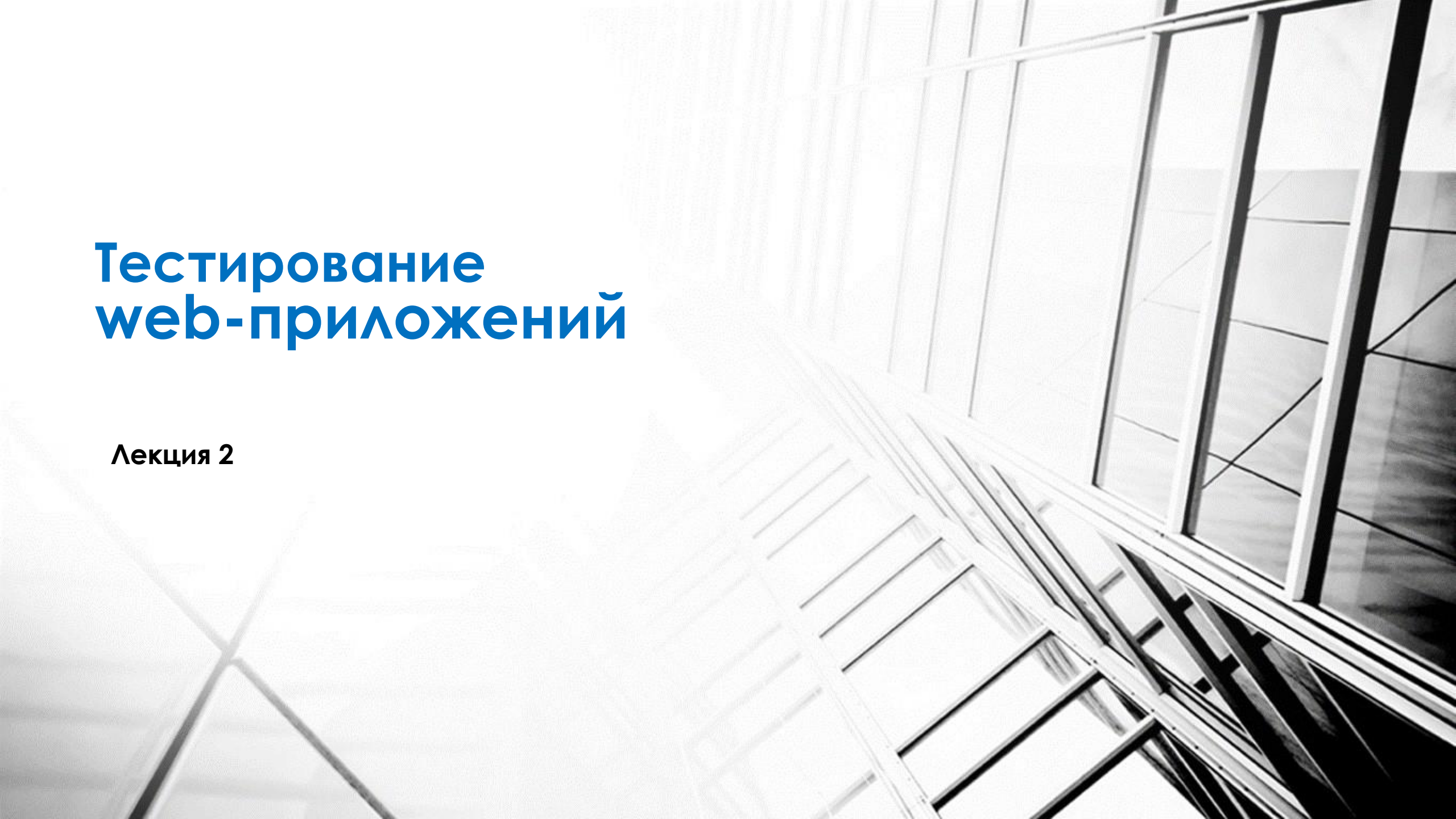


# Тестирование web-приложений

## Лекция 2



- Разработка тестов
  - Техники тест дизайна
  - Планирование тестовых испытаний.
- Тест план.
- Документирование тестов. Тест-кейсы. Чек-листы.

# Разработка тестов

# Тестовое покрытие

# Покрывтие тестами кода (Code Coverage)

**Расчет тестового покрытия относительно исполняемого кода программного обеспечения проводится по формуле:**

$$T_{cov} = (L_{tc}/L_{code}) * 100\%$$

где:

$T_{cov}$  - тестовое покрытие;

$L_{tc}$  - кол-ва строк кода, покрытых тестами

$L_{code}$  - общее кол-во строк кода.

# Матрица тестового покрытия (Requirements Traceability Matrix)

Это двумерная таблица, содержащая соответствие функциональных требований (functional requirements) продукта и подготовленных тестовых сценариев (test cases).

Требования	Тестовое покрытие
1. Open	100, 110, 145, 200
2. Save	310, 312, 350
3. Save as	380, 384
3.1. Save as DOC	380, 384, 388
3.2. Save as TXT	380, 384, 391, 394, 399
3.3. Save as PDF	380, 384, 390
3.4. Save as HTML	380, 384
4. Export	

# Техники тест дизайна



# Техника эквивалентных классов

**Это техника**, при которой мы разделяем функционал (часто диапазон возможных вводимых значений) на группы эквивалентных по своему влиянию на систему значений. Эта техника заключается в разбиении всего набора тестов на классы эквивалентности с последующим сокращением числа тестов.

**Класс эквивалентности (equivalence class)** – набор тестов, полное выполнение которого является избыточным и не приводит к обнаружению новых дефектов.



# Признаки эквивалентности

## Несколько тестов эквивалентны, если:

- Они направлены на поиск одной и той же ошибки
- Если один из тестов обнаруживает ошибку, другие её тоже, скорее всего, обнаружат
- Если один из тестов НЕ обнаруживает ошибку, другие её тоже, скорее всего, НЕ обнаружат
- Тесты используют одни и те же наборы входных данных
- Для выполнения тестов мы совершаем одни и те же операции
- Тесты генерируют одинаковые выходные данные или приводят приложение в одно и то же состояние (например открытие валидных файлов одного типа и схожего объема, но с разным содержанием)
- Все тесты приводят к срабатыванию одного и того же блока обработки ошибок («error handling block»)
- Ни один из тестов не приводит к срабатыванию определенного блока обработки ошибок («error handling block») (например, блок обработки ошибок открытия файла)

# Метод граничных значений

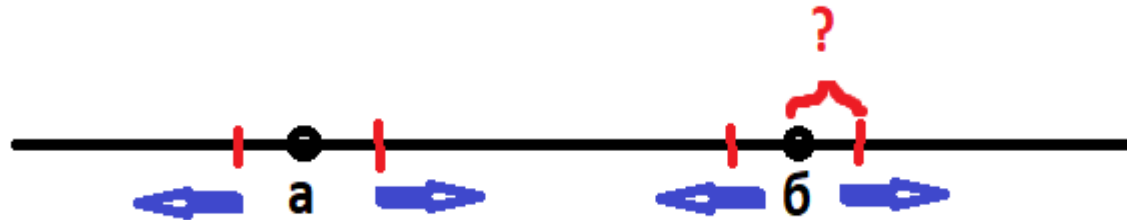
**Метод граничных значений** – техника тест-дизайна, которая направлена на проверку поведения системы на граничных значениях входных данных (границах классов эквивалентности). Очень важно проверять именно граничные значения, потому что довольно часто возникают ошибки именно на границах классов эквивалентности.

! Граничные условия очень важны, и их обязательно следует проверять в тестах, т.к. именно в этом месте чаще всего и обнаруживаются ошибки

# Выводы

- Классы эквивалентности не всегда очевидны.
- Как правило, негативных тестов получается больше, чем позитивных.
- Принадлежность теста к позитивным или негативным зависит от требований.

# Метод граничных значений



1. Какому промежутку относиться.
2. Какое следующее значение в классе после граничного

Пример. Автоматизированная система взвешивает борцов и назначает день соревнований.

$< 60$  кг – 2.01  
 $< 70$  кг – 3.01  
 $< 80$  кг – 4.01  
 $> 100$  кг – 5.01

# Граничные условия

**Пример:** Необходимо проверить, как работает поле, в которое можно ввести целое число в диапазоне от 1 до 99

## Классы эквивалентности:

- Любое целое в диапазоне от 1 до 99. Как правило, это будет середина числового отрезка (Позитивный тест)
- Любое число меньше 1 (Негативный тест)
- Любое число больше 99 (Негативный тест)
- Дробь и «не число» (буквы, спецсимволы) (Негативный тест)

# Граничные условия

- **Тесты, которые мы выполним:**

- Ввести 1, 99, 50
- Ввести 0
- Ввести 100
- Ввести 50.5
- Ввести букву
- Ввести спецсимвол: ~`!"@'#\$;%:^&?\*()[]{}.,\./+=-\_

# Выполните пример

На рейсах авиакомпании при оформлении предварительной оплаты за провоз сверхнормативного багажа ранее, чем за 24 часа до вылета рейса по расписанию действует скидка в размере 50% от общего тарифа.

При оплате за провоз сверхнормативного багажа во время посадки в самолет действует тариф на 20% выше общего тарифа.

Регистрация на рейс начинается за 3 часа до вылета.



# Выводы

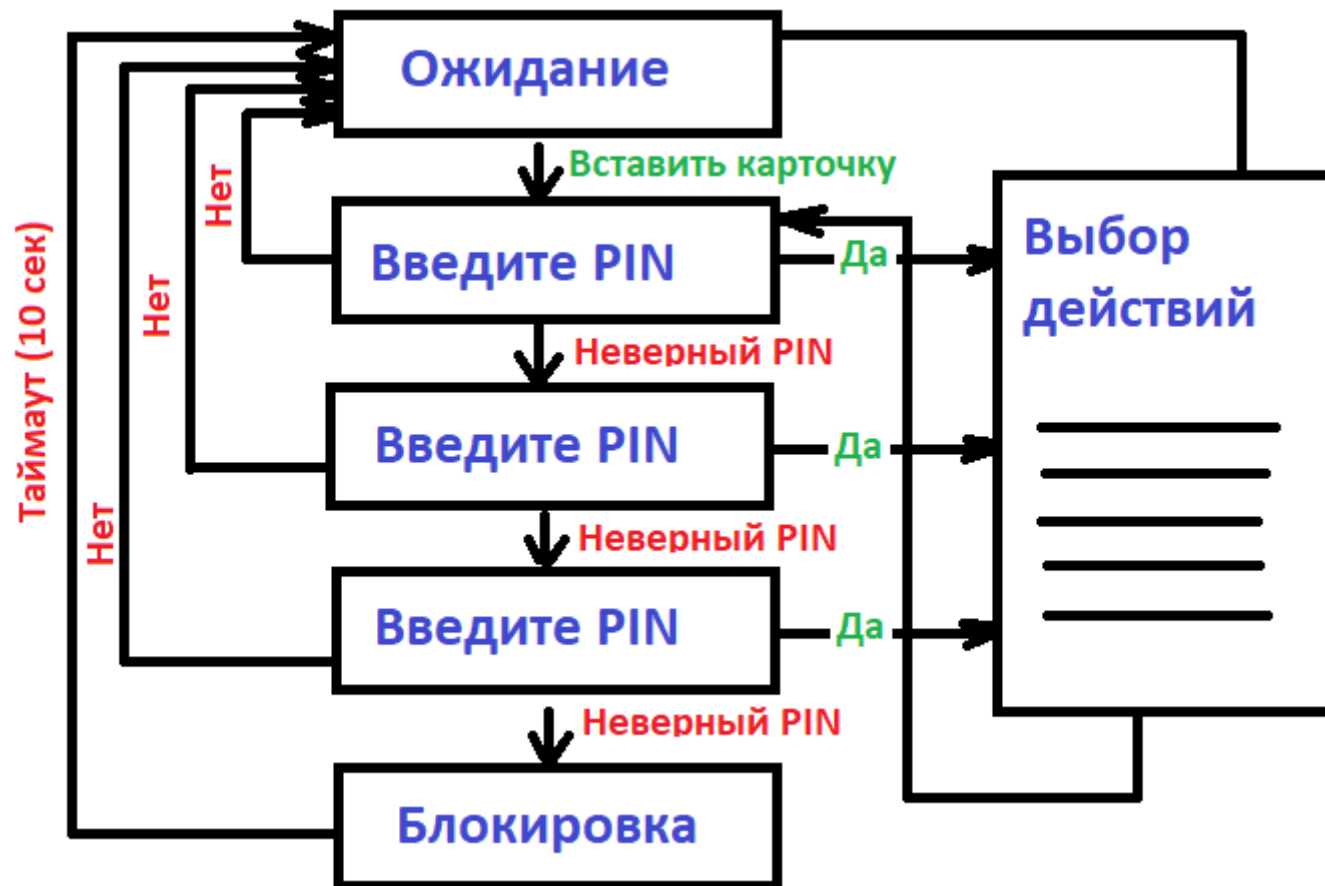
Использование техник разделения на классы **эквивалентности** и анализа **граничных значений** позволяют снизить риск пропуска ошибки, качественно улучшить результаты тестирования, а также значительно сократить количество тестов, необходимых для проведения, и, следовательно, сэкономить время.

Но не стоит также забывать, что важна правильность использования той или иной техники для достижения качественного результата.

# Анализ переходов и состояний системы

Предположим, есть объект, который может находиться в разных состояниях. Переход от одного состояния к другому обычно чем-то обусловлен, например, каким-нибудь действием над объектом. Эти состояния и действия над объектом нам необходимо смоделировать.

# Анализ переходов и состояний системы



# Анализ переходов и состояний системы

Таким образом, мы нарисовали диаграмму состояний и переходов объекта «Банкомат». Первая часть сделана.

Далее необходимо перенести нашу модель в таблицу и составить из нее чек-лист проверок.

# Метод уникальных пар (Pairwise)

**Pairwise testing (попарное тестирование)** – это техника формирования наборов тестовых данных из полного набора входных данных в системе, которая позволяет существенно сократить количество тест-кейсов.

Сформулировать суть попарного тестирования можно следующим образом: формирование таких наборов данных, в которых каждое тестируемое значение каждого из проверяемых параметров хотя бы единожды сочетается с каждым тестируемым значением всех остальных проверяемых параметров.

Главные цели Pairwise Testing:

- убрать избыточные проверки;
- обеспечить хорошее тестовое покрытие;
- выявить наибольшее количество багов на минимальном наборе тестов.

# Метод уникальных пар (Pairwise)

**Пример:** Есть два браузера Opera и Firefox. Есть две операционные системы Windows и Linux. Допустим, сайт на двух языках: русский (RU) и английский (EN).

№	Browser	OS	Language
1	Opera	Windows	RU
2	Firefox	Linux	RU
3	Opera	Linux	EN
4	Firefox	Windows	EN

## Планирование тестовых испытаний

- В небольших проектах тестировщики не углубляются в вопросы качества, не стараются предположить проектные риски, не заботятся о критериях качества и т.д.
- Тестировать нужно не только разные версии программы. Приступать к тестированию следует уже на этапе сбора требований. И тестировать далее на последующих этапах вплоть до выхода продукта. Частенько продукт продолжают тестировать и на этапе поддержки.



## Связь планирования тестовых испытаний с жизненным циклом ПО

Есть много взглядов на жизненный цикл ПО. В контексте тестирования и планирования тестовых испытаний проще всего рассматривать следующую упрощённую модель:

- Начало (inception).
- Уточнение (elaboration).
- Разработка (construction).
- Передача заказчику (transition).

# Связь планирования тестовых испытаний с жизненным циклом ПО

## 1. Начало

### А) сбор требований

- собраны пожелания заказчика ->
- сформулированы бизнес-требования ->
- написаны функциональные требования;

### Б) анализ требований для создания архитектуры и дизайна

- как только появляются первые документы – тестировщики начинают их анализировать и готовиться к тестированию;
- чуть позже разработчики начинают работу

# Связь планирования тестовых испытаний с жизненным циклом ПО

## 2. Уточнение

А) разработчики

- пишут отдельные модули и юнит-тесты

Б) тестировщики

- проводят модульное тестирование и интеграционные тесты

В) обе группы специалистов

- активно уточняют/дорабатывают требования и дизайн

# Связь планирования тестовых испытаний с жизненным циклом ПО

## 3. Разработка

А) программисты

- пишут главные функции продукта (пиковая активность)

Б) тестировщики

- тестируют функциональность на всех уровнях тестирования (пиковая активность в конце фазы)

В) работа с требованиями

- сокращается, в конце фазы уже невозможно внести серьезные изменения

# Связь планирования тестовых испытаний с жизненным циклом ПО

## 4. Передача заказчику

А) команда поддержки

- развертывает систему у заказчика

Б) сторонние тестировщики

- проводят приемочное тестирование

В) тестировщики/программисты

- активность спадает

## Области ответственности тестировщиков:

### Планирование тестов

- разработка методологии и плана тестирования
- участие в принятии стандарта качества
- разработка спецификаций тестов

### Разработка и выполнение тестов

- создание ручных и автоматизированных тестов
- выполнение тестов
- управление билдами (оценка состояния проекта)

### Отчеты по тестам

- сообщить проектной группе о качестве продукта
- отслеживать состояние дефектов

## Планирование и тестовый план

Тест план (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



TEST PLAN



## Планирование и тестовый план

Тестовый план – документ проектной документации, который описывает:

- процесс тестирования конкретного продукта в конкретном проекте;
- что, когда, кем и как будет тестироваться;
- компоненты тестирования;
- команду тестировщиков;
- стратегию и методы тестирования;
- критерии качества и риски тестирования;
- график работ.

## Необходимые действия на стадии планирования

- Понять, что за продукт будет тестироваться, как он работает и для чего он нужен
- Понять, как продукт будет использоваться
- Хорошо изучить требования к продукту
- Решить какие части продукта будут тестироваться и как
- Убедиться, что выбранные области в полном объеме покрыты тестами
- Решить, какие из методов и техник тестирования наиболее эффективны для проверки нашего продукта
- Определить критерии качества продукта
- Определить риски, то есть ситуации, которые приведут к ухудшению качества программного продукта. А также подумать о том, как предотвратить возникновение подобных ситуаций, и как из них выйти
- Выводы по всем вышеперечисленным пунктам попадают в тест-план, который нужно утвердить и разослать всем заинтересованным лицам
- Создать такое тестовое окружение, которое будет максимально приближено к условиям, в которых продукт будет эксплуатироваться (production environment)

## Артефакты, создаваемые на стадии планирования

К артефактам, создаваемым на стадии планирования можно отнести:

- тестовый план;
- матрица конфигураций, которая может быть включена в тестовый план как отдельный раздел;
- запрос на выделение тестового оборудования.



## Сложности планирования

- К сожалению невозможно всё предусмотреть и всё запланировать.

Всегда будут возникать какие-то сложности или непредусмотренные ситуации во время работы.

Будут возникать ситуации, когда будет трудно расставить приоритеты той или иной деятельности: например, какие тесты выполнить в первую очередь, а какие выполнять необязательно, (или выполнить, если останется время); какие виды тестирования более приоритетны, а какие – менее; какая функциональность более важна, а какая – менее.

- Также будут возникать различные ограничения, которые невозможно контролировать или на которые невозможно повлиять.

- Однако, всегда можно попросить помощи у менеджера, заказчика, группы разработчиков. Также, если видно, что что-то будет мешать тестированию, следует проинформировать об этом менеджера и заказчика и добиться понимания ситуации от них, а также согласовать действия, которые будут предприняты в данной ситуации.

- Пример. В процессе подготовки очередного билда программисты столкнулись с трудностями и выпустили билд на два дня позже запланированного срока. В результате график тестирования тоже сместился и у тестировщиков будет меньше времени, чтобы выполнить свою работу. Следует согласовать заранее свои действия в этом случае.

# Риски

- Риск – сочетание вероятности наступления события и последствий, вызванных этим событием.
- Думая о рисках, следует думать о том, какие риски могут возникнуть на проектном уровне. Эти риски на первый взгляд могут не касаться тестирования, но на самом деле их последствия могут напрямую влиять на работу команды тестировщиков.

Пример: В продукте разрабатывается несколько приложений. Для того, чтобы протестировать функциональность приложения А, требуется использовать функциональность приложения В. Разработчики запланировали реализовать сначала функциональность приложения А, а только затем приложения В.



# Секции тестового плана

Секции тестового плана включают в себя:

- Перечень работ;
- Критерии качества и оценка качества процесса;
- Оценка рисков;
- Документация и письма;
- Тестовая стратегия;
- Ресурсы;
- Метрики;
- Расписание;
- Ключевые точки.

# Перечень работ

Сюда включается перечень функциональных областей приложений, которые будут подвергаться тестированию. Здесь же может быть перечень компонентов или функциональности, которые не будут тестироваться по тем или иным причинам. Например, эту функциональность реализует другая фирма. Или в проекте используются какие-то уже готовые компоненты. Если есть какие-либо ограничения тестирования, их тоже можно здесь перечислить.



# Перечень работ

Пример:

В приложении используется визуальный HTML-редактор, который разработан другой фирмой. Ограничение тестирования состоит в том, что мы не собираемся тестировать функциональность самого редактора, поскольку предполагаем, что это сделала компания-разработчик данного редактора. Мы сосредоточимся на тестировании взаимодействия редактора с нашим приложением. Другими словами, будем проводить интеграционное тестирование. Однако, если в процессе интеграционного тестирования обнаружатся какие-либо дефекты в самом редакторе, их следует документировать с соответствующей пометкой, чтобы потом наши разработчики могли разобраться, в действительности ли данные дефекты являются дефектами самого редактора, и, если так, мы сообщим о них компании-производителю данного компонента для последующего исправления.

## Критерии качества и оценка качества процесса

Здесь отражается перечень критериев качества, на основании которых будет приниматься заключение об уровне качества продукта и возможности передачи продукта заказчику. Критерии качества относятся к качеству продукта. В тестовых планах могут быть записаны и критерии качества самого процесса тестирования (и разработки), с целью последующей оценки качества процесса. Это необходимо, чтобы в случае необходимости существовала возможность оценить, насколько грамотно был построен процесс тестирования, были ли какие-либо проблемы, и, если были, разобраться, почему, а также выработать рекомендации по их предотвращению в будущем.

## Оценка рисков

Здесь речь идёт как раз о тех самых рисках (негативных ситуациях), которые могут возникнуть в процессе работы над проектом и помешать (негативно повлиять) на тестирование продукта. Кроме описания самого риска, следует указать вероятность его возникновения и степень влияния на проект. Производная этих двух величин даёт показатель, который может рассматриваться как степень серьёзности риска. Чем выше этот показатель, тем больше внимания нужно уделить этому риску: обдумыванию последствий, составлению плана его предотвращения или выхода из ситуации, если риск наступит. Если риск случился (т.е. ситуация наступила), возникает реальная проблема, которую необходимо решить.

## Документация и письма

В этой секции размещается перечень артефактов (результатов деятельности).

Например:

- тест план;
- тестовые сценарии;
- тестовые автоматические скрипты;
- дефект-репорты;
- отчеты о результатах тестирования.

Также указывается, кто будет высылать данный артефакт, как часто, каким способом, кому и т.д.

# Тестовая стратегия

- Самый большой и один из самых важных разделов плана.
- Здесь расписывается стратегия тестирования, методы и типы тестов, каким образом будет выполняться тестирование, почему именно так и т. д.
- Конкретное содержание этого раздела зависит от фирмы-разработчика, проекта, заказчика и т. д.

# Тестовая стратегия

Этот раздел может включать подразделы:

- Критерии приёмки билдов;
- Методы тестирования;
- Типы тестирования;
- Уровни тестирования;
- Отслеживание ошибок;
- Использование метрик.



# Ресурсы

- **Человеческие ресурсы:** перечень ключевых людей на проекте (менеджер проекта, представители заказчика, лидер команды разработчиков и т.д.), список тестировщиков с их ролями на проекте, а также с зонами ответственности.
- **Аппаратные ресурсы (hardware):** сюда входит перечень тестовых серверов и рабочих станций, инструментов, используемых для тестирования или для вспомогательных работ, описание тестового окружения.
- **Программные ресурсы (software):** операционные системы, СУБД, серверы приложений, веб-серверы и т.д.



# Метрики

**Метрика** – это числовая характеристика, позволяющая оценить тот или иной аспект программного продукта или процесса в целом.

Например: количество дефектов, найденных за неделю в тех или иных областях продукта. Эта метрика позволяет оценить, как много дефектов в той или иной функциональности. Что, в свою очередь может позволить сделать немало выводов или, например, подтолкнуть к разбору причин такого количества багов.

# Метрики

## Правило метрики:

Билд считается неприемлимым, если в нем есть хотя бы один Critical или High баг, либо 5% функционала не протестировано или имеет Medium или Low баги.



# Расписание и ключевые точки

- В этой секции описывается график тестирования в согласовании с графиком выпуска билдов и проектным планом, который разрабатывается менеджером проекта.
- Сюда же включаются основные даты: например, даты окончания промежуточных фаз работы над проектом.
- График тестирования нужен для того, чтобы чётко понимать, когда и что следует делать, ничего не пропустить, ничего не забыть и т.д. Он же упрощает контроль за ходом работ по тестированию, а также позволяет оценить текущую ситуацию, определить, всё ли выполнено из того, что было запланировано.

# Критерии хорошего тестового плана

Тест план должен быть:

- Полным
- Корректным
- Недвусмысленным.



# Критерии хорошего тестового плана

В тест плане должны быть:

- определены цели тестирования, тестовый подход, стратегия, методы, виды тестирования. Запланированный подход должен быть реально выполним.
- установлены реалистичные критерии качества.
- определены критерии прохождения приёмочного теста и условия прекращения тестирования.
- определены все артефакты, подлежащие сдаче, поставке или распространению (заказчику, проекту и т.д.)
- перечислены тестовые ресурсы (люди, оборудование) с указанием ролей, назначений, ответственности.

# Критерии хорошего тестового плана

Также:

- Должно быть определено и описано тестовое оборудование, окружение, программное обеспечение.
- Должен быть определён график тестирования, он должен быть реалистичен и выполним.
- Тест-план должен соответствовать принятому в компании шаблону, если на проекте не решено иначе: например, использовать шаблон заказчика.



# Преимущества хорошего тестового плана

- В условиях постоянного ограничения и нехватки времени, хорошо распланированный, систематизированный подход позволяет достичь лучших результатов работы, а также позволяет обнаруживать большее количество ошибок, чем неорганизованная, плохо распланированная деятельность.
- Тестовый план позволяет управлять процессом тестирования более эффективно.
- Тестовый план позволяет увидеть и понять минимальный уровень тестирования и получить представление об уровне проводимого тестирования каждой области продукта.
- Тестовый план позволяет достичь соглашения между исполнителями, заказчиком и менеджером, о том, каким образом и в какие сроки будет проводиться тестирование.



# Тест план. Закрепляем

## Что надо тестировать?

описание объекта тестирования: системы, приложения, оборудования

## Что будете тестировать?

список функций и описание тестируемой системы и её компонент в отдельности

## 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) в разрезе запланированных фаз разработки

## 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)

...

## Риски и управление ими

поздняя поставка ПО

перебои в работе сервисов третьей стороны

# Тест-план и отчёт о результатах тестирования

**Метрика** — числовая характеристика показателя качества. Может включать описание способов оценки и анализа результата).

**Покрытие** — процентное выражение степени, в которой исследуемый элемент затронут соответствующим набором тест-кейсов.

# Тест-план и отчёт о результатах тестирования

Отчёт о результатах тестирования (test progress report, test summary report) — документ, обобщающий результаты работ по тестированию и содержащий информацию, достаточную для соотнесения текущей ситуации с тест-планом и принятия необходимых управленческих решений.

# Чек-лист

- Упрощенная форма тест-кейса
- Главный принцип чек-листов заключается в том, что каждый тестировщик по-своему проходит их, расширяя тестовый набор своей экспертизой

Проверка	Результат	Комментарии
<b>Операции с файлами</b>	ok	
Создание файла	ok	
Открытие файла	ok	
Сохранение документа	ok	
Печать	ok	
<b>Редактирование файлов</b>	bugs	
Отмена	ok	
Копирование	ok	
Вырезание	ok	
Вставка	ok	
Удаление	ok	
Поиск	fail	<a href="#">bug #123</a>
Поиск с заменой	fail	<a href="#">bug #126</a>
Вставка даты	ok	
<b>Форматирование</b>	ok	
Перенос строки	ok	
Изменение шрифта	ok	
<b>Справка</b>	ok	

# Чек-лист

- Разбивайте приложение на модули (модуль авторизации, модуль настроек и т.д.)
- Используйте «косметику»
- Используйте техники ускорения написания (copy-paste)

# Документирование тестов

- Результатом документирования тестов является **тест-кейс**.
- Набор тест-кейсов – **Test Suite**.

# Определения тест-кейсов

- **IEEE Std 610-1990:**

- «A set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement.»

(«Набор тестовых входных данных, условий выполнения и ожидаемых результатов, разработанных с конкретной целью, такой как проверка некоторого пути выполнения программы или проверка соответствия некоторому требованию.»)

- **IEEE Std 829-1983:**

- «Documentation specifying inputs, predicted results, and a set of execution conditions for a test item.»

(«Документ, определяющий набор входных данных, ожидаемых результатов и условий выполнения теста.»)



# Зачем нужны тест-кейсы

- «Планирование, и только потом – выполнение!» Тест-кейсы дают нам структурированный системный подход, что снижает вероятность пропуска ошибки.
- Тест-кейсы – хороший способ хранения части проектной информации.
- Написание тест-кейсов – один из способов протестировать проектную документацию ещё до выхода первого билда.
- Наличие тест-кейсов значительно ускоряет регрессионное тестирование.
- Тест-кейсы – прекрасный способ быстро ввести в курс дела новичка или сотрудника, только что подключившегося к проекту.
- Имея тест-кейсы, мы можем в любой момент «вспомнить», что мы делали месяц, полгода, год назад.
- Мы можем обмениваться тест-кейсами (и «чек-листами») между проектами.
- Тест-кейсы позволяют легко отслеживать прогресс (X% тестов выполнено, Y% тестов прошло (завалилось), Z% требований покрыто тестами).

# Свойства тест-кейсов

- **Тест-кейсы могут быть:**
  - Специфичными или общими.
  - Простыми или сложными.
  - Независимыми или связанными друг с другом.
  - Позитивными или негативными.

# Специфичность или общность

- Когда все детали прописаны до мелочей, при повторных выполнениях теста всегда будут выполняться строго одни и те же действия, что снижает вероятность обнаружить ошибку.
- Слишком общий тест-кейс сложно выполнять по многим объективным и субъективным причинам, а потому он вполне может остаться невыполненным.
- Однако интеграционные тесты, как правило, бывают более общими, чем иные. Это связано со спецификой интеграционного тестирования.
- Если в тесте прописано много мелких деталей, возрастает время его создания и поддержки.
- Однако недостаток деталей может усложнить работу новичка.

# Простота или сложность

- Рассмотрим на примере. Где в ниже перечисленном простые тест-кейсы, а где – сложные?
- **Набор 1:**
  - 1. Откройте файл «1.txt». Файл открыт.
  - 2. Введите слово «Дом». Появляется слово «Дом.
  - 3. Сохраните файл. Кнопка «Сохранить» становится неактивной.
- **Набор 2:**
  - 1. В документе размером более 100 Мб создайте таблицу 100×100, в ячейку 50×50 вставьте картинку размером 30 Мб, применив к ней функцию «Авторасположение». Проверьте результат.

# Простота или сложность

- Простые тесты оперируют за раз одним объектом.
- Каковы преимущества простых тест-кейсов?
  - Их легко выполнять.
  - Они понятны новичкам.
  - Они упрощают диагностику ошибки.
  - Они делают наличие ошибки очевидным.
- Каковы преимущества сложных тест-кейсов?
  - Больше шансов что-то сломать.
  - Пользователи, как правило, используют сложные сценарии.
  - Программисты сами редко проверяют такие варианты.
- Следует постепенно повышать сложность тестов.

# Независимость или связанность

- Каковы *преимущества* независимого самостоятельного тест-кейса?
  - Его легко и просто выполнить.
  - Такие тесты могут работать даже после краха приложения на других тестах.
  - Такие тесты можно группировать любым образом и выполнять в любом порядке.
- Каковы *преимущества* наборов тесно связанных тестов?
  - Они имитируют работу реальных пользователей.
  - Они удобны для интеграционного тестирования.
  - Они удобны для разбиения на части тестов с большим количеством шагов.
  - Следующий в наборе тест использует данные и состояние приложения, подготовленные предыдущим.
- Промышленным стандартом являются независимые тесты. Использование сценариев не запрещено, но не следует делать их слишком длинными.

# Позитивность или негативность

**Позитивные тесты** проверяют, что приложение делает то, на что оно рассчитано (т.е. такие тесты используют корректные данные и условия выполнения).

**Негативные тесты** проверяют работу приложения в нестандартных условиях (при получении некорректных данных или команд или при работе в некорректных условиях).

- Обе разновидности тестов важны и нужны, однако следует помнить последовательность их разработки и выполнения:
  1. Простые позитивные.
  2. Простые негативные.
  3. Сложные позитивные.
  4. Сложные негативные.



# Что должен содержать тест-кейс

- Идентификатор теста (id)
- Связанное с тестом требование (related requirement)
- Краткое заглавие теста (title)
- Модуль и подмодуль приложения, к которым относится тест (module, submodule)
- Приоритет теста (priority: smoke, critical, extended; A, B, C, D)
- Исходные данные, необходимые для теста (initial data) (обычно включается в шаги выполнения)
- Шаги для выполнения теста (steps)
- Ожидаемые результаты (expected results)
- Поле для пометки, прошёл тест или нет (status)
- Последний полученный актуальный результат (actual result), связанный с тестом баг (если есть) (related bug)
- Указать автора теста (author), время последнего выполнения теста (last time run) (часто эта информация указывается в заголовке файла)

# Документирование тестов

Приоритет	Связанное с тестом требование	Заглавие (суть) теста	Ожидаемый результат по каждому шагу
UG_U A R97	Галерея	Загрузка файла	Галерея, загрузка файла, имя со спецсимволами Приготовление: создать непустой файл с именем #\$\$%^&.jpg
Идентификатор	Модуль и подмодуль	Шаги	1. Появляется окно загрузки картинки 2. Появляется диалоговое окно браузера выбора файла для загрузки 3. Имя выбранного файла появляется в поле «Файл» 4. Диалоговое окно файла закрывается, в поле «Файл» появляется полное имя файла 5. Выбранный файл появляется в списке файлов галереи
		Исходные данные, необходимые для выполнения теста	
		Шаги	

# Рекомендации по написанию тест-кейсов

- Начинайте с простых очевидных тестов.
- Затем переходите к более сложным тестам.
- Помните о граничных условиях.
- Если остаётся время, занимайтесь исследовательским тестированием.

# Язык написания тест-кейсов

- Используйте активный залог: («open», «paste», «click»). В русском языке используйте безличную форму: «открыть» (вместо «откройте»)
- Описывайте поведение системы: «появляется окно...», «приложение закрывается»
- Используйте простой технический стиль
- **ОБЯЗАТЕЛЬНО** указывайте **ТОЧНЫЕ** названия всех элементов приложения
- Не объясняйте базовые понятия работы с ОС

# Критерии хорошего тест-кейса

- Обладает высокой вероятностью обнаружения ошибки.
- Не выполняет ненужных действий.
- Не является избыточным по отношению к другим тестам.
- Исследует соответствующую («ту, которую надо») область приложения.
- Позволяет легко диагностировать ошибку.
- Делает обнаруженную ошибку очевидной.
- Независим (каждый тест-кейс – это индивидуальный сценарий с точкой входа и точкой выхода).

# Тестовые набор (Test Suite)

**Тестовый набор** – набор тестов (тест-кейсов), собранных в последовательность для достижения некоторой цели. Хороший тестовый сценарий всегда следует некоторой логике, например: типичному использованию приложения, удобству тестирования, распределению функций по модулям и т.д.

# Рекомендации по написанию тестовых наборов

- Пишите набор для отдельной части приложения.
- Пишите отдельно набор для Smoke и Critical Path тестов.
- Постепенно повышайте сложность тестов.
- Организуйте сценарий логично.
- Используйте один тест для ОДНОЙ проверки.
- Помните, что заголовки тестов отражают их суть. Правильно формулируйте и оформляйте заголовки.
- Помните о необходимых приготовлениях к тесту. Описывайте их.
- Не повторяйте в нескольких тестах одни и те же шаги.
- Старайтесь избегать похожих тестов (таких, в которых набор шагов и ожидаемых результатов визуально кажется одинаковым).



# Техники ускорения написания тестов

- Copy-paste.
- Если по ходу разработки тестов возникают вопросы, пишите их прямо в документ с тестами, помечая красным цветом.
- Используйте т.н. «косметику» (жирный, подчёркнутый, наклонный шрифт, разные цвета т.д.) Это значительно повышает читаемость документа.
- По-максимуму используйте возможности ПО, в котором вы разрабатываете тесты (группировки, фильтры, ссылки и т.д.)
- Если вы пишете тесты в файле, обязательно прописывайте в самом файле историю его изменения.

# Шаги разработки тестовых сценариев

- Сбор информации (требования, мок-апы и т. д.)
- Разделение приложения на модули
- Написание чек-листов
- Написание тест-кейсов

# Шаги разработки тестовых сценариев

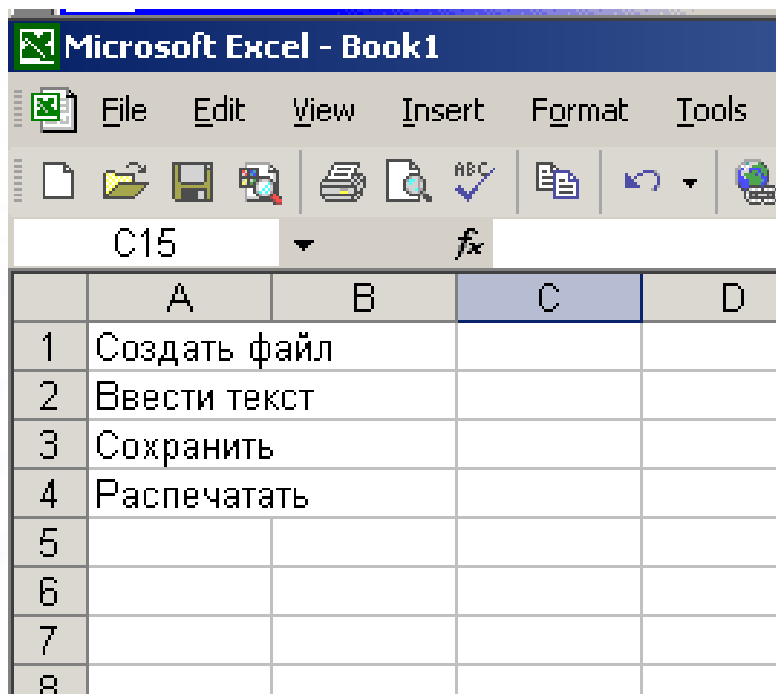
- Начинайте **как можно раньше**, ещё до выхода первого билда.
- Разбивайте приложение на **отдельные части/модули**.
- Для каждой области/модуля **пишите чек-лист**.
- Пишите **вопросы**, уточняйте детали, добавляйте «**косметику**», используйте **copy-paste**.
- Получите **рецензию** коллег-тестировщиков, разработчиков, заказчиков.
- **Обновляйте тесты**, как только обнаружили ошибку или изменилась функциональность.

# Последовательность разработки и выполнения тестов

- Простые позитивные.
- Простые негативные.
- Сложные позитивные.
- Сложные негативные.

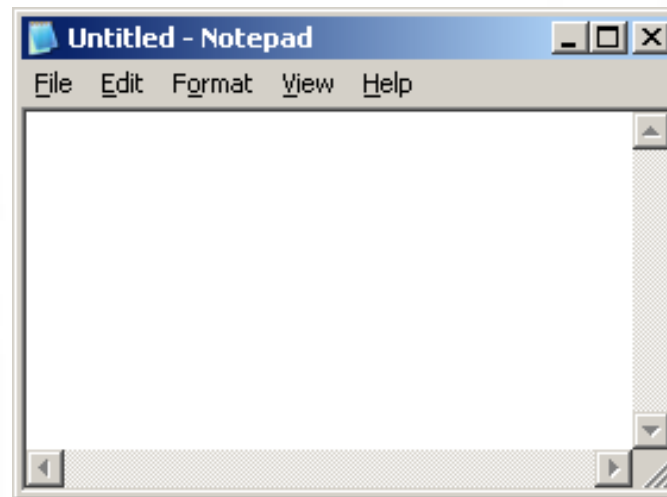
# Учимся составлять первый тест-кейс

- Что такое notepad?
- Какие функции для него важны?



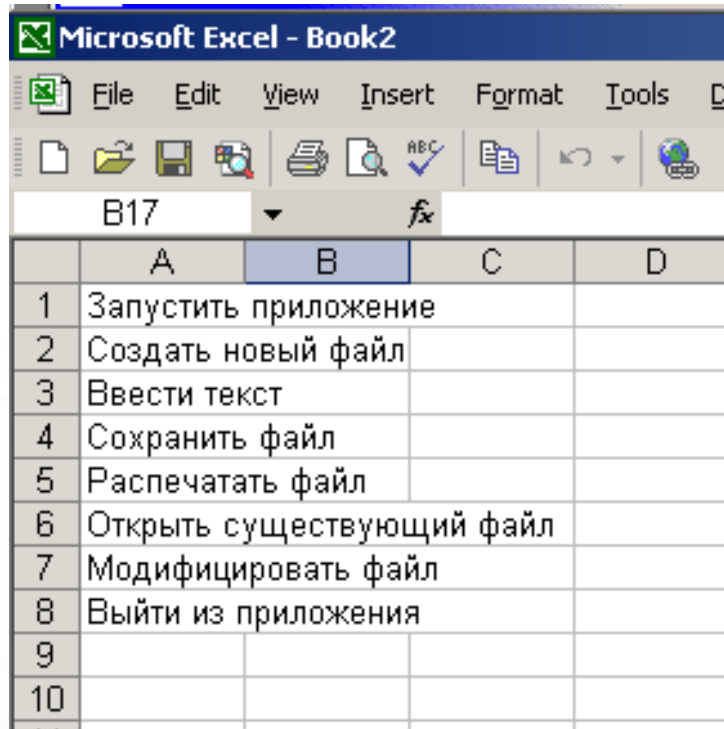
The screenshot shows the Microsoft Excel interface with a table containing test steps. The active cell is C15, and the formula bar is empty. The table has 8 rows and 4 columns (A, B, C, D). The first four rows contain the steps: 'Создать файл', 'Ввести текст', 'Сохранить', and 'Распечатать'.

	A	B	C	D
1	Создать файл			
2	Ввести текст			
3	Сохранить			
4	Распечатать			
5				
6				
7				
8				



Что еще?

# Учимся составлять первый тест-кейс



The screenshot shows a Microsoft Excel window titled "Microsoft Excel - Book2". The menu bar includes File, Edit, View, Insert, Format, and Tools. The toolbar contains icons for file operations and editing. The active cell is B17. The following table is displayed in the worksheet:

	A	B	C	D
1	Запустить приложение			
2	Создать новый файл			
3	Ввести текст			
4	Сохранить файл			
5	Распечатать файл			
6	Открыть существующий файл			
7	Модифицировать файл			
8	Выйти из приложения			
9				
10				

- Итак, вот наш Smoke test

Перенесём его в шаблон  
для разработки тестов.

# Учимся составлять первый тест-кейс

- Фактически, это – чек-лист. И сами пункты грамотно сформированного чек-листа – готовые заголовки тест-кейсов.

Microsoft Excel - Шаблон для разработки тестов v2

File Edit View Insert Format Tools Data Window Help Adobe PDF

100% Arial 10

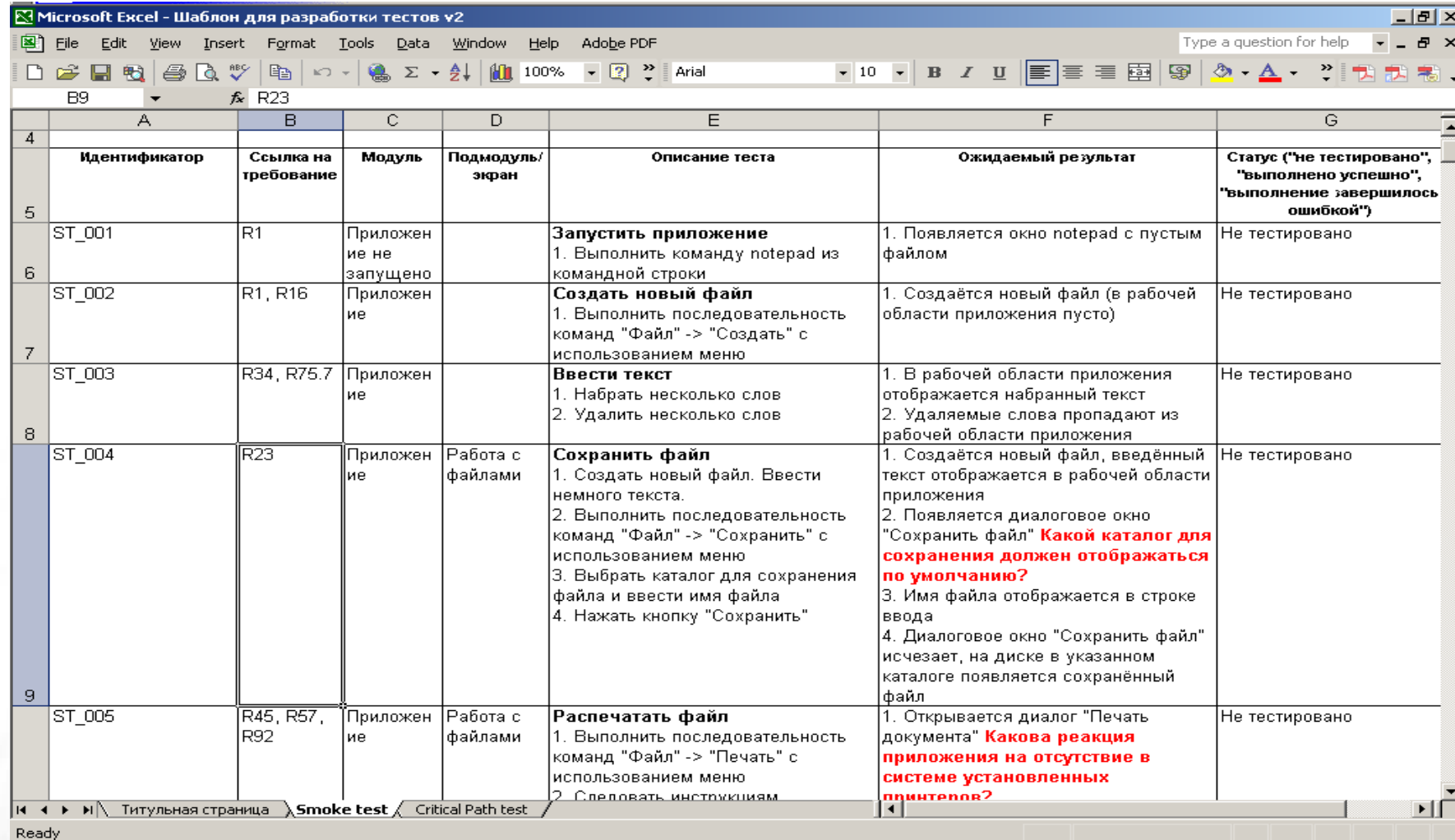
A6 fx

	A	B	C	D	E	F
1	<b>"Notepad"</b> Функциональные тест-кейсы					
2	Тестировал(и):					
3	Дата(даты) тестирования:			ОС:		
4	Идентификатор	Ссылка на требование	Модуль	Подмодуль/экран	Описание теста	Ожидаемый результат
5						
6					Запустить приложение	
7					Создать новый файл	
8					Ввести текст	
9					Сохранить файл	
10					Распечатать файл	
11					Открыть существующий файл	
12					Модифицировать файл	
13					Выйти из приложения	
14						
15						



# Учимся составлять первый тест-кейс

- Когда мы распишем наши тесты по правилам, Smoke Test примет следующий вид:

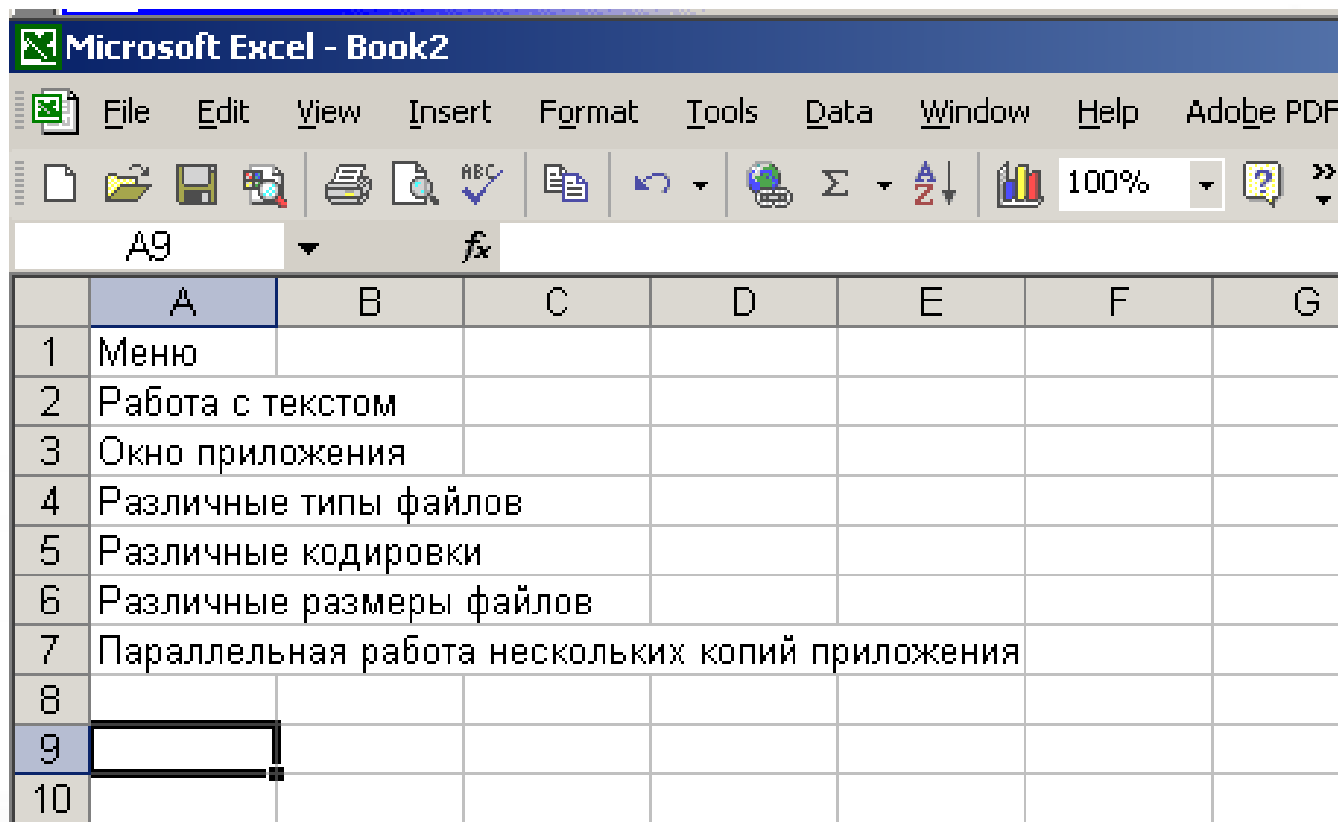


The screenshot shows a Microsoft Excel window titled "Шаблон для разработки тестов v2". The spreadsheet contains a table with 7 columns: A (Идентификатор), B (Ссылка на требование), C (Модуль), D (Подмодуль/экран), E (Описание теста), F (Ожидаемый результат), and G (Статус). The table lists five test cases (ST\_001 to ST\_005) with their respective requirements, modules, descriptions, expected results, and status. The status for all listed test cases is "Не тестировано".

	A	B	C	D	E	F	G
	Идентификатор	Ссылка на требование	Модуль	Подмодуль/экран	Описание теста	Ожидаемый результат	Статус ("не тестировано", "выполнено успешно", "выполнение завершилось ошибкой")
5	ST_001	R1	Приложение не запущено		<b>Запустить приложение</b> 1. Выполнить команду notepad из командной строки	1. Появляется окно notepad с пустым файлом	Не тестировано
6	ST_002	R1, R16	Приложение		<b>Создать новый файл</b> 1. Выполнить последовательность команд "Файл" -> "Создать" с использованием меню	1. Создаётся новый файл (в рабочей области приложения пусто)	Не тестировано
7	ST_003	R34, R75.7	Приложение		<b>Ввести текст</b> 1. Набрать несколько слов 2. Удалить несколько слов	1. В рабочей области приложения отображается набранный текст 2. Удаляемые слова пропадают из рабочей области приложения	Не тестировано
8	ST_004	R23	Приложение	Работа с файлами	<b>Сохранить файл</b> 1. Создать новый файл. Ввести немного текста. 2. Выполнить последовательность команд "Файл" -> "Сохранить" с использованием меню 3. Выбрать каталог для сохранения файла и ввести имя файла 4. Нажать кнопку "Сохранить"	1. Создаётся новый файл, введённый текст отображается в рабочей области приложения 2. Появляется диалоговое окно "Сохранить файл" <b>Какой каталог для сохранения должен отображаться по умолчанию?</b> 3. Имя файла отображается в строке ввода 4. Диалоговое окно "Сохранить файл" исчезает, на диске в указанном каталоге появляется сохранённый файл	Не тестировано
9	ST_005	R45, R57, R92	Приложение	Работа с файлами	<b>Распечатать файл</b> 1. Выполнить последовательность команд "Файл" -> "Печать" с использованием меню 2. Следовать инструкциям	1. Открывается диалог "Печать документа" <b>Какова реакция приложения на отсутствие в системе установленных принтеров?</b>	Не тестировано

# Учимся составлять первый тест-кейс

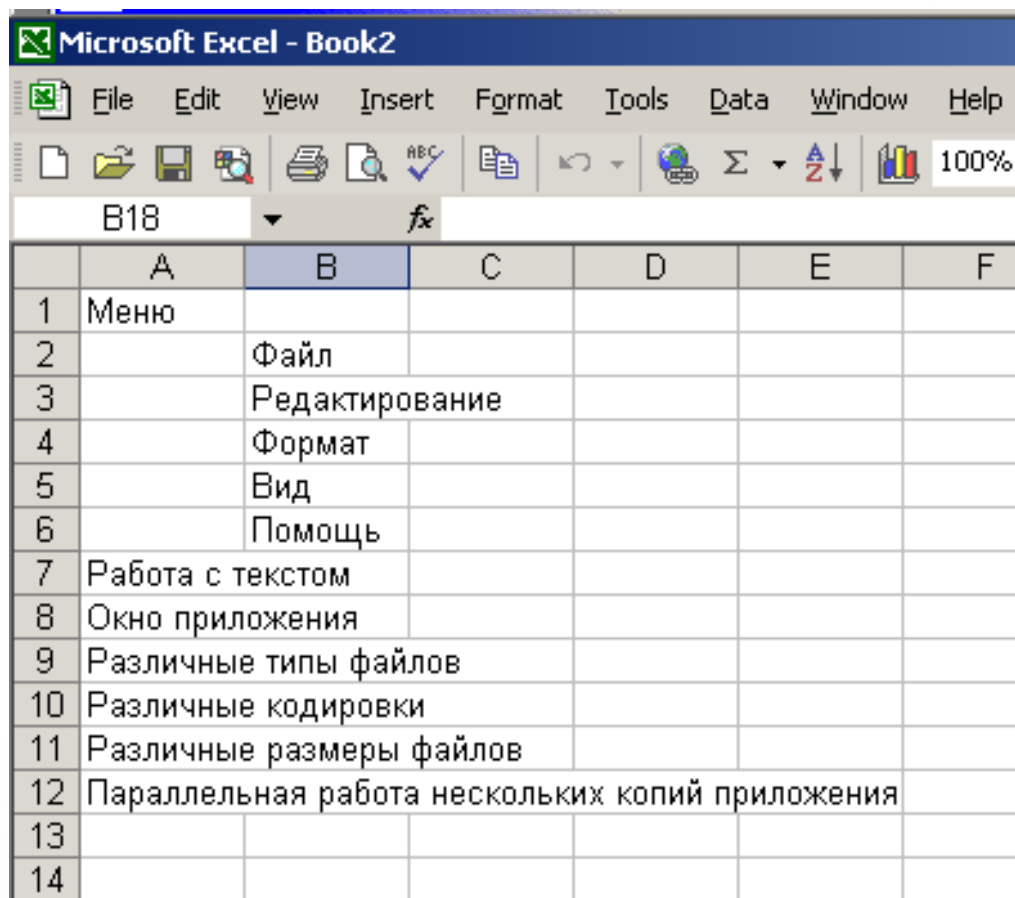
- Аналогичным образом начинаем и продолжаем работать с тестом критического пути:



Microsoft Excel - Book2							
File Edit View Insert Format Tools Data Window Help Adobe PDF							
A9 fx 100%							
	A	B	C	D	E	F	G
1	Меню						
2	Работа с текстом						
3	Окно приложения						
4	Различные типы файлов						
5	Различные кодировки						
6	Различные размеры файлов						
7	Параллельная работа нескольких копий приложения						
8							
9							
10							

# Учимся составлять первый тест-кейс

- Детализируем чек-лист:

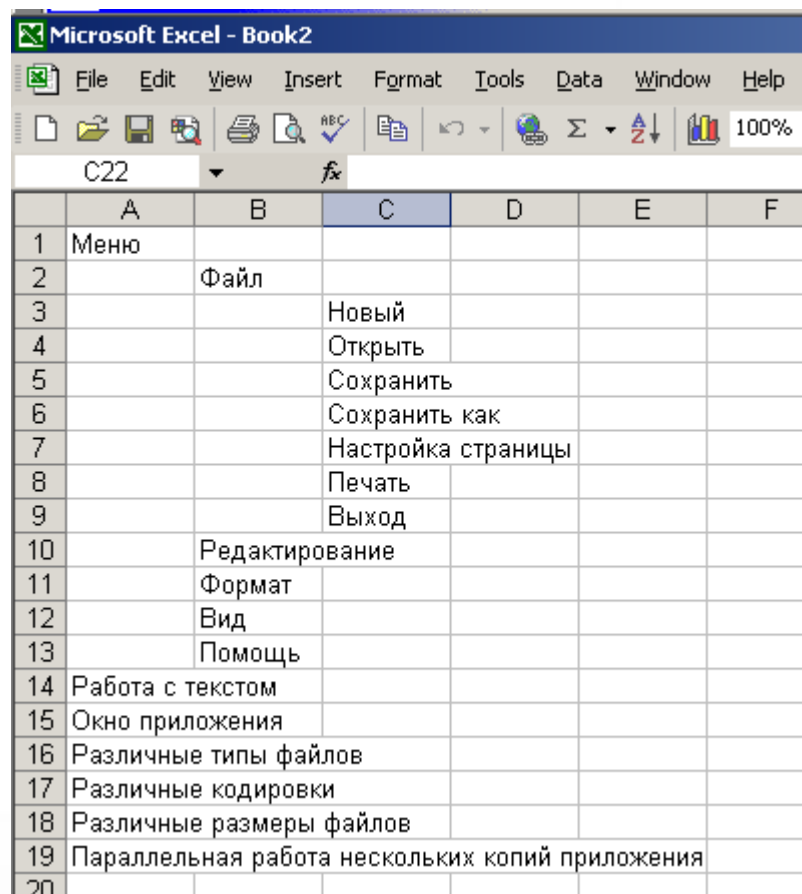


The screenshot shows the Microsoft Excel interface with a menu structure detailed in a spreadsheet. The menu items are listed in column A, and their sub-items are listed in column B. The spreadsheet is titled 'Microsoft Excel - Book2' and has a standard menu bar (File, Edit, View, Insert, Format, Tools, Data, Window, Help) and a toolbar. The active cell is B18.

	A	B	C	D	E	F
1	Меню					
2		Файл				
3		Редактирование				
4		Формат				
5		Вид				
6		Помощь				
7	Работа с текстом					
8	Окно приложения					
9	Различные типы файлов					
10	Различные кодировки					
11	Различные размеры файлов					
12	Параллельная работа нескольких копий приложения					
13						
14						

# Учимся составлять первый тест-кейс

- Продолжаем детализацию до тех пор, пока не получим логичный и достаточный набор тестов. После этого переносим его в шаблон и работаем аналогично тому, как мы делали это при разработке Smoke Test.



The screenshot shows a Microsoft Excel spreadsheet titled "Microsoft Excel - Book2". The spreadsheet contains a menu structure organized into columns A through F. Column A contains the main menu items, Column B contains sub-items, and Column C contains further sub-items. The menu structure is as follows:

	A	B	C	D	E	F
1	Меню					
2		Файл				
3			Новый			
4			Открыть			
5			Сохранить			
6			Сохранить как			
7			Настройка страницы			
8			Печать			
9			Выход			
10		Редактирование				
11		Формат				
12		Вид				
13		Помощь				
14	Работа с текстом					
15	Окно приложения					
16	Различные типы файлов					
17	Различные кодировки					
18	Различные размеры файлов					
19	Параллельная работа нескольких копий приложения					
20						

**Благодарю за внимание**