

Лекция 2

- Принципы тестирования
- Жизненный цикл ПО
- Модели разработки ПО
- Жизненный цикл тестирования
- Классификация методов тестирования

Принципы тестирования

Тестирование показывает наличие дефектов, а не их отсутствие

Очень сложно обнаружить нечто, относительно чего мы не знаем — ни «где оно», ни «как оно выглядит», ни даже «существует ли оно вообще». Так как не существует физической возможности проверить поведение сложного программного продукта во всех возможных ситуациях и условиях, тестирование не может гарантировать, что в той или иной ситуации, при стечении тех или иных обстоятельств дефект не возникнет. Тестирование может проверить наиболее вероятные, востребованные ситуации и обнаружить дефекты при их возникновении. Такие дефекты будут устранены, что ощутимо повысит качество продукта, но по-прежнему не гарантирует от возникновения проблем в оставшихся, не проверенных ситуациях и условиях

Исчерпывающее тестирование невозможно

Даже для одного простого поля для ввода имени пользователя может существовать порядка 2.4^{32} позитивных проверок и бесконечное количество негативных проверок. Потому невозможно протестировать программный продукт полностью, «исчерпывающе».

Однако, из этого не следует, что тестирование как таковое не является эффективным. Вдумчивый анализ требований, учёт рисков, расстановка приоритетов, анализ предметной области, моделирование, работа с конечными пользователями, применение специальных техник тестирования — эти и многие другие подходы позволяют выявить те области или условия эксплуатации продукта, которые требуют особенно тщательной проверки.

Принципы тестирования

Тестирование тем эффективнее, чем раньше оно выполняется

Раннее тестирование помогает устранить или сократить дорогостоящие изменения. У данного принципа есть прекрасная аналогия из обычной повседневной жизни. Представьте, что вы собираетесь в поездку и продумываете список вещей, которые необходимо взять с собой. На стадии обдумывания добавить, изменить, удалить любой пункт в этом списке не стоит ничего. На стадии поездки по магазинам для закупки необходимого недоработки в списке уже могут привести к необходимости повторной поездки в магазин. На стадии отправки на место назначения недоработки в списке вещей явно приведут к ощутимой потере нервов, времени и денег. А если фатальный недостаток списка вещей выяснится только по прибытии, может так оказаться, что вся поездка потеряла смысл.

Кластеризация дефектов

Дефекты как правило группируются в какой-то «проблемной» области приложения. Группировка дефектов по какому-то явному признаку является хорошим поводом к продолжению исследования данной области программного продукта: скорее всего, именно здесь будет обнаружено ещё больше дефектов. Обнаружение подобных тенденций к кластеризации (и особенно поиск глобальной первопричины) часто требует от тестировщиков определённых знаний и опыта, но если такой «кластер» выявлен — это позволяет ощутимо минимизировать усилия и при этом существенно повысить качество приложения.

Принципы тестирования

Парадокс пестицида

Название данного принципа происходит от общеизвестного явления в сельском хозяйстве: если долго распылять один и тот же пестицид на посевы, у насекомых вскоре вырабатывается иммунитет, что делает пестицид неэффективным.

Проявляется в повторении одних и тех же (или просто однотипных) проверок снова и снова: со временем эти проверки перестанут обнаруживать новые дефекты. Чтобы преодолеть парадокс пестицида, необходимо регулярно пересматривать и обновлять тест-кейсы, разнообразить подходы к тестированию, применять различные техники тестирования, смотреть на ситуацию «свежим взглядом».

Тестирование зависит от контекста

Набор характеристик программного продукта влияет на глубину тестирования, используемый набор техник и инструментов, принципы организации работы тестировщиков и т.д.

Отсутствие дефектов — не самоцель

Программный продукт должен не только быть избавлен от дефектов настолько, насколько это возможно, но и удовлетворять требованиям заказчика и конечных пользователей.

Именно понимание контекста продукта и потребностей пользователей позволяет тестировщикам выбрать наилучшую стратегию и добиться наилучшего результата.

ЖИЗНЕННЫЙ ЦИКЛ ПО

Это период времени, который начинается с

момента принятия решения о необходимости
создания программного продукта

и заканчивается

в момент его полного изъятия из эксплуатации

Software Development Life Cycle (SDLC) – это жизненный цикл программного обеспечения из набора типовых этапов

1. Планирование.
2. Анализ
3. Проектирование
4. Разработка
5. Тестирование и развертывание
6. Поддержка и сопровождение



МОДЕЛИ РАЗРАБОТКИ ПО

Модель разработки ПО – это структура, систематизирующая различные виды проектной деятельности, их взаимодействие и последовательность в процессе разработки ПО

Выбор той или иной модели зависит от масштаба и сложности проекта, предметной области, доступных ресурсов и множества других факторов.

- ✓ Водопадная модель
- ✓ V-модель
- ✓ Итерационная модель
- ✓ Спиральная модель

ВОДОПАДНАЯ МОДЕЛЬ

Водопадная модель (waterfall model) сейчас представляет скорее исторический интерес, т.к. в современных проектах практически неприменима (исключением могут быть крупные проекты со стабильными требованиями).



С точки зрения тестирования эта модель плоха тем, что тестирование в явном виде появляется здесь лишь с середины развития проекта, достигая своего максимума в самом конце.

!!! Забавное описание истории развития и заката водопадной модели было создано Максимом Дорофеевым в виде слайдката «The Rise And Fall Of Waterfall», который можно посмотреть в его ЖЖ.

<http://cartmendum.livejournal.com/44064.html>



V-МОДЕЛЬ

Тестирование появляется на ранних стадиях развития проекта. Это позволяет минимизировать риски, а также обнаружить и устранить множество потенциальных проблем.



ИТЕРАЦИОННАЯ ИНКРЕМЕНТАЛЬНАЯ МОДЕЛЬ



СПИРАЛЬНАЯ МОДЕЛЬ

Проработка целей,
альтернатив и
ограничений

Анализ
рисков и
прототипиро
вание

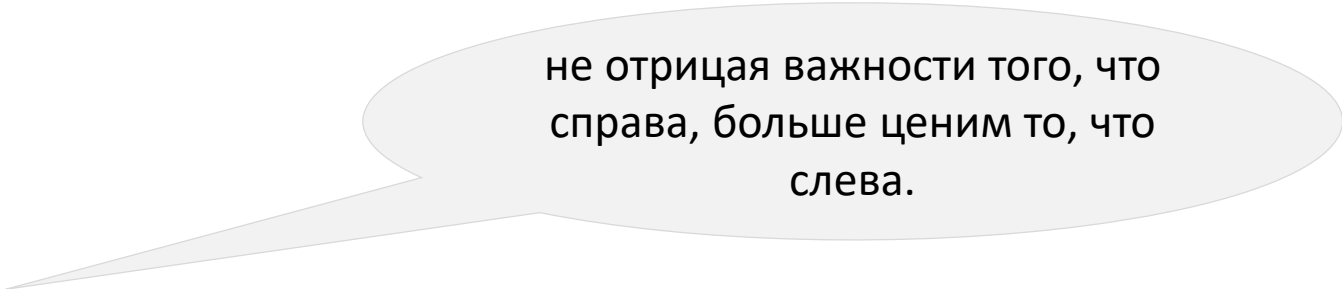


С точки зрения тестирования и управления качеством повышенное внимание рискам является ощутимым преимуществом при использовании спиральной модели для разработки концептуальных проектов, в которых требования могут многократно меняться по ходу выполнения проекта.

AGILE MODEL

Гибкая модель (agile model) представляет собой совокупность различных подходов к разработке ПО и базируется на т.н. «agile-манифесте»

AGILE МАНИФЕСТ:



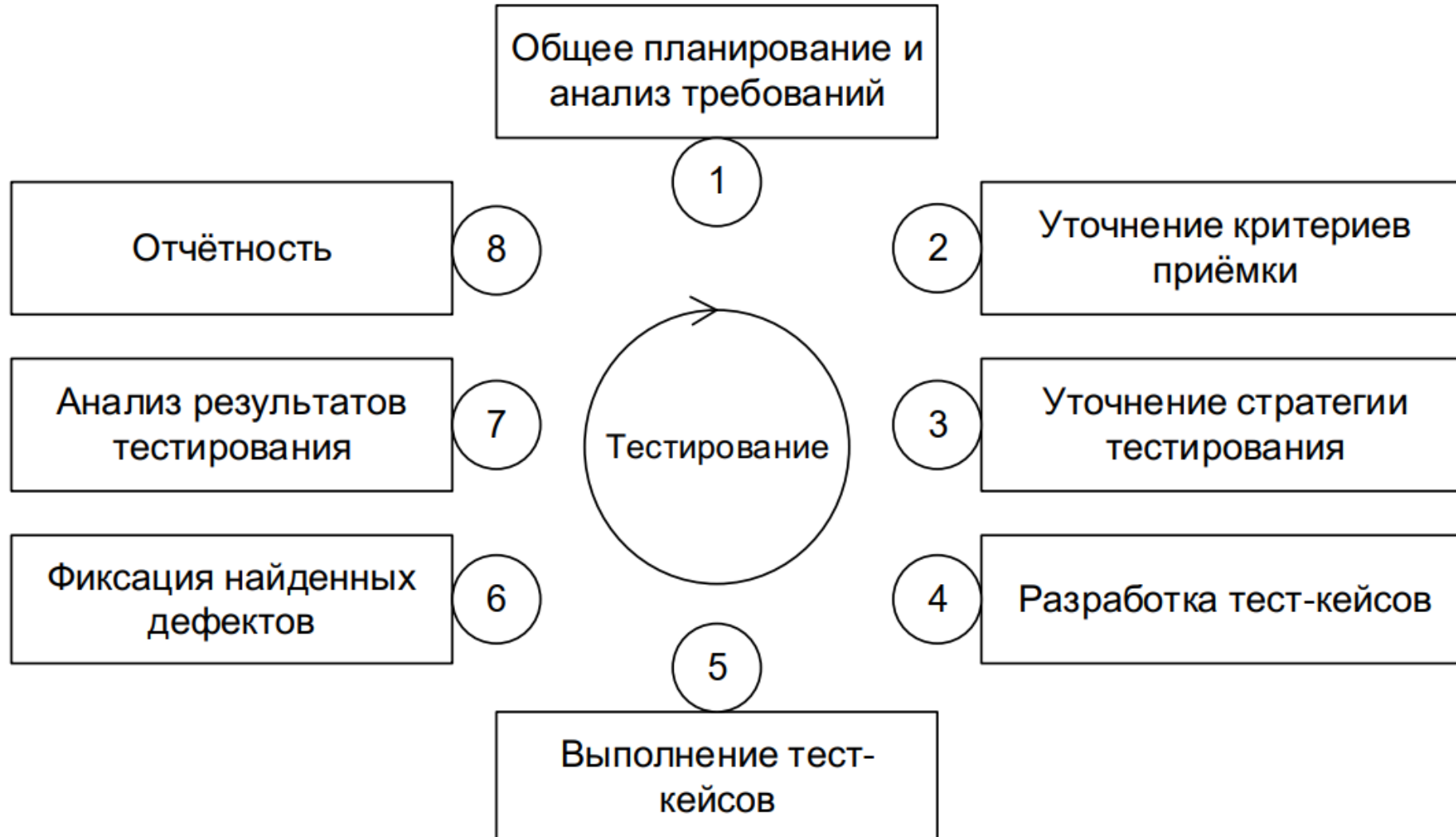
не отрицая важности того, что
справа, больше ценим то, что
слева.

- ✓ **Люди и взаимодействие** важнее процессов и инструментов.
- ✓ **Работающий продукт** важнее исчерпывающей документации.
- ✓ **Сотрудничество с заказчиком** важнее согласования условий контракта.
- ✓ **Готовность к изменениям** важнее следования первоначальному плану.

Сравнение моделей разработки ПО

| Модель | Тестирование |
|-----------------------------|---|
| Водопадная | С середины проекта. |
| V-образная | На переходах между стадиями |
| Итерационная инкрементальна | <ul style="list-style-type: none">• В определённые моменты итераций.• Повторное тестирование (после доработки) уже проверенного ранее. |
| Спиральная | |
| Гибкая | В определённые моменты итераций и в любой необходимый момент. |

Жизненный цикл тестирования



Стадия 1 (общее планирование и анализ требований) Что нам предстоит тестировать; как много будет работы; какие есть сложности; всё ли необходимое у нас есть и т.п.

Стадия 2 (уточнение критериев приёмки) позволяет сформулировать или уточнить метрики и признаки возможности или необходимости начала тестирования (entry criteria), приостановки (suspension criteria) и возобновления (resumption criteria) тестирования, завершения или прекращения тестирования (exit criteria).

Стадия 3 (уточнение стратегии тестирования) планирование, но уже на локальном уровне: рассматриваются и уточняются те части стратегии тестирования (test strategy), которые актуальны для текущей итерации.

Стадия 4 (разработка тест-кейсов) посвящена разработке, пересмотру, уточнению, доработке, переработке и прочим действиям с тест-кейсами, наборами тесткейсов, тестовыми сценариями и иными артефактами, которые будут использоваться при непосредственном выполнении тестирования.

Стадия 5 (выполнение тест-кейсов) и стадия 6 (фиксация найденных дефектов) тесно связаны между собой и фактически выполняются параллельно: дефекты фиксируются сразу по факту их обнаружения в процессе выполнения тест-кейсов. Однако зачастую проводится явно выделенная стадия уточнения, на которой все отчёты о дефектах рассматриваются повторно с целью формирования единого понимания проблемы и уточнения таких характеристик дефекта, как важность и срочность.

Стадия 7 (анализ результатов тестирования) и стадия 8 (отчётность) также тесно связаны между собой и выполняются практически параллельно. Формулируемые на стадии анализа результатов выводы напрямую зависят от плана тестирования, критериев приёмки и уточнённой стратегии, полученных на стадиях 1, 2 и 3.

Полученные выводы оформляются на стадии 8 и служат основой для стадий 1, 2 и 3 следующей итерации тестирования. Таким образом, цикл замыкается.

ПРОЦЕСС ТЕСТИРОВАНИЯ

- ✓ Планирование и управление
- ✓ Анализ и проектирование
- ✓ Внедрение и реализация
- ✓ Оценка критериев выхода и создание отчетов
- ✓ Действия по завершению тестов

ПЛАНИРОВАНИЕ ТЕСТИРОВАНИЯ ПО

- ✓ Анализ требований;
- ✓ Определение целей тестирования;
- ✓ Определение общего подхода к тестированию (уровни тестирования, виды тестирования, критерия входа);
- ✓ Интегрирование с разработкой ПО (требования, архитектура, дизайн, разработка, тестирование, релиз);
- ✓ Решение, какие роли нужны для выполнения тестирования, когда и как проводить тестирование и как оценивать результаты;
- ✓ Составление графика тестирования;
- ✓ Определение шаблонов для тестовой документации;
- ✓ Выбор метрик для мониторинга и контроля подготовки и проведения тестирования, исправления дефектов, проблем и рисков.

ТЕХНИКИ ТЕСТИРОВАНИЯ ТРЕБОВАНИЙ

- ✓ Взаимный просмотр
- ✓ Вопросы
- ✓ Тест-кейсы и чек-листы
- ✓ Исследование поведения системы
- ✓ Рисунки
- ✓ Прототипы

КРИТЕРИИ КАЧЕСТВА ТРЕБОВАНИЙ

- ✓ Корректность
- ✓ Недвусмысленность
- ✓ Полнота
- ✓ Непротиворечивость
- ✓ Упорядоченность по важности
- ✓ Проверяемость
- ✓ Модифицируемость
- ✓ Трассируемость

МЕТОДЫ ТЕСТИРОВАНИЯ ТРЕБОВАНИЙ

- ✓ Метод просмотра
- ✓ Метод экспертизы
- ✓ Метод составления вариантов
тестирования

Качества хороших требований

- ✓ Работоспособность сценариев
- ✓ Полнота описания
- ✓ Внимание обязательным пунктам
- ✓ Актуальность описания
- ✓ Адаптированность к спешке
- ✓ Адаптированность к настроению
- ✓ Структурированность
- ✓ Наличие указаний на необратимость действий
- ✓ Указание на ожидаемый результат
- ✓ Описание последствий отсутствия действий пользователя

Качества хороших требований (продолжение)

- ✓ Ясность изложения
- ✓ Логика и согласованность
- ✓ Последовательность изложения
- ✓ Орфография, синтаксис, пунктуация
- ✓ Описание настроек по умолчанию
- ✓ Адаптированность к аудитории
- ✓ Атомарность
- ✓ Адаптированность к квалификации

Оформление вопросов к требованиям

- ✓ Вопросы пишите короткими и простыми;
- ✓ Вопрос не должен содержать «или»;
- ✓ Формулируйте вопрос так, чтобы ответ на него был максимально коротким;
- ✓ Обдумывайте ответ, который можно получить;
- ✓ Предлагая улучшения, подкрепляйте их фактами и весомыми доводами;

ПРИМЕРЫ

Требование

Приложение должно быстро запускаться.

«Плохие» вопросы

«Насколько быстро?»

(Рискуете получить ответы в стиле «очень быстро», «максимально быстро»).

«А если не получится быстро?»

(Этим вы рискуете удивить или разозлить заказчика.)

«Всегда?»

(«Да, всегда». Хм, а вы ожидали другого ответа?)

ПРИМЕРЫ

Требование

Приложение должно быстро запускаться.

«Хорошие» вопросы

«Каково максимально допустимое время запуска приложения?»

«Допускается ли фоновая загрузка отдельных компонентов приложения?»

«Что является критерием того, что приложение закончило запуск?»

ПРИМЕРЫ

Требование

Опционально должен поддерживаться экспорт документов в формат PDF.

«Плохие» вопросы

«Любых документов?»

(Ответы «да, любых» или «нет, только открытых» вам всё равно не помогут.)

«В PDF какой версии должен производиться экспорт?»

(Вопрос хороший, но он не даёт понять, что имелось в виду под «опционально».)

«Зачем?»

(«Нужно!» Именно так хочется ответить, если вопрос не раскрыт полностью.)

ПРИМЕРЫ

Требование

Опционально должен поддерживаться экспорт документов в формат PDF.

«Хорошие» вопросы

«При каких условиях будет доступен экспорт в PDF?»

«Для каких документов будет поддерживаться экспорт?»

«Какие действия пользователю необходимо выполнить, чтобы экспортировать документ в PDF?»

ПРИМЕРЫ

Требование

Если дата события не указана, она выбирается автоматически.

«Плохие» вопросы

«А если указана?»

(То она указана. Логично, не так ли?)

«А если дату невозможно выбрать автоматически?»

(Сам вопрос интересен, но без пояснения причин невозможности звучит как издёвка.)

«А если у события нет даты?»

(Автор вопроса, скорее всего, хотел уточнить, обязательно ли это поле. Но из самого требования видно, что обязательно.)

ПРИМЕРЫ

Требование

Если дата события не указана, она выбирается автоматически.

«Хорошие» вопросы

«Возможно, имелось в виду, что дата **генерируется** автоматически, а не **выбирается**?

Если «Да», то по какому алгоритму она генерируется?

Если «Нет», то из какого набора выбирается дата и как генерируется этот набор?»

Пример оформления вопросов

Системные характеристики

- СХ-1: Приложение является консольным.
- СХ-2: Для работы приложение использует интерпретатор PHP. ???
- СХ-3: Приложение является кроссплатформенным.

Пользовательские требования

- Также см. диаграмму вариантов использования.
- ПТ-1: Запуск и остановка приложения.
 - ПТ-1.1: Запуск приложения производится из консоли командой "PHP-php converter.php параметры".
 - ПТ-1.2: Остановка приложения производится выполнением команды Ctrl+C в окне консоли, из которого было запущено приложение.
- ПТ-2: Конфигурирование приложения.
 - ПТ-2.1: Конфигурирование приложения сводится к указанию путей в файловой системе.
 - ПТ-2.2: Целевой кодировкой является UTF8.
- ПТ-3: Просмотр журнала работы приложения.
 - ПТ-3.1: В процессе работы приложение должно выводить журнал своей работы в консоль и лог-файл.
 - ПТ-3.2: При первом запуске приложения лог-файл создаётся, а при последующих – дописывается.



Author

1) Какая минимальная версия интерпретатора PHP поддерживается приложением?
2) Существует ли некая специфика настройки интерпретатора PHP для корректной работы приложения?



Author

Должна ли в руководстве пользователя быть описана процедура установки и настройки интерпретатора PHP?

Author

Formatted: Russian



Author

Какие ОС должны поддерживаться? В чём цель кроссплатформенности?



Author

Какие параметры передаются скрипту при запуске? Какова реакция скрипта на:

- Отсутствие параметров.
- Неверное количество параметров.
- Неверные значения каждого из параметров.



Author

Путей к чему?

Ошибки при анализе требований

Более полный список см. в книге С.Куликова «Тестирование ПО»

...

Критика текста или даже его автора. Помните, что ваша задача — сделать требования лучше, а не показать их недостатки (или недостатки автора). Потому комментарии вида «плохое требование», «неужели вы не понимаете, как глупо это звучит», «надо переформулировать» неуместны и недопустимы.

Категоричные заявления без обоснования.

Заявления наподобие «это невозможно», «мы не будем этого делать», «это не нужно». Даже если вы понимаете, что требование бессмысленно или невыполнимо, эту мысль стоит сформулировать в корректной форме и дополнить вопросами, позволяющими автору документа самому принять окончательное решение. Например, «это не нужно» можно переформулировать так: «Мы сомневаемся в том, что данная функция будет востребована пользователями. Какова важность этого требования? Уверены ли вы в его необходимости?»

...

ПЛАНИРОВАНИЕ ТЕСТИРОВАНИЯ ПО

- ✓ Анализ требований;
- ✓ Определение целей тестирования;
- ✓ Определение общего подхода к тестированию (уровни тестирования, виды тестирования, критерия входа);
- ✓ Интегрирование с разработкой ПО (требования, архитектура, дизайн, разработка, тестирование, релиз);
- ✓ Решение, какие роли нужны для выполнения тестирования, когда и как проводить тестирование и как оценивать результаты;
- ✓ Составление графика тестирования;
- ✓ Определение шаблонов для тестовой документации;
- ✓ Выбор метрик для мониторинга и контроля подготовки и проведения тестирования, исправления дефектов, проблем и рисков.

КРИТЕРИИ ВХОДА В ТЕСТИРОВАНИЕ

Критерий входа определяет, когда нужно начинать тестирование.

- ✓ Готовность и доступность тестового окружения;
- ✓ Готовность средства тестирования в окружении;
- ✓ Доступность тестируемого кода;
- ✓ Доступность тестовых данных.

КРИТЕРИИ ВЫХОДА

Критерий выхода определяет, когда нужно прекращать тестирование.

- ✓ Степень покрытие кода, функциональности или рисков тестами;
- ✓ Оценку плотности дефектов или измерение надежности;
- ✓ Стоимость;
- ✓ Остаточные риски (неисправленные дефекты или недостаток тестового покрытия какой-либо области);
- ✓ План, основанный на времени выхода ПО на рынок.

Классификация тестирования

Тестирование можно классифицировать по очень большому количеству признаков, на рисунке приведена **упрощенная классификация**



1 По запуску кода на исполнение:

- ✓ Статическое тестирование — без запуска.
- ✓ Динамическое тестирование — с запуском.

???

Статическое тестирование

Тестирование требований
Вычитка исходного кода

Динамическое тестирование

Модульное тестирование
Интеграционное тестирование
Приемочное тестирование

2 ПО СТЕПЕНИ АВТОМАТИЗАЦИИ

- ✓ Ручное тестирование;
- ✓ Автоматизированное тестирование.

ПЛЮСЫ РУЧНОГО ТЕСТИРОВАНИЯ

- ✓ Отчет тестировщика;
- ✓ Обратная связь по UI;
- ✓ Низкая стоимость;
- ✓ Мгновенное начало тестирования;
- ✓ Возможность тестировать нетипичные сценарии.

ПЛЮСЫ АВТО.ТЕСТИРОВАНИЯ

- ✓ Скорость работы;
- ✓ Отсутствие человеческого фактора;
- ✓ Затраты (при многократном использовании);
- ✓ Способность выполнять непосильные для человека тесты;
- ✓ Работа с большими объемами данных;
- ✓ Низкоуровневость;
- ✓ Многократное использование.

МИНУСЫ РУЧНОГО ТЕСТИРОВАНИЯ

- ✓ Человеческий фактор;
- ✓ Трудозатраты и продолжительность;
- ✓ Отсутствие возможности моделирования большой нагрузки.

МИНУСЫ АВТОМАТИЗИР. ТЕСТИРОВАНИЯ

- ✓ Необходимость в квалифицированном персонале;
- ✓ Большое количество средств;
- ✓ Низкая адаптивность;
- ✓ Отсутствие обратной связи;
- ✓ Невозможность взглянуть от лица пользователя;
- ✓ Невозможность протестировать дизайн;
- ✓ Относительно низкая надежность;
- ✓ Затраты (при однократном использовании).

3 По доступу к коду и архитектуре приложения:

- ✓ Метод «Белого ящика»
- ✓ Метод «Черного ящика»
- ✓ Метод «Серого ящика»

МЕТОД «БЕЛОГО ЯЩИКА»

Тестирование исходного кода ПО

МЕТОД «ЧЕРНОГО ЯЩИКА»

Тестирование ПО через интерфейс
(без доступа к исходному коду)

МЕТОД «БЕЛОГО ЯЩИКА»

Плюсы

- Упрощение диагностики
- Легко автоматизировать тесты
- Легко собирать данные
- Стимулирует разработчиков
- «Олдовость»

Минусы

- Высокий порог входа
- Фокус только на имеющемся функционале
- Не учитывается среда выполнения
- Не учитывается непредсказуемость пользователей

МЕТОД «ЧЕРНОГО ЯЩИКА»

Плюсы

- Низкий порог входа
- Учитывается среда выполнения
- Учитывается непредсказуемость пользователей
- Раннее тестирование
- Тестирование требований
- Многократность использования

Минусы

- Повторяемость
- Вероятность неполного покрытия
- Необходимость качественной документации
- Повышенная сложность
- Трудности с планированием
- Потенциальная дороговизна

МЕТОД «СЕРОГО ЯЩИКА»

Плюсы

- Комбинация методов Белого и Черного ящиков
- Возможность усложнения тестов
- Дает много времени на дебаг
- Легко убрать ненужные тесты

Минусы

- Ограничен анализ кода
- Неполное покрытие тестами
- Ситуации ненужности тестировщиков

4 По уровню детализации приложения (по уровню тестирования) :

- ✓ **Модульное** (компонентное) тестирование — проверяются отдельные небольшие части приложения
- ✓ **Интеграционное** тестирование — проверяется взаимодействие между несколькими частями приложения.
- ✓ **Системное** тестирование — приложение проверяется как единое целое

Скидки

Рейтинг
товаров

Оплата
WebMoney

Отзывы о
товарах

**КОМПОНЕНТНОЕ
ТЕСТИРОВАНИЕ**

Рейтинг товаров +
Каталог товаров

Скидки + Оплата

Рейтинг товаров +
Поиск товаров

Оплата WebMoney +
Профайл покупателя

**ИНТЕГРАЦИОННОЕ
ТЕСТИРОВАНИЕ**

Каталог товаров + Поиск товаров + Оплата +
Профайл покупателя + Скидки + Рейтинг товаров +
Оплата WebMoney + Отзывы о товарах

**СИСТЕМНОЕ
ТЕСТИРОВАНИЕ**

5 По (убыванию) степени важности тестируемых функций (по уровню функционального тестирования) :

- ✓ **Дымовое тестирование** (smoke test) — проверка самой важной, самой ключевой функциональности, неработоспособность которой делает бессмысленной саму идею использования приложения.
- ✓ **Тестирование критического пути** (critical path test) — проверка функциональности, используемой типичными пользователями в типичной повседневной деятельности.
- ✓ **Расширенное тестирование** (extended test) — проверка всей (остальной) функциональности, заявленной в требованиях.

6 По принципам работы с приложением:

- ✓ **Позитивное тестирование** — все действия с приложением выполняются строго по инструкции без никаких недопустимых действий, некорректных данных и т.д. Можно образно сказать, что приложение исследуется в «тепличных условиях»
- ✓ **Негативное тестирование** — в работе с приложением выполняются (некорректные) операции и используются данные, потенциально приводящие к ошибкам (классика жанра — деление на ноль).



Рассмотрели упрощенную классификацию, схема и описание более подробной классификации приведена в книге «Тестирование программного обеспечения» Святослава Куликова на стр.72