Лекция 6. Основные принципы тестирования

Что такое тестирование?

Для начала мы ...

… удостоверяемся, все ли в порядке

Почему тестирование необходимо?

- Тестирование необходимо, потому что люди склонны ошибаться. Одни ошибки незначительны, другие же опасны и дорого обходятся.
- Поскольку ошибки допускают все люди, мы должны внимательно проверять результаты своей (и чужой ;-)) работы, всего, что мы делаем.

Что такое тестирование? 1/2

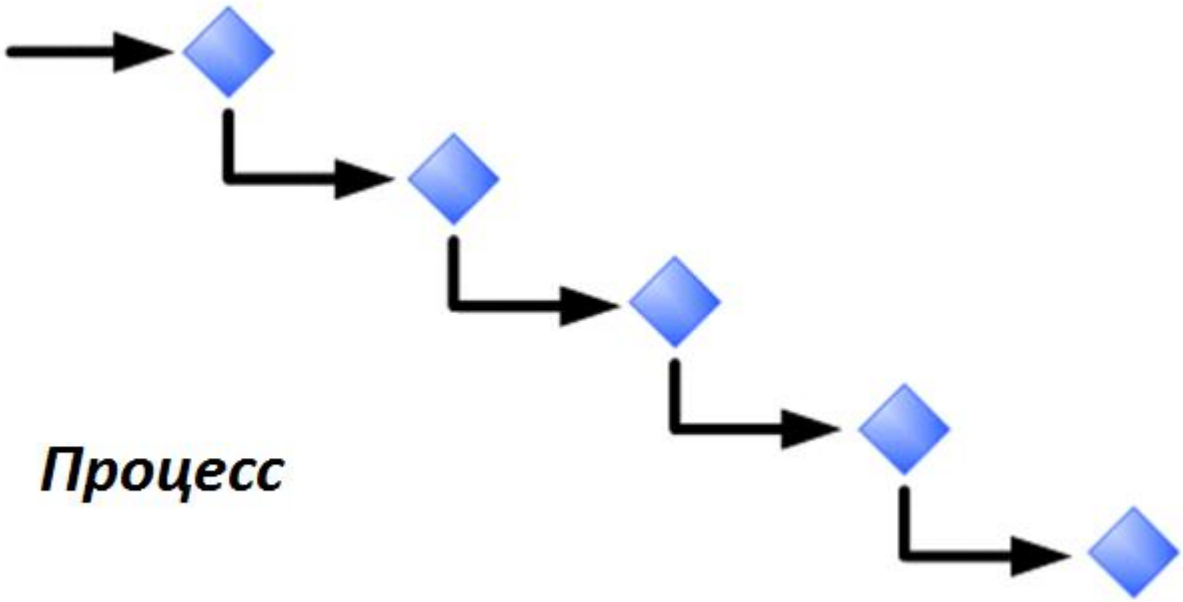
- Это процесс исполнения программы с целью обнаружения ошибок (“Искусство тестирования программ”, Г. Майерс, 1979)
- Процесс наблюдения за выполнением программы в специальных условиях и вынесения на этой основе оценки каких-либо ее аспектов ([ANSI/IEEE standard 610.12-1990: Glossary of SE Terminology. NY:IEEE, 1987])
- Техническое исследование программы для получения информации о ее качестве с точки зрения определенного круга заинтересованных лиц [C. Kaner, 1999]
- Проверка соответствия между реальным поведением программы и ее ожидаемым поведением на конечном наборе тестов, выбранном определенным образом [IEEE Guide to Software Engineering Body of Knowledge, SWEBOK, 2004]

Что такое тестирование? 2/2

Процесс, содержащий в себе все активности жизненного цикла, как динамические, так и статические, касающиеся планирования, подготовки и оценки программного продукта и связанных с этим результатов работ с целью определить, что они соответствуют описанным требованиям, показать, что они подходят для достижения заявленных целей, а также для нахождения дефектов.

Определение тестирования «по частям» 1/5

Во-первых, тестирование – это процесс, а не единичное действие



Определение тестирования «по частям» 2/5

Процесс тестирования включен во все активности жизненного цикла

Все активности жизненного цикла

Определение тестирования «по частям» 3/5

Тестирование ПО может быть статическим и динамическим

Статическое тестирование: Тестирование компонента или системы на уровне спецификации или реализации **без исполнения кода программного продукта**, например рецензирование или статический анализ кода.

Динамическое тестирование: Тестирование, проводимое **во время выполнения** программного обеспечения, компонента или системы.

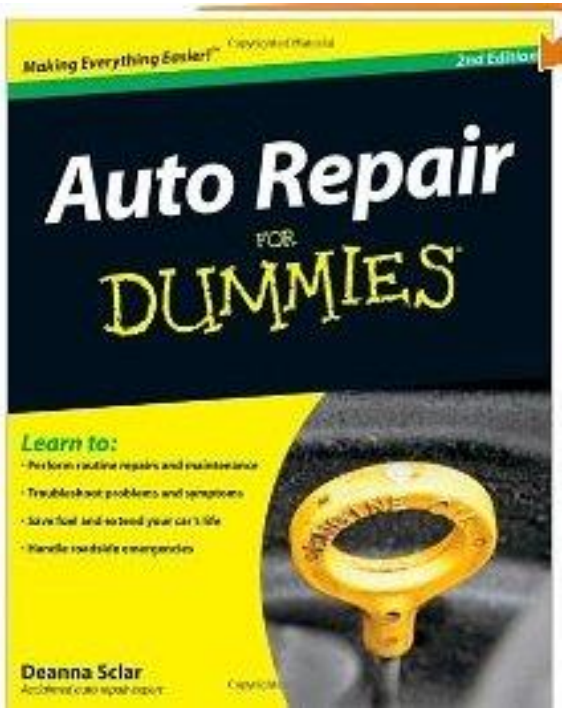
ISTQB International Software Testing Qualifications Board

Определение тестирования «по частям» 4/5

- Планирование
- Подготовка
- Оценка

Определение тестирования «по частям» 5/5

Тестированию подлежит программный продукт и связанные с ним рабочие продукты



Цели тестирования

- Предоставление информации для принятия решений
- Повышение уверенности в уровне качества
- Обнаружение дефектов
- Предотвращение дефектов

Тестирование помогает уменьшить общий уровень риска в системе после обнаружения и устранения дефектов и порождает уверенность в качестве ПО

Определение тестирования: сравнение как ключевое понятие

Тестирование всегда предполагает сравнение.

Что с чем сравнивается?

1. Объект тестирования (что сравнивается)
2. Базис тестирования (с чем сравнивается)

Терминология

Объект тестирования: Компонент или система, которые должны быть протестированы.

Базис тестирования: Документ, на основании которого определяются требования к компоненту или системе. Документация, на которой базируются тестовые сценарии.

Если правка данного документа может быть осуществлена только в процессе формальной процедуры внесения изменения, то такой базис тестирования называется **замороженным базисом тестирования.**

Рабочие продукты 1/2

Рабочие продукты, поставляемые команде тестировщиков в качестве объектов тестирования, могут быть разными:

- отдельный модуль
- компонент (несколько модулей)
- подсистема
- система

Рабочие продукты 2/2

- документация с требованиями (маркетинговая, пользовательская, техническая)
- требования (функциональные , проектные, базы данных)
- модели, диаграммы, макеты
- сценарии использования
- код
- тестовые планы и сценарии
- проектная документация по автоматизации тестирования, код автоматизации тестирования
- другие документы или код

Что такое дефект?

Дефект: Изъян в компоненте или системе, который может привести компонент или систему к невозможности выполнить требуемую функцию, например неверный оператор или определение данных. Дефект, обнаруженный во время выполнения, может привести к отказам компонента или системы.

Баг – синоним слова «дефект»

Как определить дефект перед нами или нет?

1. Программа не делает чего-то, что она должна делать согласно техническим требованиям.
2. Программа делает что-то, чего она не должна делать согласно техническим требованиям.
3. Программа делает что-то, о чем в требованиях не упоминалось (?).
4. Программа не делает чего-то, о чем не говорится в требованиях, однако подразумевается, что она должна делать это.
5. Программа трудна для понимания, неудобна в использовании.

Связанные понятия: ошибка и отказ 1/2

Люди делают ошибки.

Если кто-то допустит ошибку в архитектуре или коде программы, то эта программа будет содержать дефект.

При исполнении программы любой дефект может привести к отказу.

Связанные понятия: ошибка и отказ 2/2

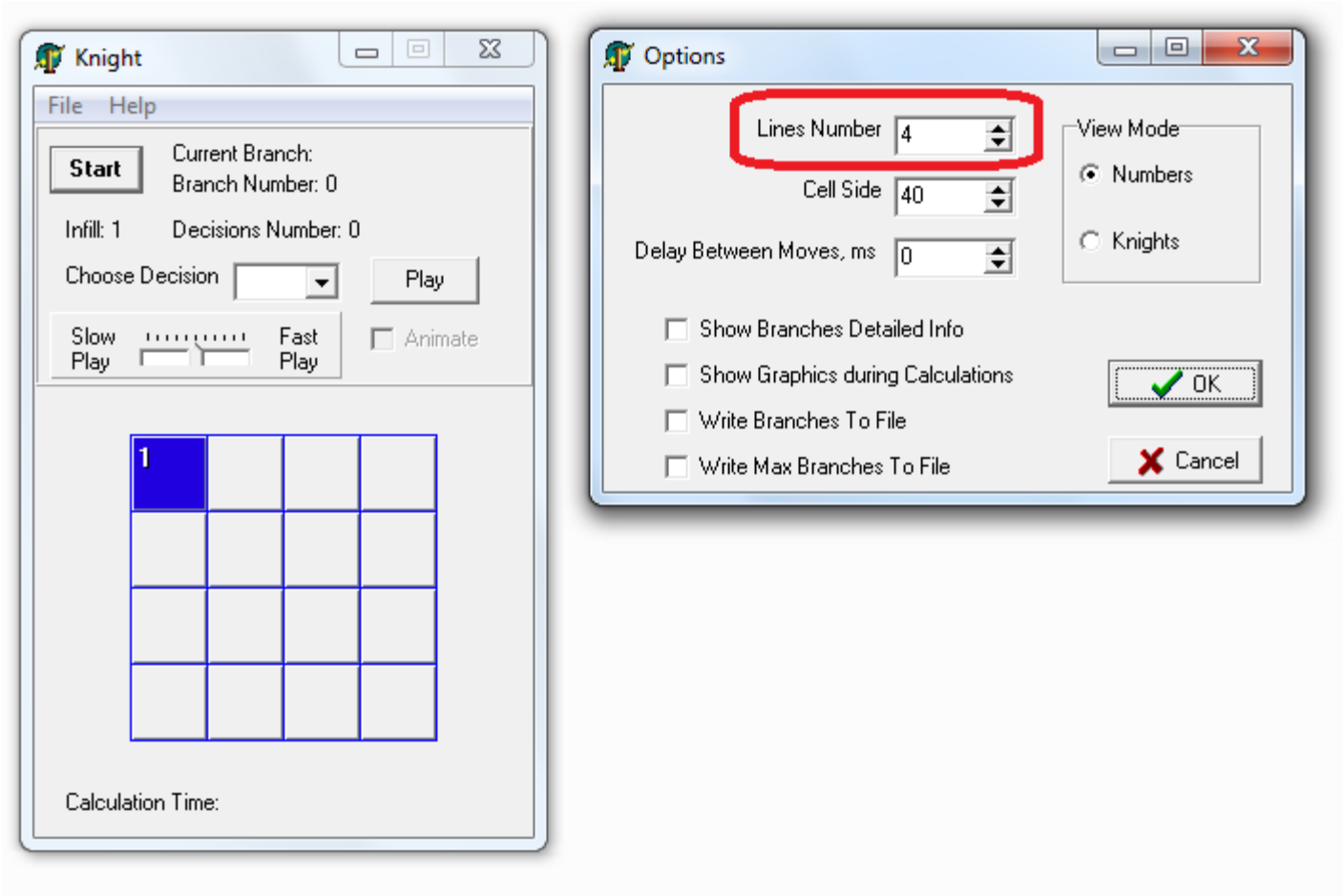
Ошибка: Действие человека, которое приводит к неправильному результату .

Отказ: Отклонение компонента или системы от ожидаемого выполнения, эксплуатации или результата.

Демонстрация дефекта - Требования

- На примере программы TestKnight
- Фрагмент требований:
  - **Диалоговое окно Опций**
    - **Lines Number** – размер шахматной доски (3…10)
    - **Cell Side** – размер стороны клетки в пикселях
    - **Delay Between Moves, ms** – пауза между движениями в процессе вычислений (0…5000). Используется для понимания принципов работы программы. Данную опцию следует использовать вместе с опцией
    - **Show …**
    - **…..**

Демонстрация дефекта - Программа



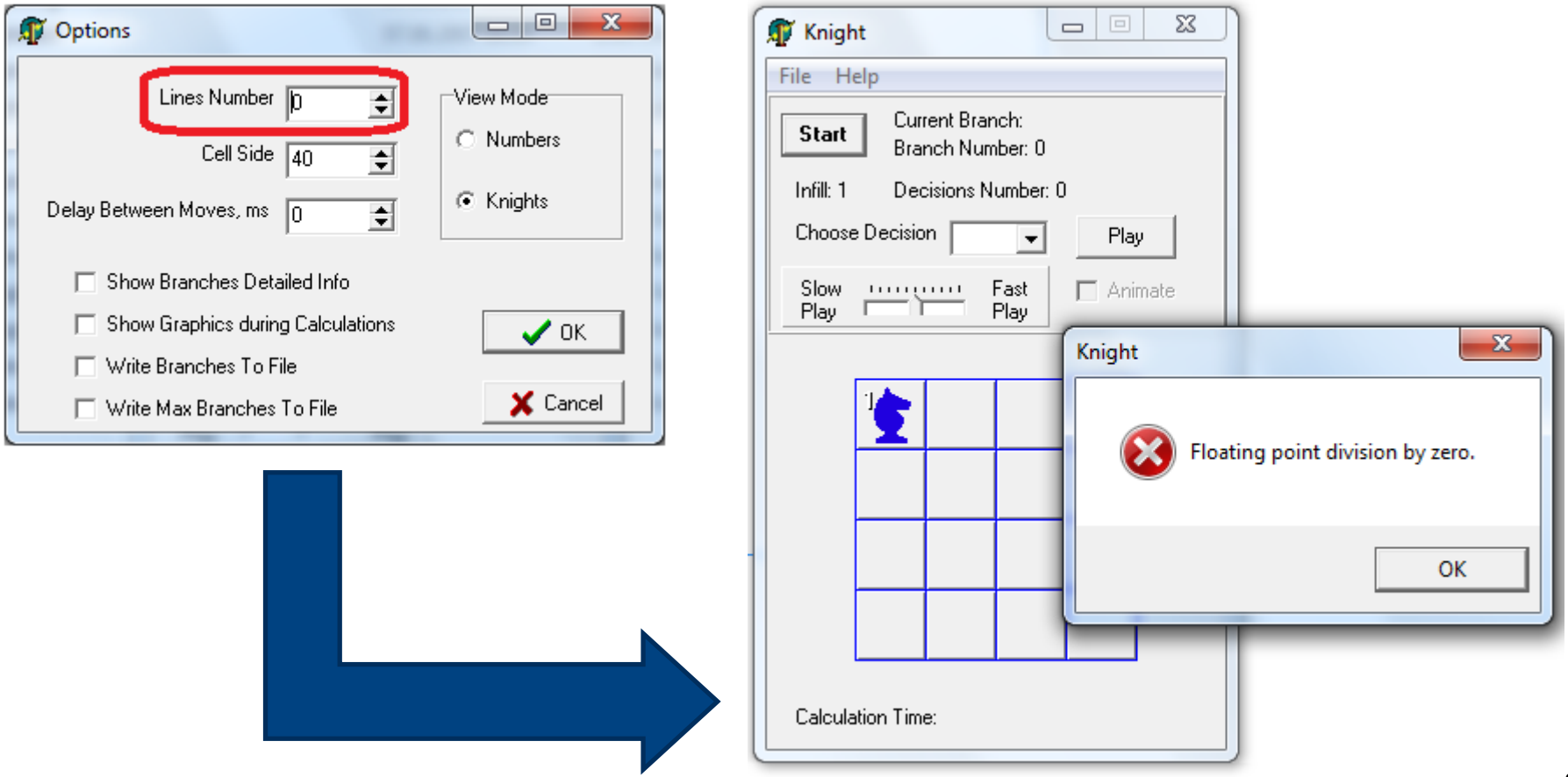
Демонстрация дефекта – Ошибка кодирования

- Нет проверки (забыли ;-)) на диапазон обозначенный в требованиях 3-10

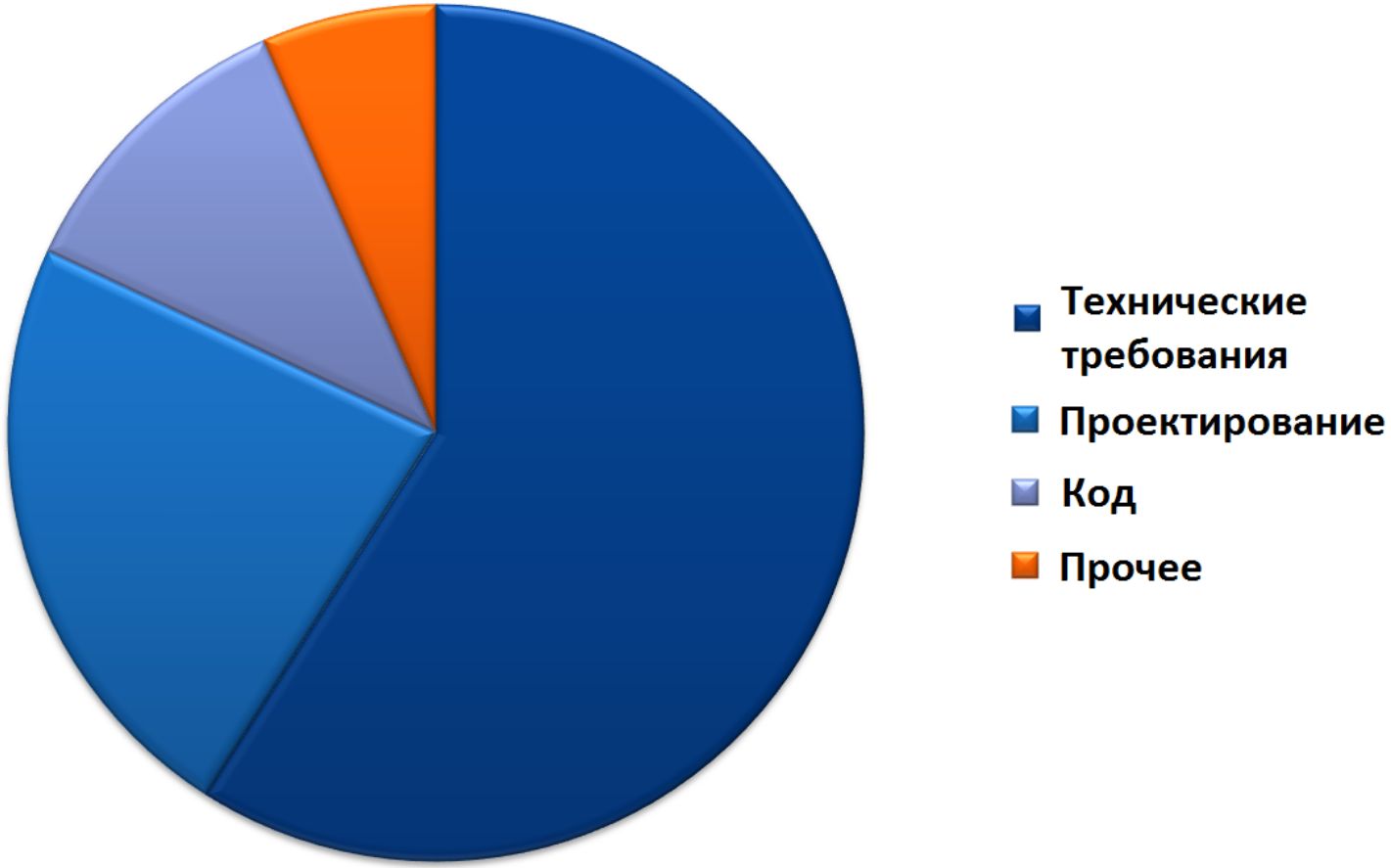
```
procedure TBoard.GetOptions;
{ Gets option significance from Frm_Options }
begin
  Frm_Knight.DeselectStartPosMode;

  SetOptions;

  Frm_Options.ShowModal;

  if (NumLines <> Frm_Options.Spin_NumLines.Value) or
    (NeedWriteToFile <> Frm_Options.Chk_WriteBranchesToFile.Checked) or
    (NeedWriteToFileMax <> Frm_Options.Chk_WriteMaxBranchesToFile.Checked) then
  begin
    NumLines := Frm_Options.Spin_NumLines.Value;

    NeedWriteToFile := Frm_Options.Chk_WriteBranchesToFile.Checked;
    NeedWriteToFileMax := Frm_Options.Chk_WriteMaxBranchesToFile.Checked;

    ChessBoardZeroing;
    Initialize;
    SetStartPos(1, 1);

    Frm_Knight.Cmb_Decisions.Items.Clear;
    DecisionsList.Clear;
    NumDecisions := 0;
  end;
```

Демонстрация дефекта – Сбой

- Вводим в параметрах значение 0
- Нажимаем Ок



Источники дефектов 1/2

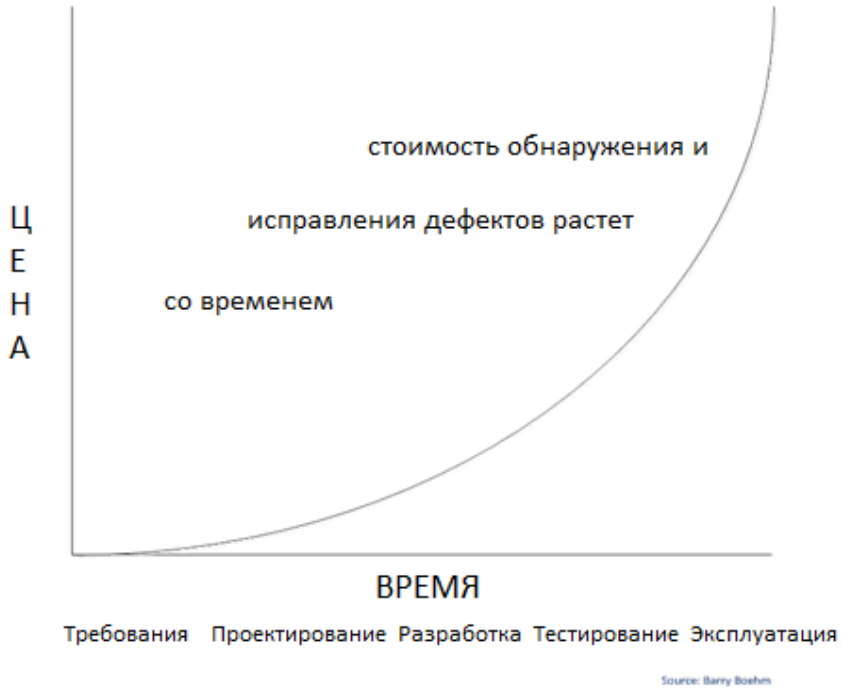


Дефекты появляются по разным причинам, но, как правило, их источником являются технические требования (спецификация).

Источники дефектов 2/2

Source: Dorothy Graham, Erik van Veenendaal, Isabel Evans, Rex Black "Foundations of Software Testing"

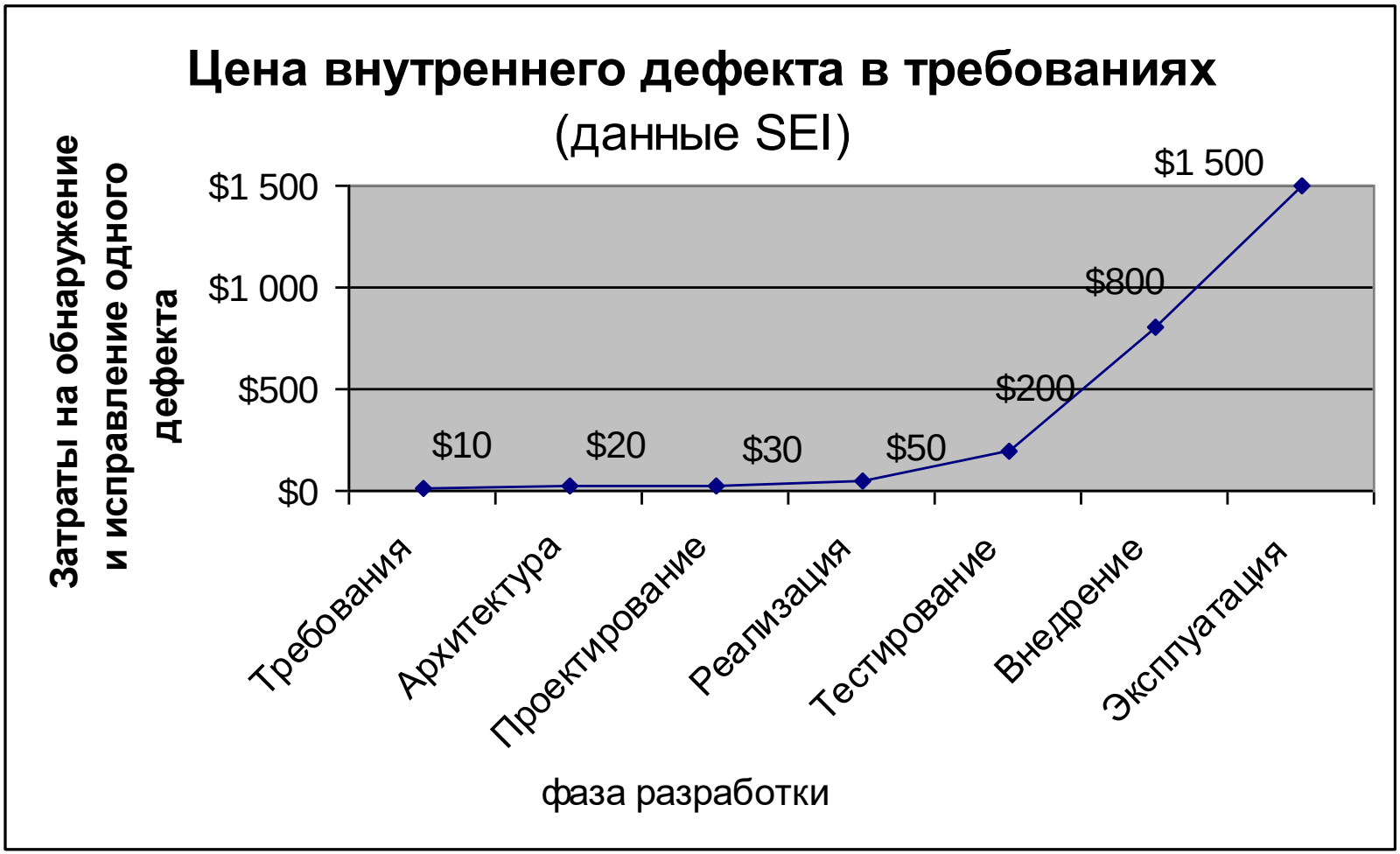
Цена дефектов 1/2



Обнаружение и исправление дефекта программы после поставки обходится в **100 раз** дороже, чем на стадии формирования требований и проектирования.

Как отмечал Боэм в 1987 г., «именно это понимание заставляло разработчиков уделять главное внимание тщательному анализу требований и проектированию, ранней верификации и валидации, а также моделированию и имитации, которые помогали избежать затратных послепродажных работ по устранению неисправностей».

Цена дефектов 2/2



Чем раньше дефект обнаружен, тем дешевле обходится его исправление

Тестирование и качество 1/9

Что такое качество?

«Качество – это ценность для индивидуума…»

Дж. Вайнберг (1992)

Тестирование и качество 2/9

Вопрос:

Отвечает ли тестировщик за качество?

Тестирование и качество 3/9

В IT-индустрии широко используется два понятия, которые напрямую связаны с тестированием программных продуктов:

- обеспечение качества (QA)
- контроль качества (QC)

Зачастую роль тестирования понимается неправильно.

Мы, как специалисты по тестированию, не обеспечиваем качество своей деятельностью.

Тестирование и качество 4/9

Тестирование и качество 5/9

В контроль качества входят:

- Тестирование
- Рецензирование кода
- Статический анализ кода
- Внешняя оценка и аудит

реактивные действия

В обеспечение качества входят :

- Усовершенствование процессов
- *Контроль качества*
- Управление изменениями

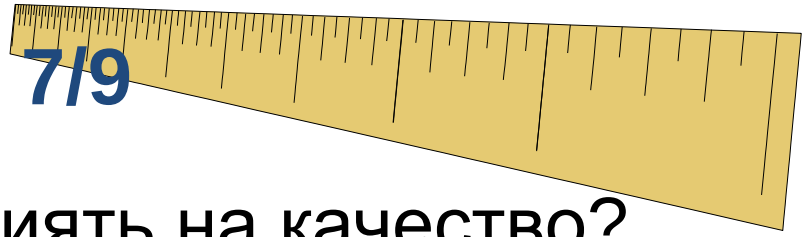
реактивные и проактивные действия

Тестирование и качество 6/9

Тестирование выполняется для сбора информации.

Поэтому тестирование – это лишь один из информационных сервисов.

Тестирование и качество 7/9



Как тестировщик может повлиять на качество?

Тестирование - это возможный способ оценки качества программного обеспечения в терминах найденных дефектов, исполненных тестов и протестированных систем. Это может быть сделано как для функциональных требований, так и для нефункциональных требований и характеристик программного обеспечения.

Когда во время тестирования находятся ошибки, качество систем программного обеспечения повышается, если эти дефекты исправлены.

Тестирование и качество 8/9

Можно думать о себе, как о гаранте качества, но вы не создаете качество и не можете лишить продукт его.

Качество должно закладываться создателями продукта и зачастую для них это становиться неподъемной ношей.

Тестировщик призван помочь им решать эту задачу более эффективно.

Тестирование и качество 9/9

Любой проект похож на езду по дороге. Проекты бывают легкие и типовые, но большинство напоминают заснеженную горную трассу. В этих проектах не обойтись без света фар.

Как тестировщик, *вы освещаете дорогу*.

7 принципов тестирования

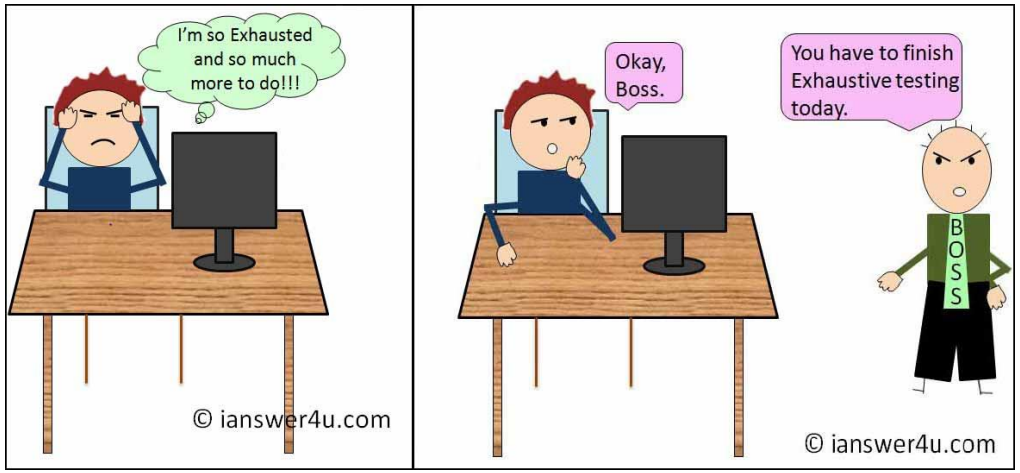
Принцип 1 – Тестирование демонстрирует наличие дефектов

Тестирование может показать, что дефекты присутствуют, но не может доказать, что их нет. Тестирование снижает вероятность наличия дефектов, находящихся в программном обеспечении, но, даже если дефекты не были обнаружены, это не доказывает его корректности.

7 принципов тестирования

Принцип 2 – Исчерпывающее тестирование недостижимо

Полное тестирование с использованием всех комбинаций вводов и предусловий физически невыполнимо, за исключением тривиальных случаев. Вместо исчерпывающего тестирования должны использоваться анализ рисков и расстановка приоритетов, чтобы более точно сфокусировать усилия по тестированию.



ISTQB International Software Testing Qualifications Board

7 принципов тестирования

Принцип 3 – Раннее тестирование

Чтобы найти дефекты как можно раньше, активности по тестированию должны быть начаты как можно раньше в жизненном цикле разработки программного обеспечения или системы, и должны быть сфокусированы на определенных целях.

7 принципов тестирования

Принцип 4 – Скопление дефектов

Б**о**льшая часть дефектов, обнаруженных при тестировании или повлекших за собой основное количество сбоев системы, содержится в небольшом количестве модулей.

ISTQB International Software Testing Qualifications Board

7 принципов тестирования

Принцип 5 – Парадокс пестицида

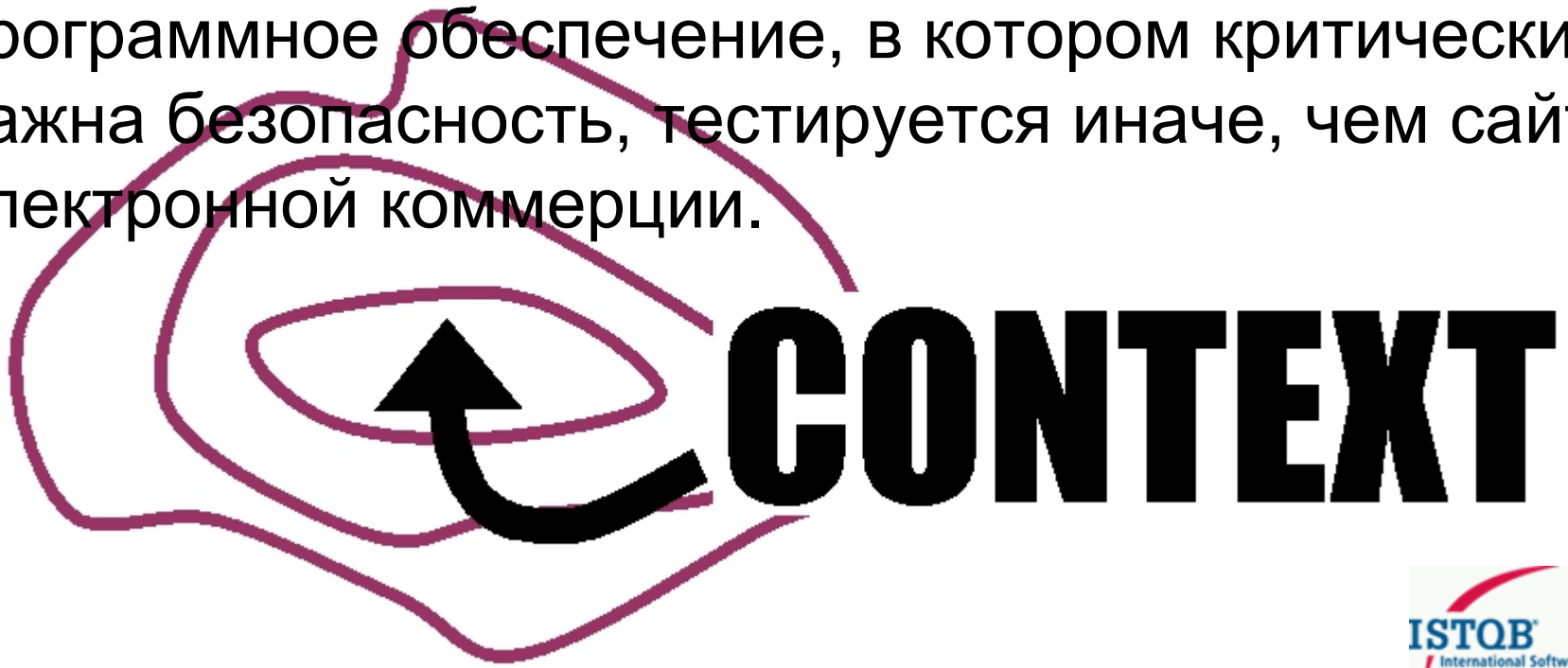
Если одни и те же тесты будут прогоняться много раз, в конечном счете этот набор тестовых сценариев больше не будет находить новых дефектов. Чтобы преодолеть этот “парадокс пестицида”, тестовые сценарии должны регулярно пересматриваться и корректироваться, новые тесты должны быть разносторонними, чтобы охватить все компоненты программного обеспечения, или системы, и найти как можно больше дефектов.

ISTQB International Software Testing Qualifications Board

7 принципов тестирования

Принцип 6 – Тестирование зависит от контекста

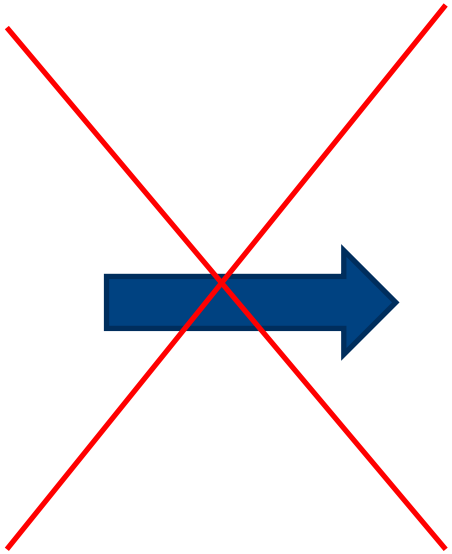
Тестирование выполняется по-разному в зависимости от контекста. Например, программное обеспечение, в котором критически важна безопасность, тестируется иначе, чем сайт электронной коммерции.



7 принципов тестирования

Принцип 7 – Заблуждение об отсутствии ошибок

Обнаружение и исправление дефектов не помогут, если созданная система не подходит пользователю и не удовлетворяет его ожиданиям и потребностям.



Вот такие ошибки …

- F-16 вверх ногами
  - Испытания американского истребителя F-16 проводились, понятное дело, в северном полушарии. На заключительном этапе самолет решили проверить где-то в Латинской Америке, но уже с другой стороны экватора. При переводе самолета в режим автопилота он автоматически развернулся «вверх ногами».
- Правильно выбирайте типы данных
  - Причиной взрыва 4 июня 1996 г. ракеты Ариан-5, была программная ошибка. В системе управления ракеты использовалось модифицированное программное обеспечение ранее успешно работавшее на Ариан-4, но Ариан-5 ускорялась быстрее предыдущей модификации, в результате когда на 40 секунде полета одна из вспомогательных подпрограмм попыталась преобразовать длинное целое значение в короткое без проверки величины значения, то вышло за границы типа, произошло отключение системы управления ракеты, и она была взорвана по команде на самоликвидацию. Прямой (вместе с ракетой-носителем был потерян коммуникационный спутник) и косвенный ущерб от этого программного сбоя был оценен в полмиллиарда долларов.

Независимость тестирования

Тип мышления, требуемый для тестирования и рецензирования, отличается от типа мышления, требуемого для разработки. С правильной установкой разработчики сами могут тестировать собственный код, однако ответственность за это передается тестировщику, как правило, для того чтобы сфокусироваться именно на тестировании и получить ряд дополнительных преимуществ, таких как независимый взгляд обученных и профессиональных тестировщиков. Независимое тестирование может быть выполнено на любом уровне тестирования.

Независимость тестирования

Независимость тестирования: Разделение ответственностей, которое позволяет выполнять объективное тестирование.

Уровни независимости

Ниже описываются несколько уровней независимости, в порядке от низкого к высокому:

- Разработчики тестируют собственный код (низкий уровень независимости)
- Независимые тестировщики (например, из команды разработчиков)
- Независимая команда или группа тестирования из другой организационной группы или независимые тестировщики (например, специалисты по тестированию удобства использования и производительности)
- Независимые тестировщики, привлеченные на аутсорсинг или сторонние по отношению к организации.

Важность независимости тестирования 1/2

Причина 1 – Редактировать и править собственный код – не самая лучшая идея. Свежий взгляд необходим, т.к. проверяя свою работу, вы руководствуетесь теми же предположениями, что и при написании, а, значит, серьезные дефекты останутся незамеченными.

Например, программа может содержать ошибки, обусловленные непониманием программиста поставленной задачи или технического задания. В этом случае, непонимание программиста, скорее всего, отразится и в самой программе.

Важность независимости тестирования 2/2

Причина 2 – Никому не нравится находить ошибки в своей работе. Это распространяется и на разработчиков программных продуктов.

Причина 3 – Смена фокусировки в проектной активности так же представляет собой проблему. После *конструктивной работы по проектированию и написанию кода* программисту чрезвычайно сложно переключиться, и вести в отношении собственной же программы *деструктивную деятельность.*