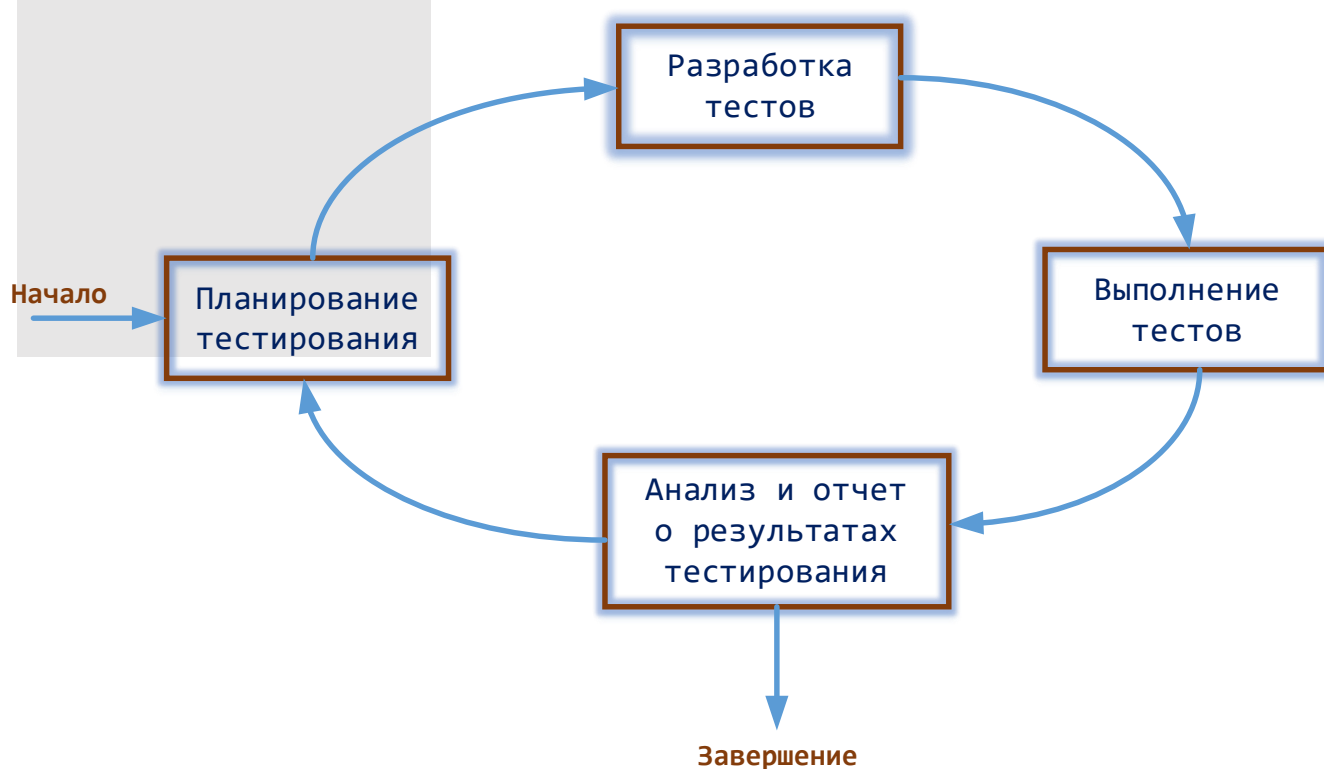
4.Цели и задачи тестирования

Цели тестирования	● убедиться, что ПО отвечает заявленным требованиям.
	● выявить ситуации, в которых поведение программы является неправильным, нежелательным или не соответствующим требованиям.

Задачи тестирования	● убедиться, что ПО отвечает заявленным требованиям.
----------------------------	---

	<ul style="list-style-type: none"> • выявить ситуации, в которых поведение программы является неправильным, нежелательным или не соответствующим требованиям.
	<ul style="list-style-type: none"> • предотвратить как можно больше дефектов
	<ul style="list-style-type: none"> • проверить, что известные дефекты устранены
	<ul style="list-style-type: none"> • проверить, что при устранении известных дефектов, не было внесены новые дефекты
	<ul style="list-style-type: none"> • информировать всех заинтересованных лиц о качестве системы.

5. Цикл тестирования:



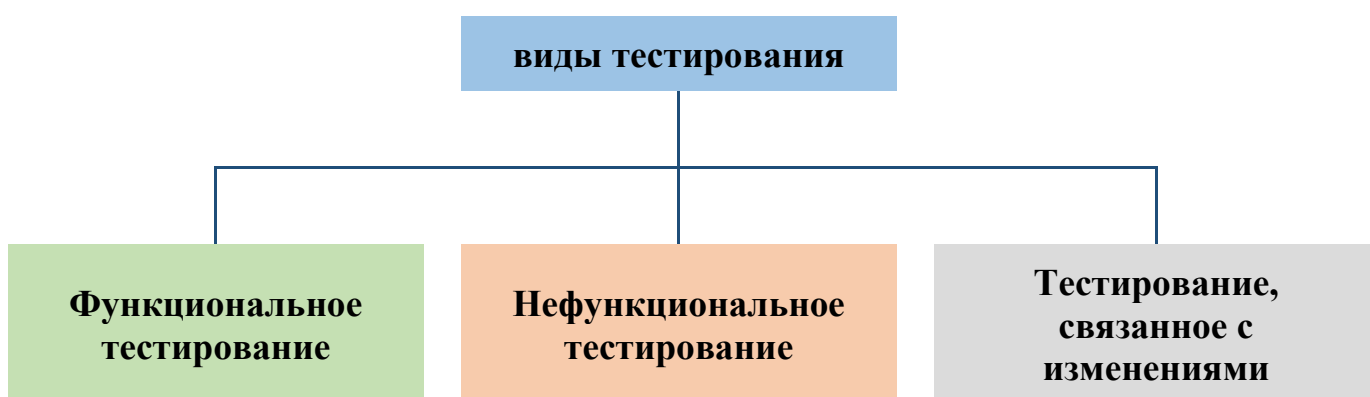
<i>Принципы тестирования</i>	1. Тестирование демонстрирует наличие дефектов.
	2. Исчерпывающее тестирование недостижимо.
	3. Раннее тестирование.
	4. Скопление дефектов.
	5. Парадокс пестицида.

	6. Тестирование зависит от контекста.
	7. Заблуждение, что ошибки отсутствуют.

6.Классификация видов тестирования

Все виды тестирования программного обеспечения, в зависимости от преследуемых целей, можно условно разделить на следующие группы:

Классификация видов тестирования



Функциональные виды тестирования:

Функциональное тестирование рассматривает заранее указанное поведение и основывается на анализе функциональных требований.

Тестирование безопасности проверяет безопасность системы, а также выполняется для анализа рисков, связанных с защитой приложения, атак хакеров, вирусов, несанкционированного доступа к конфиденциальным данным.

Тестирование взаимодействия оценивает способность приложения взаимодействовать с одним и более компонентами или системами.

Нефункциональные виды тестирования:

Тестирование производительности – определение масштабируемости приложения под нагрузкой. Виды тестирования производительности:

- нагрузочное тестирование (Performance and Load Testing);
- стрессовое тестирование (Stress Testing);
- тестирование стабильности или надежности (Stability / Reliability Testing);

- объемное тестирование (Volume Testing).

Тестирование установки направленно на проверку успешной инсталляции и настройки, а также обновления или удаления программного обеспечения.

Тестирование удобства пользования – это метод тестирования, направленный на установление степени удобства использования, обучаемости, понятности и привлекательности для пользователей разрабатываемого продукта в контексте заданных условий. [ISO 9126]

Тестирование на отказ и восстановление (Failover and Recovery Testing) проверяет тестируемое ПО на способность противостоять и успешно восстанавливаться после возможных сбоев (ошибки программного обеспечения, отказы оборудования или проблемы связи).

Конфигурационное тестирование (Configuration Testing) направлено на проверку работы ПО при различных конфигурациях системы (заявленных платформах, поддерживаемых драйверах, при различных конфигурациях компьютеров и т.д.).

Тестирование, связанное с изменениями:

Дымовое тестирование – минимальный набор тестов, проверяющих базовую функциональность (нового или исправленного ПО).

Регрессионное тестирование – это вид тестирования направленный на проверку изменений, сделанных в приложении или окружающей среде (починка дефекта, слияние кода, миграция на другую операционную систему, базу данных, веб сервер или сервер приложения), для подтверждения того факта, что существующая ранее функциональность работает как и прежде (используются тест кейсы, написанные на ранних стадиях разработки и тестирования).

Тестирование сборки (Build Verification Test) – это тестирование, направленное на определение соответствия, выпущенной версии, критериям качества для начала тестирования.

Санитарное тестирование или проверка согласованности/исправности (Sanity Testing) – это узконаправленное тестирование для доказательства того, что **конкретная функция** работает согласно заявленным в спецификации требованиям (подмножество регрессионного тестирования). Используется для определения работоспособности определенной части приложения после изменений произведенных в ней или окружающей среде. **Обычно выполняется вручную.**

7.Уровни тестирования программного обеспечения

Уровни тестирования программного обеспечения

тестирование на разных уровнях производится на протяжении всего жизненного цикла разработки и сопровождения программного обеспечения. Уровень

тестирования определяет то, над чем производятся тесты: над отдельным модулем, группой модулей или системой, в целом.

Компонентное или Модульное тестирование (Component Testing or Unit Testing) проверяет функциональность и ищет дефекты в отдельных частях приложения (модули программ, объекты, классы, функции и т.д.).

Интеграционное тестирование (Integration Testing) предназначено для проверки связи между компонентами, а также взаимодействия с различными частями системы (операционной системой, оборудованием либо связи между различными системами). **Подходы к интеграционному тестированию:**

- снизу вверх (Bottom Up Integration);
- сверху вниз (Top Down Integration);
- большой взрыв («Big Bang» Integration).

Системное тестирование (System Testing) – это тестирование проверка как функциональных, так и нефункциональных требований в системе в целом.

Приемочное тестирование (Acceptance Testing) – это формальный процесс тестирования, который проверяет соответствие системы требованиям и проводится с целью:

- определения удовлетворяет ли система приемочным критериям;
- вынесения решения заказчиком принимается приложение или нет.

Используется набор типичных тестовых случаев и сценариев, разработанный на основании требований к данному приложению.

Тестирование по признаку позитивности:

Позитивное тестирование – это тестирование на тестовых данных или сценариях, которые соответствуют ожидаемому (штатному, нормальному) поведению системы согласно техническим требованиям и документации.

Негативное тестирование – это тестирование на тестовых данных или сценариях, которые соответствуют внештатному поведению тестируемой системы. Это могут быть исключительные ситуации (ошибки) или неверные данные.

Пример. Позитивное тестирование:

- умножить на калькуляторе целые числа 3 и 5;
- в игре посадить морковь на грядку для овощей;

- оплатить покупку действующей картой.

Пример. Негативное тестирование:

- умножить на калькуляторе число 3 на грушу (значение «груша» не является валидным для калькулятора);
- в игре посадить морковь на асфальте;
- оплатить покупку несуществующей картой.

Сценарии **позитивного тестирования** направлены на проверку работы системы с теми типами данных для которых, она разрабатывалась.

Негативное тестирование направлено на проверку устойчивости системы к различным воздействиям, валидации неверных данных, обработку исключительных ситуаций и т.п.

Реакция продукта на тесты. Какой результат можно ожидать от позитивных и негативных тестов?

Позитивное тестирование должно всегда давать результат в виде **отсутствия багов**.

Негативное тестирование могут дать 2 результата:

1. На данный ввод у объекта **есть ответ** в виде либо сообщения, либо определенного для данной ситуации действия.
2. Система **не знает**, как реагировать на введенные данные.

Существует **три реакции** на действия по вводу данных:

- **действие**: создание новой сущности, переход на новый шаг и т.п.
- **контроль**: сообщение с контролем, блокировка дальнейших действий и т.п.
- **отказ**: возникает исключение (Exception, 404-я ошибка и т.п.).

Создание позитивных сценариев (тест-кейсов), как правило, предшествует созданию негативных.

8. Методы тестирования ПО

Тестирование по степени подготовленности к тестированию:

Тестирование по документации – тестирование проводится по заранее подготовленным тестовым случаям.

Интуитивное тестирование – тестирование проводится без какой-либо подготовки, без цели и плана.

Исследовательское тестирование – тестирование с целью изучения проекта и документации на него, но без подготовленной заранее тестовой документации

Тестирование по степени автоматизации:

Ручное тестирование – все тестирование проводится тестировщиком вручную без помощи скриптов.

Полуавтоматизированное тестирование – часть тестирования проводится вручную, а часть автоматизирована.

Автоматизированное тестирование – тестирование полностью автоматизировано.

Тестирование по знанию системы:

Тестирование черного ящика – тестировщик не имеет доступа к коду.

Тестирование белого ящика – тестировщик имеет доступа к коду.

Тестирование серого ящика – тестировщик знает общую структуру приложения.

Тест Дизайн (Test Design)

Роли, ответственные за тест дизайн

- Тест аналитик – определяет «**ЧТО тестировать?**»
- Тест дизайнер – определяет «**КАК тестировать?**»

План тестирования

Тест план (Test Plan) – это документ, описывающий работы по тестированию, начиная с описания объекта, стратегии, расписания, критериев начала и окончания тестирования, до всего необходимого в процессе работы (оборудования, специальных знаний, оценки рисков, вариантов их разрешения).

Каждая методология разработки ПО предлагают свои форматы оформления планов тестирования. Например:

- Test Plan Template RUP
- Test Plan Template IEEE 829

Шаблон тест плана:

- Что надо тестировать?
 - описание объекта тестирования: системы, приложения, оборудования, ...
- Что будете тестировать?

- список функций и описание объекта тестирования и его состав
- Как будете тестировать?
 - стратегия тестирования, а именно: виды тестирования и их применение по отношению к объекту тестирования
- Когда будете тестировать?
 - последовательность проведения работ: подготовка (Test Preparation), тестирование (Testing), анализ результатов (Test Result Analysis)
- Критерии начала тестирования:
 - готовность тестовой платформы (тестового стенда)
 - законченность разработки требуемого функционала
 - наличие всей необходимой документации
 - ...
- Критерии окончания тестирования:
 - результаты тестирования удовлетворяют критериям качества продукта:
 - требования к количеству открытых багов выполнены
 - выдержка определенного периода без изменения исходного кода приложения Code Freeze (CF)
 - выдержка определенного периода без открытия новых багов Zero Bug Bounce (ZBB)
 - ...
- Окружение тестируемой системы (описание программно-аппаратных средств)
- Необходимое для тестирования оборудование и программные средства (тестовый стенд и его конфигурация, программы для автоматизированного тестирования и т.д.)
- Риски и пути их разрешения

Набор тест кейсов и тестов (Test Case & Test suite) – это последовательность действий, по которой можно проверить соответствует ли тестируемая функция установленным требованиям.

Тестовый случай (Test Case) – это артефакт, описывающий совокупность шагов, конкретных условий и параметров, необходимых для проверки реализации тестируемой функции или её части.

Структура Test Case:

Action (Действие)	>	Expected Result (Ожидаемый результат)	>	Test Result (Результат тестирования) (passed/failed/blocked)
-----------------------------	---	---	---	---

Результат тест кейса: *passed* / *failed* / *blocked* (успешно / ошибка / заблокирован)

Пример:

Действие	Ожидаемый результат	Результат тестирования (passed / failed / blocked)
сложение на калькуляторе целых чисел 3 и 5	целое число, равное сумме чисел 3 и 5	passed (или успешно)

Виды тестовых случаев (по ожидаемому результату):

тест кейсы разделяются по ожидаемому результату на **позитивные** и **негативные**

Позитивный тест кейс

- использует только корректные данные
- проверяет, что приложение правильно выполнило вызываемую функцию.

Позитивный тест кейс

- использует корректные данные
- использует некорректные данные (минимум 1 некорректный параметр)
- проверяет, что приложение не выполняет вызываемую функцию (срабатывание валидатора).

! При автоматизированном тестировании актуально:

Каждый тест кейс должен иметь 3 части:

PreConditions (предусловия)	Список действий, которые приводят систему к состоянию готовности для проведения основного тестирования.
Test Case Description (описание действий)	Список действий, переводящих систему из одного состояния в другое, для получения результата тестирования
PostConditions (постусловия)	Список действий, переводящих систему в первоначальное состояние (состояние до проведения теста – <i>initial state</i>)

Примечание: *постусловия* не является обязательной частью. Это правило хорошего тона: «*намусорил – убери за собой*» (актуально при автоматизированном тестировании, т.к. за один прогон база данных наполняется сотней и более некорректных документов).

Пример позитивного тестирования

Тест кейс 1:

Проверка отображения страницы		
Действие	Ожидаемый результат	Результат теста
Открыть страницу «Вход в систему»	<ul style="list-style-type: none">- окно «Вход в систему» открыто- название окна – Вход в систему- логотип компании отображается в левом верхнем углу- на форме 2 поля – Логин и Пароль- кнопка Вход доступна- ссылка «забыл пароль» – доступна	...

9.Процесс тестирования

Автоматизация – ручное тестирование (или иное).

Последовательность действий:

1. Тестирование начинается после получения спецификаций на разрабатываемое ПО.
2. Подготовлен первый прототип. Необходимо провести **дымовое тестирование**, по результатам которого делается вывод о возможности дальнейшего тестирования.
3. В случае если «*smoke test failed!!!*», переходим к п.2 (приложение отправляется на доработку).
4. В случае если «*smoke test passed!!!*», переходим к следующему виду тестирования – **регрессионное тестирование** (Regression testing) и **санитарное тестирование** (Sanity testing).

Последовательность выполнения тестирования:

Дымовое > Регрессионное > все остальные виды тестирования

Введение

Проектов без документации не существует, вопрос в том, что что именно должно входить в конкретный базовый набор на старте и как оптимально поддерживать документацию в процессе.

Для создания действительно полезных документов надо учитывать два фактора:

- цель – для чего создается конкретный документ;
- потребитель – кто будет использовать то, что вы написали.

Базовый набор стартовой документации включает:

1. **Паспорт проекта** – документ, в котором описано кто клиент, что делаем, сроки, цель и т.д.
2. **Фич-лист** обязателен, именно с него начинается проект: РМ оценивает фич-лист, согласовывает с заказчиком, и на его основе создает план проекта.
3. **Иерархическая структура работ** – тот же фич-лист, только более структурированный и распределенный по ролям.
4. **Оценка проекта** – обязательная часть.
5. **План проекта** – важно правильно выбрать инструмент, в котором будете вести план, это может быть: Excel, Miro, другое.

Список самых используемых инструментов:

- **Google Drive** для хранения разного вида файлов с общим доступом и редактированием онлайн.
- **Dropbox** – альтернатива Google Drive.
- **Confluence** – общее рабочее пространство, база знаний для создания, хранения, редактирования документов.
- **Notion** – «википодобный» ресурс. Хорошо работает для базовой документации, особенно о каком-то продукте.
- **Miro** помогает сделать быстрое ревью документа.

Базовые правила ведения документации:

- документация должна быть синхронизированной и, по-возможности, онлайн. Для удаленных команд это особенно актуально: можно быстро редактировать документ и все это видят;
- всегда проверяйте права доступа.

Соглашения об именовании

Обычно достаточно трех простых правил:

- 1) использовать в качестве префикса дату в формате: YYYY-MM-DD;
- 2) значимое обозначение документа: например, Test_Concept_Module-A;
- 3) номер версии документа: v01.

Понятная навигация

1. Документирование

Документация – это рабочее пространство проекта.

- ✓ документирование позволяет четко разграничить зоны ответственности между участниками проекта;
- ✓ документы избавляют от ненужных конфликтов;
- ✓ позволяет в любой момент времени быстро найти нужную информацию и понять, как решать конкретные задачи;

- ✓ в документах четко прописано, кто что делает, кто за что отвечает, как работает система и что делать, если что-то пошло не так;
- ✓ команда говорит «на одном языке»;
- ✓ только тщательно описанные требования могут быть проверены на полноту и непротиворечивость.

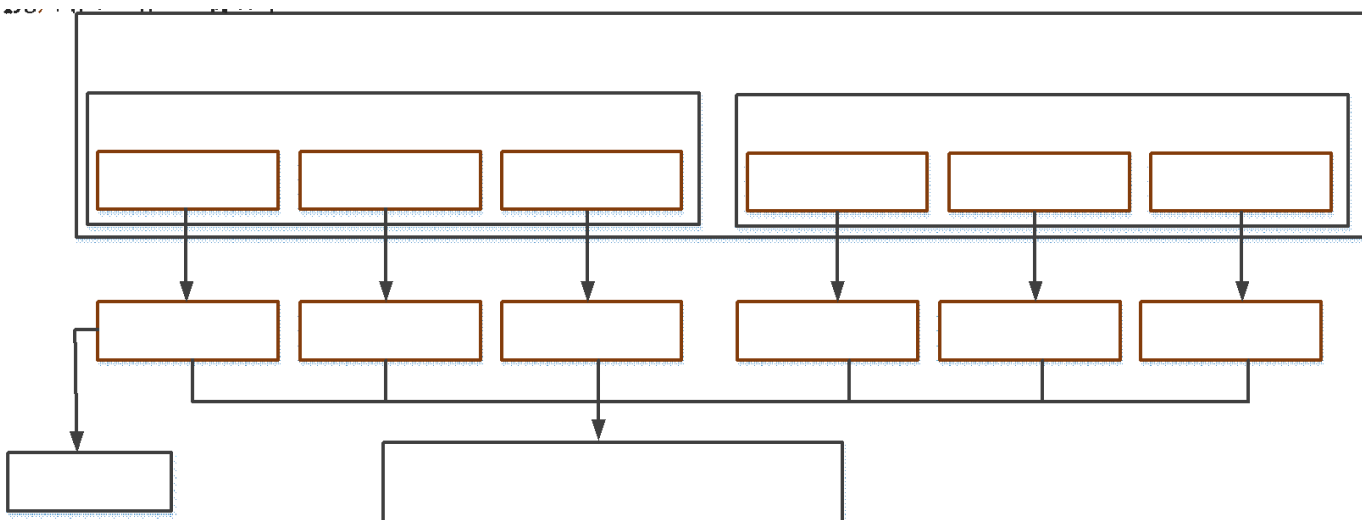
Правила хорошей документации:

- 1) Документация не должна быть избыточной и объемной. Избыточное количество текста – раздражает и затрудняет восприятие.
- 2) Вся схема документирования проекта должна быть взаимоувязанной и логичной. Если в схеме существует документ, который не связан ссылкой с каким бы то ни было другим документом, то его можно безболезненно из схемы исключить.
- 3) Вся оценка трудозатрат должна производиться только на основании описанных атомарных задач. Чем мельче оцениваемый элемент – тем точнее будет агрегированная оценка.
- 4) Всегда необходимо формировать списки оповещения заинтересованных участников.

Типы документов:

- 1) Техническое задание.
- 2) Частное техническое задание (опционально).
- 3) Сценарий использования (Use Case).
- 4) Сценарий тестирования (Test Case).
- 5) Отчет об ошибке (Bug Report).
- 6) Руководство пользователя.
- 7) Руководство администратора (опционально).

Схема связи между документами.



где

Техническое задание

В первую очередь формируется ***Техническое задание*** системы.

ТЗ включает в себя:

- словарь терминов предметной области;
- описание предметной области;
- описание ролевой системы (действующие лица);
- описание функциональных требований;
- описание нефункциональных требований.

Все требования формируются на основании описания бизнес-процессов заказчика (функциональные требования).

Описание требований в ТЗ фиксирует необходимые функции на верхнем уровне.

Требования оптимально разбивать на смысловые группы по подсистемам.

Например

На верхнем уровне системы «Калькулятор» описываем подсистему «Калькулятор. Арифметические операции» с функциями «Выполнить сложение», «Выполнить вычитание» и т.д.

Частные технические задания (ЧТЗ) подсистем должны содержать:

- ссылку на пункт ТЗ;
- максимально подробную информацию по каждой функции;
- список Use Cases для функции.

Таким образом реализуется преемственность документов, что позволяет:

- во-первых, унифицировать их форму;
- во-вторых – частично реализовать повторное использование, то есть снизить затраты времени на повторное написание кода.

Например

формируем ЧТЗ на подсистему «Калькулятор. Арифметические операции». Тогда описание функции:

«Выполнить сложение.

Необходимо реализовать следующий функционал:

выполнить бинарную арифметическую операцию сложения целых чисел;

входные данные: два целых числа.

выходные данные: целое число;

...».

Use Case

Use Case – вариант использования, который описывает все действия, которые пользователь может произвести, и реакцию системы на эти действия.

Каждый вариант использования должен быть привязан к своему пункту ЧТЗ.

Наиболее оптимальным и простым форматом описания является UML-диаграммы вариантов использования, выполненные в Visio или с помощью аналогичного инструмента.

Необходимо включать описание входных и выходных данных: название переменной, её тип, ограничение на ввод данных (логические проверки и т.д.); для вычисляемых переменных указать формулу для расчета значения.

Test Case

Тестовый случай должен содержать описание тестовых сценариев.

Каждый такой документ привязывается к соответствующему Use Case

Структура Test Case:

Action (Действие)	>	Expected Result (Ожидаемый результат)	>	Test Result (Результат тестирования) (passed/failed/blocked)
-----------------------------	---	---	---	---

Результат тест кейса: *passed / failed / blocked* (успешно / ошибка / заблокирован)

Не забыть про проверку валидности выходных данных!

Bug Report

Отчет об ошибке (Bug Report) возникает в процессе тестирования системы как реакция на ошибку. Каждый документ должен обязательно ссылаться на соответствующий Test Case.

Документ должен содержать:

- описание предшествующих действий (можно разработать удобный для всех шаблон такого описания – это позволит сэкономить время разработчикам при воспроизведении бага);
- текстовое описание самой ошибки;
- скриншот возникшей ошибки (при необходимости).

Руководство пользователя | Руководство администратора

Это жизненно необходимые документы.

Они являются документацией для пользователей – это пользователи приложений. Можно использовать подход к построению документации по принципу описания вариантов использования (Use Case). По списку задач, которые выполняют пользователи, составляются пошаговые инструкции.

Если все Use Cases и Test Cases были тщательно описаны, аккуратно выполнены и правильно оформлены, то эти документы будут легко сформированы (почти автоматически).

Документация для разработчиков: это могут быть комментарии в коде, readme- файлы, стиль оформления кода. Логике приложения следует описать в виде блок-схем или псевдокода.

Начинать процесс документирования всегда проще «сверху» (принцип модульного проектирования сверху вниз).

Руководство пользователя

- введение
 - область применения
 - краткое описание возможностей
 - требования к уровню подготовки пользователя
 - перечень эксплуатационных документов, с которыми необходимо ознакомиться пользователю
- назначение и условия применения
 - виды деятельности и функции для автоматизации которых предназначено данное ПО
 - условия, при соблюдении которых обеспечивается применение ПО в соответствии с назначением
- подготовка к работе
 - состав и содержание дистрибутивного носителя данных
 - порядок загрузки данных и программ
 - порядок контроля и проверки работоспособности
- описание операций – для каждой операции обработки данных должно быть указано
 - наименование
 - условия, при соблюдении которых возможно выполнение операции
 - подготовительные действия
 - основные действия в требуемой последовательности
 - заключительные действия
 - ресурсы, расходуемые на операцию
 - описание всех выполняемых функций, задач, комплексов задач, процедур
 - описание операций технологического процесса обработки данных, необходимых для выполнения функций, комплексов программ, процедур.
- аварийные ситуации
 - действия в случае несоблюдения условий выполнения технологического процесса, в том числе при длительных отказах технических средств
 - действия по восстановлению программ и данных при отказе или обнаружении ошибок в данных
 - действия в случае обнаружения несанкционированного вмешательства в данные системы

2.Разработка документации программных средств и её стандартизация

Процесс документирования по стандарту ISO/IEC 12207:1995

Рост применения программных средств и их сложность требуют полной, точной и понятной документации на ПО, доступной пользователям.

Процесс документирования программных средств и систем регламентирует международный стандарт ***ISO/IEC 12207:1995*** и ***СТБ ИСО/МЭК12207-2003***.

В данном стандарте **процесс документирования** определяется как процесс формализованного описания информации, созданной в процессе или работе жизненного цикла.

Документирование включает планирование, проектирование, разработку, выпуск, редактирование, распространение и сопровождение документов по программному продукту.

Документация разработки

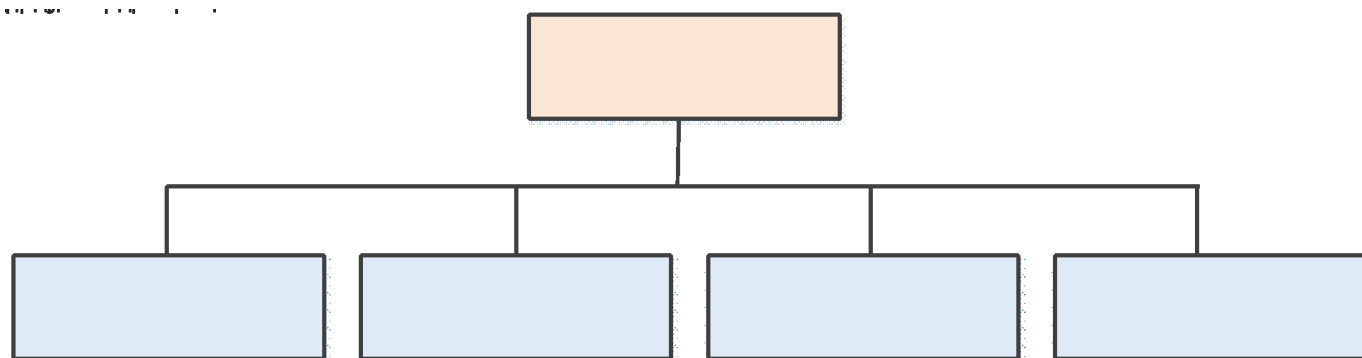
Разработка документов преследует пять целей;

- 1) документы являются средством связи между всеми вовлеченными в процесс разработки. Они описывают подробности решений, принятых относительно требований к программному обеспечению, проекту, программированию и тестированию;
- 2) документы описывают обязанности группы разработки. Они определяют, кто, что и когда делает, учитывая роль программного обеспечения, предмета работ, документации, персонала, обеспечивающего качество, и каждого вовлеченного в процесс разработки;
- 3) документы выступают как контрольные точки, которые позволяют руководителям оценивать ход разработки. Если документы разработки отсутствуют, неполны или устарели, руководители теряют важное средство для отслеживания и контроля проекта программного обеспечения;
- 4) документы образуют основу документации сопровождения программного обеспечения;
- 5) документы описывают историю разработки программного обеспечения.

Типовыми документами разработки являются:

- анализы осуществимости и исходные заявки;
- спецификации требований;
- спецификации функций;
- проектные спецификации, включая спецификации программ и данных;
- планы разработки;
- планы сборки и тестирования программного обеспечения;
- планы обеспечения качества, стандарты и графики;
- защитная и тестовая информация.

Структура процесса документирования в соответствии с СТБ ISO/IEC 12207



В соответствии с **ISO/IEC 12207** процесс документирования включает в себя:

- подготовку процесса документирования:
 - разработку и реализацию плана обозначения документов, выпускаемых в процессах жизненного цикла программных средств;
- проектирование и разработку документации:
 - проектирование документов согласно стандартам на документацию;
 - подтверждение источника и соответствия исходных материалов для документов;
 - проверку и редактирование документов согласно стандартам, утверждение компетентными лицами;
- выпуск документации:
 - издание и распространение документов в соответствии с планом;
 - управление документированием в соответствии с процессом управления конфигурацией;
- сопровождение документации:
 - внесение изменений в документацию согласно процессам сопровождения и управления конфигурацией.

В плане обозначения документов должны быть определены:

- заголовок или наименование;
- назначение;
- пользователи документа;
- процедуры и обязанности по подготовке исходных материалов, разработке, проверке, изменению, утверждению, выпуску, хранению, распространению, сопровождению и управлению конфигурацией документов.

Реализация процесса документирования в соответствии со стандартом ISO/IEC 15910:1999

В 1999 году введен в действие международный стандарт ISO/IEC 15910:1999 – Информационная технология – Процесс создания документации пользователя программного средства (аналогичный стандарт ГОСТ Р ИСО/МЭК 15910-2002 действует с 2003 г.).

Стандарт ISO/IEC 15910 определяет одну из реализаций процесса документирования, описанного в ISO/IEC 12207, и предоставляет пользователю метод применения данного процесса при создании конкретного программного средства. Стандарт содержит структуру комплексного плана разработки документации, однако не определяет состав требований к стилю оформления документов, устанавливая только их диапазоны.

Стандарт ISO/IEC 15910 должен использоваться при создании пользовательской документации всех видов:

- печатной документации (руководство пользователя, краткие справочные карты);
- диалоговой (оперативной) информации;

- справочного текста (Help);
- системы диалоговой документации.

Основные определения стандарта ISO/IEC 15910:1999

Стандарт *ISO/IEC 15910* использует следующие основные определения.

Аудитория (*audience*): категория пользователей, предъявляющих к документации одинаковые или аналогичные требования, определяющие содержание, структуру и назначение данной документации.

Документатор (*do cum enter*): сторона, создающая документацию.

Справочный текст (*help text*): текст, автоматически выбираемый в зависимости от контекста, в котором он вызывается, облегчающий и убыстряющий при эксплуатации ПС поиск содержащихся в издании объектов. Справочный текст контекстно зависим.

Диалоговая документация (*on-line documentation*): информация, доступная пользователю при эксплуатации ПС, которая необязательно привязана к конкретному контексту.

Система диалоговой информации или справочная система (*on-line documentation system or help system*): часть программы или отдельная программа, запрашиваемая пользователем и позволяющая ему просматривать части диалоговой документации или справочного текста.

Продукт (*программный продукт, product*): полный набор компьютерных программ, процедур и соответствующих им документации и данных, предназначенный для поставки пользователю.

Тестирование на практичность (*usability testing*): формальный процесс оценки соответствия документации установленным требованиям.

3. Программная документация

Существует четыре основных типа документации на ПО:

- архитектурная/проектная – обзор программного обеспечения, включающий описание рабочей среды и принципов, которые должны быть использованы при создании ПО;
- техническая – документация на код, алгоритмы, интерфейсы, API;
- пользовательская – руководства для конечных пользователей, администраторов системы и другого персонала;
- маркетинговая.

Программную документацию по отношению к пользователю делят на:

- внутренняя (технологическая) – используется в процессе разработки ПО и недоступна пользователю;
- внешняя (пользовательская) – всевозможные руководства для пользователя.

Документы, описывающие процесс разработки программного обеспечения, определяют требования, которым должно удовлетворять программное обеспечение, определяют проект программного обеспечения, определяют, как его контролируют и как обеспечивают его качество. Документация разработки, также включает в себя подробное техническое описание программного обеспечения (программную логику, программные взаимосвязи, форматы и хранение данных и т. д.).

Стандарт документирования

ГОСТ при разработке программных продуктов дает возможность:

- унифицировать программные изделия для взаимного обмена и применения ранее разработанных программ в новых разработках;
- снизить трудоемкость и повысить эффективность разработки, сопровождения, изготовления и эксплуатации программных продуктов;
- автоматизировать изготовление и хранение программной документации.
- основу отечественной нормативной базы в области документирования ПО составляет **комплекс стандартов Единой системы программной документации (ЕСПД)**.
- стандарты ЕСПД в основном охватывают ту часть документации, которая создается в процессе разработки ПО, и связаны, по большей части, с документированием функциональных характеристик ПО.

Значение ЕСПД

- стандарты ЕСПД вносят элемент упорядочения в процесс документирования ПС;
- предусмотренный стандартами ЕСПД состав программных документов не является «жестким»: стандарты позволяют вносить в комплект документации на ПС дополнительные виды;
- стандарты ЕСПД позволяют мобильно изменять структуры и содержание установленных видов программной документации исходя из требований заказчика и пользователя.

Недостатки ЕСПД

- ориентацию на единственную, «каскадную» модель жизненного цикла ПО;
- отсутствие четких рекомендаций по документированию характеристик качества ПО;
- отсутствие системной увязки с другими действующими отечественными системами стандартов по ЖЦ и документированию продукции в целом;
- нечетко выраженный подход к документированию ПО как товарной продукции;

- отсутствие рекомендаций по самодокументированию ПО;
- отсутствие рекомендаций по составу, содержанию и оформлению перспективных документов на ПО, согласованных с рекомендациями международных и региональных стандартов.

Группы стандарта ЕСПД

Код группы	Наименование группы
0	Общие положения
1	Основополагающие стандарты
2	Правила выполнения документации разработки
3	Правила выполнения документации изготовления
4	Правила выполнения документации сопровождения
5	Правила выполнения эксплуатационной документации
6	Правила обращения программной документации

Перечень документов ЕСПД

- ГОСТ 19.001-77 ЕСПД. Общие положения.
- ГОСТ 19.101-77 ЕСПД. Виды программ и программных документов.
- ГОСТ 19.102-77 ЕСПД. Стадии разработки.
- ГОСТ 19.103-77 ЕСПД. Обозначение программ и программных документов.
- ГОСТ 19.104-78 ЕСПД. Основные надписи.
- ГОСТ 19.105-78 ЕСПД. Общие требования к программным документам.
- ГОСТ 19.106-78 ЕСПД. Требования к программным документам, выполненным печатным способом.
- ГОСТ 19.201-78 ЕСПД. Техническое задание. Требования к содержанию и оформлению.
- ГОСТ 19.202-78 ЕСПД. Спецификация. Требования к содержанию и оформлению.
- ГОСТ 19.301-79 ЕСПД. Порядок и методика испытаний
- ГОСТ 19.401-78 ЕСПД. Текст программы. Требования к содержанию и оформлению.
- ГОСТ 19.402-78 ЕСПД. Описание программы.
- ГОСТ 19.404-79 ЕСПД. Пояснительная записка. Требования к содержанию и оформлению.

- ГОСТ 19.501-78 ЕСПД. Формуляр. Требования к содержанию и оформлению.
- ГОСТ 19.502-78 ЕСПД. Описание применения. Требования к содержанию и оформлению.
- ГОСТ 19.503-79 ЕСПД. Руководство системного программиста. Требования к содержанию и оформлению.
- ГОСТ 19.504-79 ЕСПД. Руководство программиста.
- ГОСТ 19.505-79 ЕСПД. Руководство оператора.
- ГОСТ 19.506-79 ЕСПД. Описание языка.
- ГОСТ 19.508-79 ЕСПД. Руководство по техническому обслуживанию. Требования к содержанию и оформлению.
- ГОСТ 19.604-78 ЕСПД. Правила внесения изменений в программные документы, выполняемые печатным способом.
- ГОСТ 19.701-90 ЕСПД. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения.
- ГОСТ 19.781-90. Обеспечение систем обработки информации программное.