Системная и программная инженерия

ФИО преподавателя: Туманова М.Б.

Гусев К.В., Миронов А.Н.

e-mail: tumanova@mirea.ru

Тема 14
Техники тестирования систем

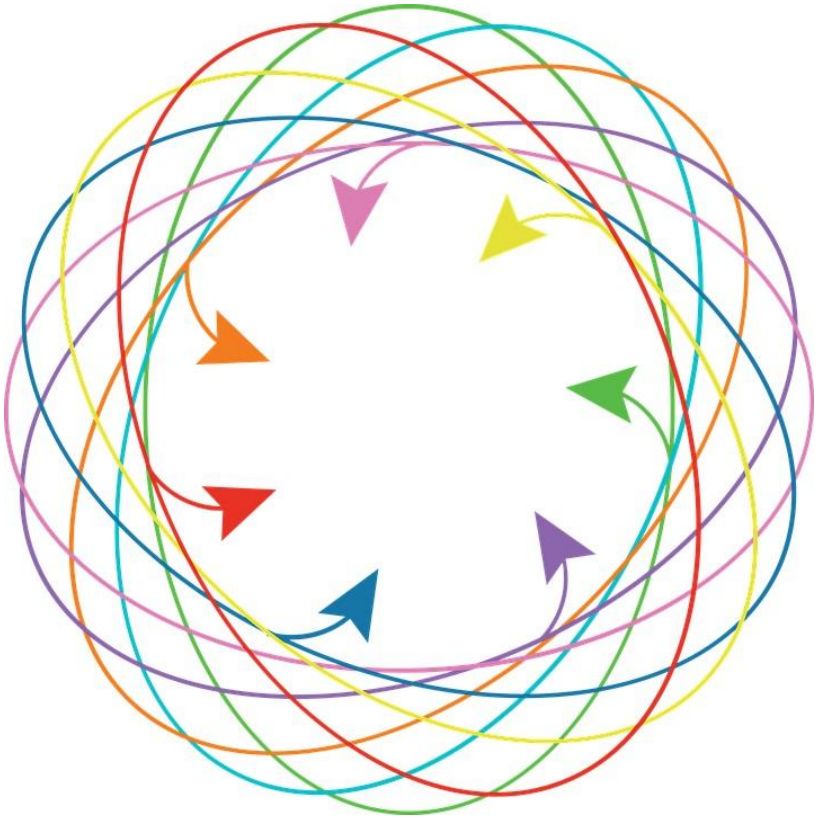
Уровни тестирования

1. Модульное тестирование (Unit Testing) Компонентное (модульное) тестирование проверяет функциональность и ищет дефекты в частях приложения, которые доступны и могут быть протестированы по-отдельности (модули программ, объекты, классы, функции и т.д.).

Уровни тестирования

2. Интеграционное тестирование (Integration Testing)

Проверяется взаимодействие между компонентами системы после проведения компонентного тестирования.



Уровни тестирования

3. Системное тестирование (System Testing)
Основной задачей системного тестирования является проверка как функциональных, так и не функциональных требований в системе в целом.

При этом выявляются дефекты:

- неверное использование ресурсов системы;
- непредусмотренные комбинации данных пользовательского уровня;
- несовместимость с окружением;
- непредусмотренные сценарии использования;
- отсутствующая или неверная функциональность;
- неудобство использования и т.д.

Уровни тестирования

4. Операционное тестирование (Release Testing). Даже если система удовлетворяет всем требованиям, важно убедиться в том, что она удовлетворяет нуждам пользователя и выполняет свою роль в среде своей эксплуатации, как это было определено в бизнес-моделе системы.

Следует учесть, что и бизнес модель может содержать ошибки. Поэтому так важно провести операционное тестирование как финальный шаг валидации.

Уровни тестирования

4. **Операционное тестирование (Release Testing)** Тестирование в среде эксплуатации позволяет выявить и нефункциональные проблемы:

- конфликт с другими системами, смежными в области бизнеса или в программных и электронных окружениях;
- недостаточная производительность системы в среде эксплуатации и др.

Очевидно, что нахождение подобных вещей на стадии внедрения — критичная и дорогостоящая проблема. Поэтому так важно проведение не только верификации, но и валидации, с самых ранних этапов разработки ПО.

Уровни тестирования

5. Приемочное тестирование (Acceptance Testing)
Формальный процесс тестирования, который проверяет соответствие системы требованиям и проводится с целью:
• определения удовлетворяет ли система приемочным критериям;
• вынесения решения заказчиком или другим уполномоченным лицом принимается приложение или нет.

Виды / типы тестирования

Функциональные виды тестирования

- Функциональное тестирование (Functional testing).
- Тестирование безопасности (Security and Access Control Testing).
- Тестирование взаимодействия (Interoperability Testing).

Функциональные виды тестирования

Функциональное тестирование рассматривает заранее указанное поведение и основывается на анализе спецификаций функциональности компонента или системы в целом.

Преимущества функционального тестирования:

- имитирует фактическое использование системы.

Недостатки функционального тестирования:

- возможность упущения логических ошибок в программном обеспечении;
- вероятность избыточного тестирования.

Тестирование безопасности — это стратегия тестирования, используемая для проверки безопасности системы, а также для анализа рисков, связанных с обеспечением целостного подхода к защите приложения, атак хакеров, вирусов, несанкционированного доступа к конфиденциальным данным.

Принципы безопасности программного обеспечения

Общая стратегия безопасности основывается на трех основных принципах:

- Конфиденциальность.
- Целостность.
- Доступность.

Функциональные виды тестирования

Тестирование взаимодействия (Interoperability Testing) — это функциональное тестирование, проверяющее способность приложения взаимодействовать с одним и более компонентами или системами и включающее в себя тестирование совместимости (compatibility testing) и интеграционное тестирование.

Виды / типы тестирования

Нефункциональные виды тестирования

• Все 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 (Installation testing);
• Тестирование удобства пользования (Usability Testing);
• Тестирование на отказ и восстановление (Failover and Recovery Testing);
• Конфигурационное тестирование (Configuration Testing);

Нефункциональные виды тестирования

Нагрузочное тестирование — это автоматизированное тестирование, имитирующее работу определенного количества бизнес пользователей на каком-либо общем (разделяемом ими) ресурсе.

Стрессовое тестирование (Stress Testing) позволяет проверить насколько приложение и система в целом работоспособны в условиях стресса и также оценить способность системы к регенерации, т.е. к возвращению к нормальному состоянию после прекращения воздействия стресса. Стрессом в данном контексте может быть повышение интенсивности выполнения операций до очень высоких значений или аварийное изменение конфигурации сервера.

Нефункциональные виды тестирования

Объемное тестирование (Volume Testing). Задачей объемного тестирования является получение оценки производительности при увеличении объемов данных в базе данных приложения.

Тестирование стабильности или надежности (Stability / Reliability Testing). Задачей тестирования стабильности (надежности) является проверка работоспособности приложения при длительном (многочасовом) тестировании со средним уровнем нагрузки.

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

Нефункциональные виды тестирования

Тестирование удобства пользования — это метод тестирования, направленный на установление степени удобства использования, обучаемости, понятности и привлекательности для пользователей разрабатываемого продукта в контексте заданных условий. Сюда также входит:

- Тестирование пользовательского интерфейса (англ. UI Testing) — это вид тестирования исследования, выполняемого с целью определения, удобен ли некоторый искусственный объект для его предполагаемого применения.
- User eXperience (UX) — ощущение, испытываемое пользователем во время использования цифрового продукта, в то время как User interface — это инструмент, позволяющий осуществлять интеракцию «пользователь — веб-ресурс».

Нефункциональные виды тестирования

Тестирование на отказ и восстановление (Failover and Recovery Testing) проверяет тестируемый продукт с точки зрения способности противостоять и успешно восстанавливаться после возможных сбоев, возникших в связи с ошибками программного обеспечения, отказами оборудования или проблемами связи (например, отказ сети).

Целью данного вида тестирования является проверка систем восстановления (или дублирующих основной функционал систем), которые, в случае возникновения сбоев, обеспечат сохранность и целостность данных тестируемого продукта.

Нефункциональные виды тестирования

Конфигурационное тестирование (Configuration Testing) — специальный вид тестирования, направленный на проверку работы программного обеспечения при различных конфигурациях системы (заявленных платформах, поддерживаемых драйверах, при различных конфигурациях компьютеров и т.д.)

Виды / типы тестирования

Связанные с изменениями виды тестирования:

- Дымовое тестирование (Smoke Testing).
- Регрессионное тестирование (Regression Testing).
- Повторное тестирование (Re-testing).
- Тестирование сборки (Build Verification Test).
- Санитарное тестирование или проверка согласованности/исправности (Sanity Testing).

Связанные с изменениями виды тестирования

Дымовое (Smoke) тестирование рассматривается как короткий цикл тестов, выполняемый для подтверждения того, что после сборки кода (нового или исправленного) устанавливаемое приложение, стартует и выполняет основные функции.

Связанные с изменениями виды тестирования

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. Регрессионными могут быть как функциональные, так и нефункциональные тесты.

3 основных типа регрессионного тестирования (Сэм Канер):

Регрессия багов (**Bug regression**) - попытка доказать, что исправленная ошибка на самом деле не исправлена.

Регрессия старых багов (**Old bugs regression**) - попытка доказать, что недавнее изменение кода или данных сломало исправление старых ошибок, т.е. старые баги стали снова воспроизводиться.

Регрессия побочного эффекта (**Side effect regression**) - попытка доказать, что недавнее изменение кода или данных сломало другие части разрабатываемого приложения.

– Re-testing
– Regression testing
- Build Verification Test

Связанные с изменениями виды тестирования

Повторное тестирование — тестирование, во время которого исполняются тестовые сценарии, выявившие ошибки во время последнего запуска, для подтверждения успешности исправления этих ошибок.

Связанные с изменениями виды тестирования

Тестирование сборки или Build Verification Test — тестирование направленное на определение соответствия, выпущенной версии, критериям качества для начала тестирования.

По своим целям является аналогом Дымового Тестирования, направленного на приемку новой версии в дальнейшее тестирование или эксплуатацию.
Вглубь оно может проникать дальше, в зависимости от требований к качеству выпущенной версии.

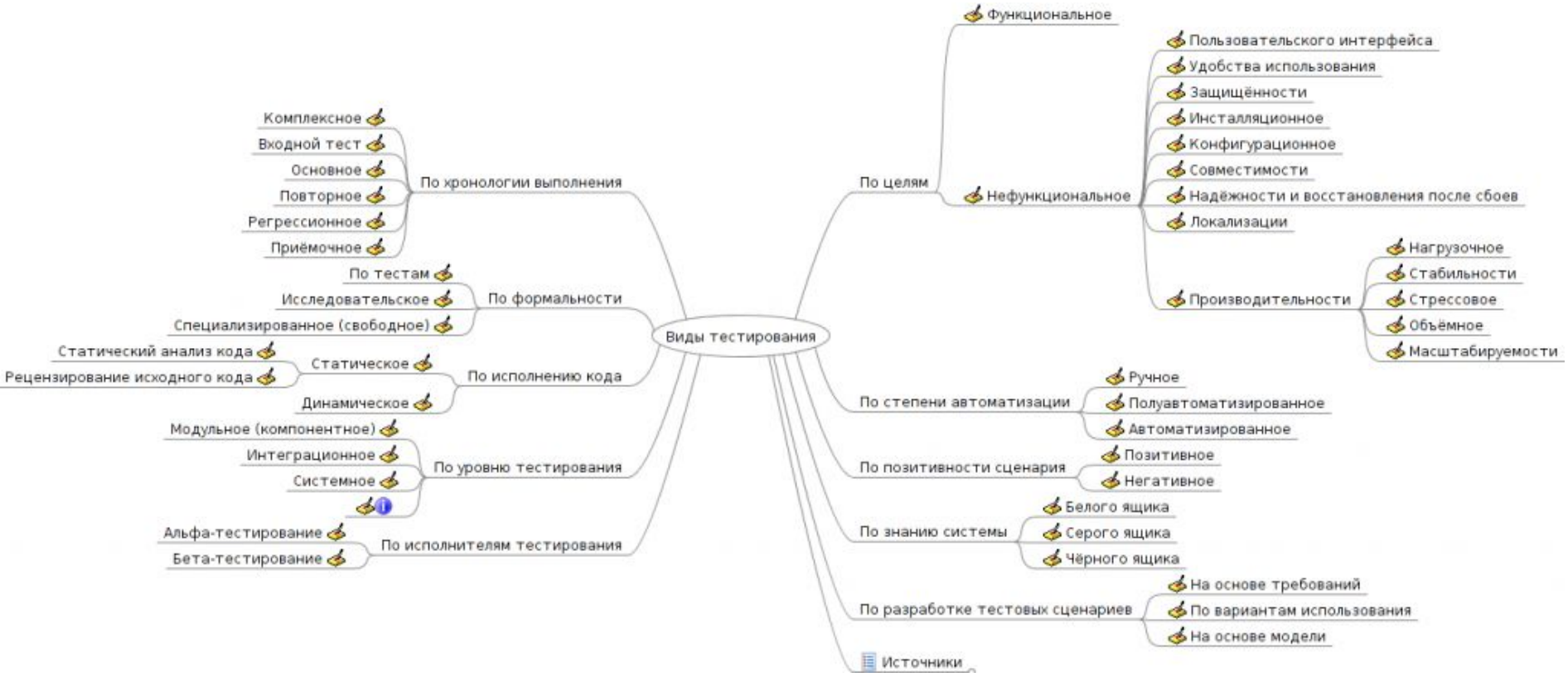
Связанные с изменениями виды тестирования

Санитарное тестирование — это узконаправленное тестирование достаточное для доказательства того, что конкретная функция работает согласно заявленным в спецификации требованиям.

Является подмножеством регрессионного тестирования.

Используется для определения работоспособности определенной части приложения после изменений произведенных в ней или окружающей среде. Обычно выполняется вручную.

Виды тестирования



Подходы к интеграционному тестированию

- **Снизу вверх (Bottom Up Integration)**

Все низкоуровневые модули, процедуры или функции собираются воедино и затем тестируются. После чего собирается следующий уровень модулей для проведения интеграционного тестирования. Данный подход считается полезным, если все или практически все модули, разрабатываемого уровня, готовы. Также данный подход помогает определить по результатам тестирования уровень готовности приложения.

Подходы к интеграционному тестированию

- **Сверху вниз (Top Down Integration)**

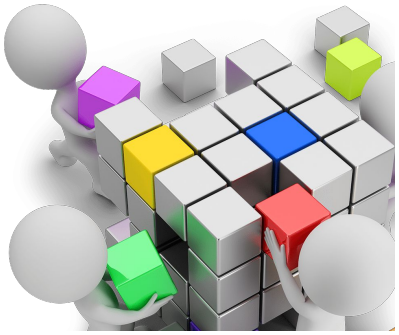
Вначале тестируются все высокоуровневые модули, и постепенно один за другим добавляются низкоуровневые. Все модули более низкого уровня симулируются заглушками с аналогичной функциональностью, затем по мере готовности они заменяются реальными активными компонентами. Таким образом мы проводим тестирование сверху вниз.

Подходы к интеграционному тестированию

- **Большой взрыв («Big Bang» Integration)**

Все или практически все разработанные модули собираются вместе в виде законченной системы или ее основной части, и затем проводится интеграционное тестирование.

Такой подход очень хорош для сохранения времени. Однако если тест кейсы и их результаты записаны не верно, то сам процесс интеграции сильно осложнится, что станет преградой для команды тестирования при достижении основной цели интеграционного тестирования.



Принципы тестирования

Принцип 1 — Тестирование демонстрирует наличие дефектов (Testing shows presence of defects) 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.
Принцип 2 — Исчерпывающее тестирование недостижимо (Exhaustive testing is impossible) 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.

Принципы тестирования:

Принцип 3 — Раннее тестирование (Early testing) 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.

Принцип 4 — Скопление дефектов (Defects clustering) Усилия тестирования должны быть сосредоточены пропорционально ожидаемой, а позже реальной плотности дефектов по модулям. Как правило, большая часть дефектов, обнаруженных при тестировании или повлекших за собой основное количество сбоев системы, содержится в небольшом количестве модулей.

Принципы тестирования

Принцип 5 — Парадокс пестицида (Pesticide paradox) 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 рецензироваться и корректироваться, новые тесты должны быть разносторонними, чтобы охватить все компоненты программного обеспечения, или системы, и найти как можно больше дефектов.

Принципы тестирования

Принцип 6 — Тестирование зависит от контекста (Testing is concept depending)
Тестирование выполняется по-разному в зависимости от контекста. Например, программное обеспечение, в котором критически важна безопасность, тестируется иначе, чем сайт электронной коммерции.

Принцип 7 — Заблуждение об отсутствии ошибок (Absence-of-errors fallacy)
Обнаружение и исправление дефектов не помогут, если созданная система не подходит пользователю и не удовлетворяет его ожиданиям и потребностям.

Методики тестирования

Статическое и динамическое тестирование
Статическое тестирование отличается от динамического тем, что производится без запуска программного кода продукта. Тестирование осуществляется путем анализа программного кода (code review) или скомпилированного кода. Анализ может производиться как вручную, так и с помощью специальных инструментальных средств. Целью анализа является раннее выявление ошибок и потенциальных проблем в продукте. Также к статическому тестированию относится тестирования спецификации и прочей документации.

Методики тестирования

Ручное и автоматизированное

При **ручном тестировании (manual testing)** тестировщики вручную выполняют тесты, не используя никаких средств автоматизации. Ручное тестирование – самый низкоуровневый и простой тип тестирования, не требующий большого количества дополнительных знаний.

Автоматизированное тестирование (automation testing) предполагает использование специального программного обеспечения (помимо тестируемого) для контроля выполнения тестов и сравнения ожидаемого фактического результата работы программы. Этот тип тестирования помогает автоматизировать часто повторяющиеся, но необходимые для максимизации тестового покрытия, задачи.

Методики тестирования

Основные виды автоматизированного тестирования:

- **Автоматизация тестирования кода (Code-driven testing)** – тестирование на уровне программных модулей, классов и библиотек (фактически, автоматические юнит-тесты).
- **Автоматизация тестирования графического пользовательского интерфейса (Graphical user interface testing)** – специальная программа (фреймворк автоматизации тестирования) позволяет генерировать пользовательские события– нажатия клавиш, клики мышкой, и отслеживание реакции программы на эти действия: соответствует ли она спецификации.
- **Автоматизация тестирования API (Application Programming Interface**) – тестирование программного интерфейса программы. Тестируются интерфейсы, предназначенные для взаимодействия, например, с другими программами или с пользователем. Здесь, опять же, как правило, используются специальные фреймворки.

Методики тестирования

Исследовательское (exploratory testing) / Свободное (интуитивное) тестирование (ad hoc testing)

Исследовательское тестирование — это разработка и выполнения тестов в одно и то же время. Что является противоположностью сценарного подхода (с его предопределенными процедурами тестирования, неважно ручными или автоматизированными). Исследовательские тесты, в отличие от сценарных тестов, не определены заранее и не выполняются в точном соответствии с планом.

Разница между ad hoc и exploratory testing в том, что теоретически, ad hoc может провести кто угодно, а для проведения exploratory необходимо мастерство и владение определенными техниками. Обратите внимание, что определенные техники это не только техники тестирования.

Требования в тестировании

Требования — это спецификация (описание) того, что должно быть реализовано.
Требования описывают то, что необходимо реализовать, без детализации технической стороны решения. Что, а не как.

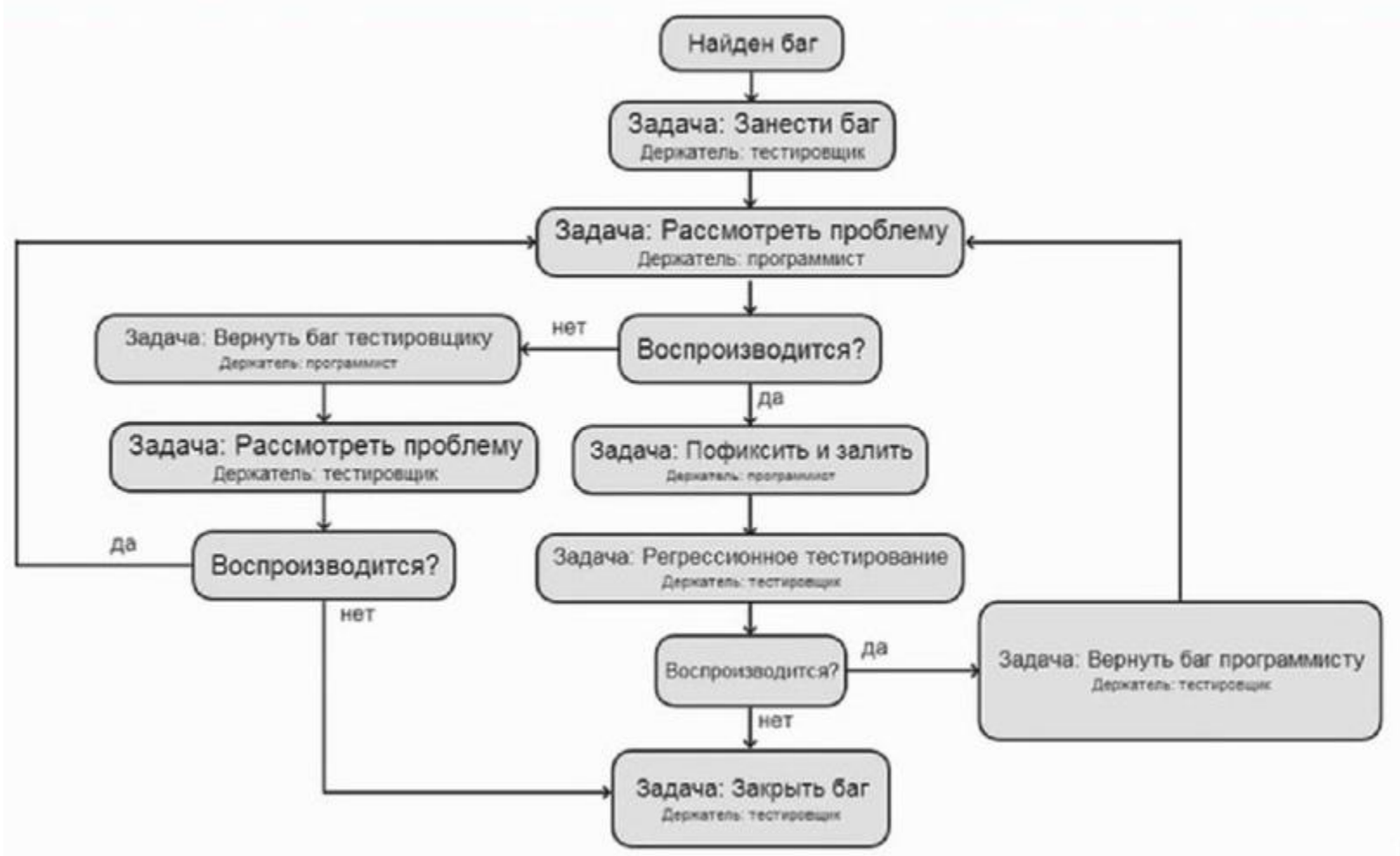
Требования к требованиям:

- Корректность.
- Недвусмысленность.
- Полнота набора требований.
- Непротиворечивость набора требований.
- Проверяемость (тестопригодность).
- Трассируемость.
- Понимаемость.

ЖЦ бага

Новый (New)
Отказ (Rejected)
Назначен (Assigned)
Решен (Fixed)
Переоткрыт (Re-opened)
Закрыт (Closed).

ЖЦ бага с точки зрения команды



ЖЦ версий ПО

Жизненный цикл разработки ПО:

Преальфа (Pre-alpha).
Альфа-версия (Alpha).
Бета-версия (Beta Escrow).
Релиз-кандидат (RC).
Релиз (RTM) .
Пост-релиз (Post-RTM).

QA/QC/Test

Quality Assurance	Quality Control	Тестирование
Комплекс мероприятий, который охватывает все технологические аспекты на всех этапах разработки, выпуска и введения в эксплуатацию программных систем для обеспечения необходимого уровня качества программного продукта	Процесс контроля соответствия разрабатываемой системы предъявляемым к ней требованиям	Процесс, отвечающий непосредственно за составление и прохождение тест-кейсов, нахождение и локализацию дефектов и т.д.
Фокус в большей степени на процессы и средства, чем на непосредственно исполнение тестирования системы	Фокус на исполнение тестирования путем выполнения программы с целью определения дефектов с использованием утвержденных процессов и средств	Фокус на исполнение тестирования как такового
Процессно-ориентированный подход	Продуктно-ориентированный подход	Продуктно-ориентированный подход
Превентивные меры	Корректирующий процесс	Превентивный процесс
Подмножество процессов Software Test Life Cycle – цикла тестирования ПО	Подмножество процессов QA	Подмножество процессов QC

Тестирование — часть QC. QC — часть QA.

СПАСИБО ЗА ВНИМАНИЕ