Вопрос 3. Проверка работоспособности программного обеспечения

1. Стадии разработки программного продукта

Стадии разработки ПО — это этапы, которые проходят команды разработчиков ПО, прежде чем программа станет доступной для широко круга пользователей. Разработка ПО начинается с первоначального этапа разработки (стадия «пре-альфа») и продолжается стадиями, на которых продукт дорабатывается и модернизируется.
Финальным этапом этого процесса становится выпуск на рынок окончательной версии программного обеспечения («общедоступного релиза»).

Различают следующие стадии разработки программного обеспечения:

• **Пре-альфа (Pre-Alpha)** — Начальная стадия разработки, период времени со старта разработки до выхода стадии Альфа. Также так называются программы, не вышедшие еще в стадию альфа или бета, но прошедшие стадию разработки, для первичной оценки функциональных возможностей в действии.

Pre-alpha относится ко всем действиям, выполняемым во время разработки программного проекта перед его формальным тестированием. Эти мероприятия могут включать анализ требований, разработку программного обеспечения и модульное тестирование. В типичной разработке с открытым исходным кодом существует несколько типов предварительных альфа-версий. "Milestone" (*англ.* этап) версии включают в себя определенные функции и выпускаются, как только разработка функционала завершена.

• **Альфа (Alpha)** — внутренняя разработка. Стадия начала тестирования программы в целом специалистами-тестировщиками, обычно не разработчиками программного продукта, но, как правило, внутри организации или сообществе разрабатывающих продукт. Также это может быть стадия добавления новых функциональных возможностей.

На этом этапе разработчики обычно тестируют программное обеспечение с использованием техники «белого ящика». Затем дополнительная проверка выполняется с использованием методов «черного ящика» или «серого ящика» другой группой тестирования. Переход к тестированию методом «черного ящика» внутри организации известен как альфа-релиз и представляет собой метод тестирования программного обеспечения, посредством которого анализируется функциональность ПО без вмешательства в его внутреннюю структуру (программный код).

• **Бета (Beta)** — общественная разработка. Стадия активного тестирования и отладки программы, прошедшей альфа-тестирование (если таковое было). Программы этого уровня могут быть использованы другими разработчиками программного обеспечения для испытания совместимости.

Бета-этап обычно начинается, когда программное обеспечение полнофункционально, но, вероятно, содержит ряд известных или неизвестных ошибок. Программное обеспечение на бета-этапе, как правило, содержит больше ошибок, чем готовое программное обеспечение, а также проблемы с производительностью и может вызывать сбои или потери данных.

Бета-тестирование также концентрирует внимание на снижении негативного воздействия ПО на пользователей, поэтому оно зачастую включает тесты на удобность использования ПО. Процесс доставки бета-версии пользователям называется бета-выпуском,

и обычно это становится первым разом, когда новое ПО становится доступно вне организации, которая его разработала.

• **Релиз-кандидат (Release Candidate)** — бета-версия, которая имеет потенциал к становлению готовым к выпуску конечным продуктом, но только если не выявятся какие-либо значительные ошибки.

• **Релиз (Release) —** этап публичного выпуска программного обеспечения называется релизом. После выпуска программное обеспечение обычно называется «стабильным выпуском». Формальный термин часто зависит от метода выпуска: физических носителей, онлайновой версии или веб-приложения.

Производственный релиз (*англ.* Release to Manufacturing, RTM) — этап, также известный как «выход на золото», который начинается когда программный продукт готов к поставке. Сборка может быть подписана цифровой подписью, позволяя конечному пользователю проверить целостность и подлинность покупки программного обеспечения. Копия сборки RTM отправляется для массового дублирования, если это применимо для данной сборки. RTM предшествует общей доступности (GA), когда продукт становится выпущен публично.

Общая доступность (*англ.* General Availability, GA) — это этап, на котором все необходимые коммерциализационные мероприятия завершены, а программный продукт доступен для приобретения, в зависимости, однако, от языка, региона, наличия электронных средств и доступности носителей. Коммерциализация может включать в себя тесты безопасности и соответствия, а также локализацию и доступность по всему миру.

В некоторых случаях время между RTM и GA может составлять от недели до нескольких месяцев, прежде чем можно будет объявить общедоступный выпуск из-за времени, необходимого для завершения всех коммерческих процессов, требуемых GA. На этом этапе считается, что программа «вышла в прямом эфире». Такая программа считается надежной, свободной от серьезных ошибок, готовой для широкого доступа через интернет или тиражирования на физических носителях.

Сетевой релиз (*англ.* Release to Web RTW) — этот этап релиза с целью распространения использует сеть Интернет. В случае с сетевым релизом изготовитель не производит никаких физических носителей. Сетевые релизы становятся все более распространенными по мере роста использования Интернета.

• **Пост-релиз (Post-RTM**) — издание продукта, у которого есть несколько отличий от RTM и помечается как самая первая стадия разработки следующего продукта. Такие релизы не выпускаются на продажу, а раздаются бета-тестировщикам.

Прекращение поддержки. Когда программное обеспечение больше не продается или не поддерживается разработчиками, то считают, что продукт достиг конца своего срока службы. Но лояльность пользователей может продолжаться в течение времени, даже задолго после того, как платформа определённого ПО устарела (например, Atari ST и Sinclair ZX Spectrum).

2. Уровни Тестирования

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

2. Интеграционное тестирование (Integration Testing) проверяет взаимодействие между компонентами системы после проведения компонентного тестирования. Подходы к интеграционному тестированию:

• **Снизу вверх:** 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.

• **Сверху вниз:** 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.

• **Большой взрыв:** 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.

3. Системное тестирование: основной задачей здесь является проверка как функциональных, так и не функциональных требований в системе в целом. При этом выявляются дефекты, такие как неверное использование ресурсов системы, непредусмотренные комбинации данных пользовательского уровня, несовместимость с окружением, непредусмотренные сценарии использования, отсутствующая или неверная функциональность, неудобство использования и т.д.

4. Приемочное тестирование: 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.

Тестирование программного обеспечения является неотъемлемой частью создания программного продукта. От того, насколько досконально проведены тесты, зависит то, как скоро проект будет сдан окончательно, и будет ли необходимость впоследствии устранять ошибки. Тестирование программного продукта на разных стадиях создания — залог качественного выполнения заказа.

Отсутствие должного тестирования программного продукта может вызвать волну отрицательных отзывов и впоследствии сформировать к программе негативное отношение. Осознавая значимость тестирования ПО, необходимо выполнять его на всех стадиях разработки ПО. Любой продукт тестируется во всех операционных системах и на всех устройствах, на которых предполагается эксплуатация.

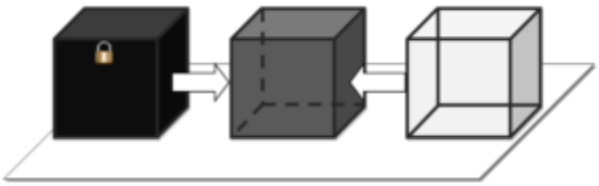
3. Типы тестирования ПО

1	Регрессионное тестирование	Проверка работоспособности программного обеспечения после внесения изменений.
2	Функциональное тестирование	Проверка соответствия продукта функциональным требованиям и спецификациям
3	Нагрузочное тестирование	Обеспечение надежности работы программного

		обеспечения при заданных условиях. Измерение производительности, определение системной конфигурации.
4	Модульное тестирование ПО	Полный цикл тестирования отдельных компонентов на возможность интеграции и использования в составе более крупных систем. Тестирование модулей исходных кодов на соответствие требованиям к оформлению.
5	Оптимизационное тестирование	Устранение узких мест с помощью улучшения алгоритмов, а также использования верных технологий.
6	Тестирование интерфейса	Тестирование пользовательского интерфейса продукта для проверки функциональных характеристик, удобства использования, понятности и соответствия определенному стилю.
7	Анализ исходного кода	Анализ кода в соответствии с предъявляемыми на проекте требованиями.
8	Анализ документации	Анализ спецификаций на полноту и достоверность, проверка пользовательских инструкций и документации программного продукта.
9	Общее (финальное) тестирование	Тестирование методом «черного ящика», основанное на проверке функциональности, которую должен иметь тестируемый продукт в соответствии со спецификацией и документацией.

4. Методы тестирования

Самым высоким уровнем в иерархии подходов к тестированию будет понятие типа, которое может охватывать сразу несколько смежных техник тестирования. То есть, одному типу тестирования может соответствовать несколько его видов. Рассмотрим, для начала несколько типов тестирования, которые отличаются знанием внутреннего устройства объекта тестирования.



Black Box Summary: Мы не знаем, как устроена тестируемая система. Тестирование методом «черного ящика», также известное как тестирование, основанное на спецификации или тестирование поведения – техника тестирования, основанная на работе исключительно с внешними интерфейсами тестируемой системы.

Это тестирование, не предполагающее знания внутреннего устройства компонента или системы. Тест-дизайн, основанный на технике черного ящика – процедура написания или выбора тест-кейсов на основе анализа функциональной или нефункциональной спецификации компонента или системы без знания ее внутреннего устройства.

Почему именно «черный ящик»? Тестируемая программа для тестировщика – как черный непрозрачный ящик, содержания которого он не видит. Целью этой техники является поиск ошибок в таких категориях:

– неправильно реализованные или недостающие функции;
– ошибки интерфейса;
– ошибки в структурах данных или организации доступа к внешним базам данных;
– ошибки поведения или недостаточная производительности системы;

Таким образом, мы не имеем представления о структуре и внутреннем устройстве системы. Нужно концентрироваться на том, *что* программа делает, а не на том, *как* она это делает.

Пример: *Тестировщик проводит тестирование веб-сайта, не зная особенностей его реализации, используя только предусмотренные разработчиком поля ввода и кнопки. Источник ожидаемого результата – спецификация.*

Поскольку это тип тестирования, по определению он может включать другие его виды. Тестирование черного ящика может быть как функциональным, так и нефункциональным. Функциональное тестирование предполагает проверку работы функций системы, а нефункциональное – соответственно, общие характеристики нашей программы.

Техника черного ящика применима на всех уровнях тестирования (от модульного до приемочного), для которых существует спецификация. Например, при осуществлении системного или интеграционного тестирования, требования или функциональная спецификация будут основой для написания тест-кейсов.

Техники тест-дизайна, основанные на использования черного ящика, включают:
– классы эквивалентности;
– анализ граничных значений;
– таблицы решений;
– диаграммы изменения состояния;
– тестирование всех пар.

Преимущества:
– тестирование производится с позиции конечного пользователя и может помочь обнаружить неточности и противоречия в спецификации;
– тестировщику нет необходимости знать языки программирования и углубляться в особенности реализации программы;
– тестирование может производиться специалистами, независимыми от отдела разработки, что помогает избежать предвзятого отношения;
– можно начинать писать тест-кейсы, как только готова спецификация.

Недостатки:
– тестируется только очень ограниченное количество путей выполнения программы;
– без четкой спецификации (а это скорее реальность на многих проектах) достаточно трудно составить эффективные тест-кейсы;
– некоторые тесты могут оказаться избыточными, если они уже были проведены разработчиком на уровне модульного тестирования;

White Box Summary: Нам известны все детали реализации тестируемой программы. Противоположность черного ящика.

Тестирование методом белого ящика (также: прозрачного, открытого, стеклянного ящика; основанное на коде или структурное тестирование) – метод тестирования программного обеспечения, который предполагает, что внутренняя структура/устройство/реализация системы известны тестировщику.

Мы выбираем входные значения, основываясь на знании кода, который будет их обрабатывать. Точно так же мы знаем, каким должен быть результат этой обработки. Знание всех особенностей тестируемой программы и ее реализации – обязательны для этой техники. Тестирование белого ящика – углубление во внутренне устройство системы, за пределы ее внешних интерфейсов.

Тестирование белого ящика – это:

– тестирование, основанное на анализе внутренней структуры компонента или системы.
– тест-дизайн, основанный на технике белого ящика – процедура написания или выбора тест-кейсов на основе анализа внутреннего устройства системы или компонента.
Почему «белый ящик»? Тестируемая программа для тестировщика – прозрачный ящик, содержимое которого он прекрасно видит.

Пример:
Тестировщик, который, как правило, является программистом, изучает реализацию кода поля ввода на веб-странице, определяет все предусмотренные (как правильные, так и неправильные) и не предусмотренные пользовательские вводы, и сравнивает фактический результат выполнения программы с ожидаемым. При этом ожидаемый результат определяется именно тем, как должен работать код программы.

Тестирование методом белого ящика похоже на работу механика, который изучает двигатель машины, чтобы понять, почему она не заводится. Техника белого ящика применима на разных уровнях тестирования – от модульного до системного, но главным образом применяется именно для реализации модульного тестирования компонента его автором.

Преимущества:

– тестирование может производиться на ранних этапах: нет необходимости ждать создания пользовательского интерфейса;

– можно провести более тщательное тестирование, с покрытием большого количества путей выполнения программы.

Недостатки:

– для выполнения тестирования белого ящика необходимо большое количество специальных знаний;
– при использовании автоматизации тестирования на этом уровне, поддержка тестовых скриптов может оказаться достаточно накладной, если программа часто изменяется.

Grey Box

Summary: Нам известны только некоторые особенности реализации тестируемой системы.

Тестирование методом серого ящика – метод тестирования программного обеспечения, который предполагает, комбинацию White Box и Black Box подходов. То есть, внутреннее устройство программы нам известно лишь частично. Предполагается, например, доступ к внутренней структуре и алгоритмам работы ПО для написания максимально эффективных тест-кейсов, но само тестирование проводится с помощью техники черного ящика, то есть, с позиции пользователя. Эту технику тестирования также называют методом полупрозрачного ящика: что-то мы видим, а что-то – нет.

Пример: *Тестировщик изучает код программы с тем, чтобы лучше понимать принципы ее работы и изучить возможные пути ее выполнения. Такое знание поможет написать тест-кейс, который наверняка будет проверять определенную функциональность.*

Техника серого ящика применима на разных уровнях тестирования – от модульного до системного, но главным образом применяется на интеграционном уровне для проверки взаимодействия разных модулей программы.

Другие типы тестирования

Статическое тестирование – тип тестирования, который предполагает, что программный код во время тестирования не будет выполняться. При этом само тестирование может быть как ручным, так и автоматизированным. Статическое тестирование начинается на ранних этапах жизненного цикла ПО и является, соответственно, частью процесса верификации. Для этого типа тестирования в некоторых случаях даже не нужен компьютер – например, при проверке требований.

Большинство статических техник могут быть использованы для «тестирования» любых форм документации, включая вычитку кода, инспекцию проектной документации, функциональной спецификации и требований. Даже статическое тестирование может быть автоматизировано – например, можно использовать автоматические средства проверки синтаксиса программного кода.

Виды статического тестирования: вычитка исходного кода программы, проверка требований.

Динамическое тестирование – тип тестирования, который предполагает запуск программного кода. Таким образом, анализируется *поведение* программы во время ее работы.

Для выполнения динамического тестирования необходимо чтобы тестируемый программный код был написан, скомпилирован и запущен. При этом, может выполняться проверка внешних параметров работы программы: загрузка процессора, использование памяти, время отклика и т.д. – то есть, ее производительность.

Динамическое тестирование является частью процесса валидации программного обеспечения. Кроме того динамическое тестирование может включать разные подвиды, каждый из которых зависит от:

• Доступа к коду (тестирование черным, белым и серым ящиками).

• Уровня тестирования (модульное интеграционное, системное, и приемочное тестирование).

• Сферы использования приложения (функциональное, нагрузочное, тестирование безопасности и пр.).

Позитивное тестирование и негативное тестирование.

Тестируя некую функциональность или программный продукт в целом, мы все время задаем два вопроса: что эта программа или часть программы может делать, а что нет? В данных вопросах и кроется основное различие между позитивным и негативным тестированием.

Позитивный тест – использование данных или тестовых сценариев, которые предусмотрены для нормального функционирования приложения. Как Вы уже догадались, позитивное тестирование служит для подтверждения того, что программный продукт может выполнять то, для чего его разработали.

Негативное тестирование – противоположность позитивного. Его суть заключается в выполнении программой функций или использование данных, которые не предусмотрены ни разработчиками, ни идейным создателем приложения. Например, как отреагирует программа, если в числовое поле ввести буквы. Негативное тестирование используется для

определения таких параметров работы приложения, как: проверка обработки исключительных ситуаций, исследование поведения приложения в незапланированном режиме работы и других возможных нефункциональных атрибутов.

Зачастую, тестировщики придерживаются следующего порядка проверки приложения:

1. Дымное тестирование.
2. Позитивные тесты.
3. Негативные тесты.

Дымное тестирование служит для быстрой проверки работоспособности ПО, позитивные тесты помогают удостоверится в том, что приложение делает именно то, что от него и ожидают, а негативное тестирование характеризует систему, как работоспособный, устойчивый продукт.

5. Виды Тестирования

Функциональные виды тестирования

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

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)

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)
-

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

Тестирование пользовательского интерфейса (GUI Testing) — функциональная проверка интерфейса на соответствие требованиям — размер, шрифт, цвет, consistent behavior.

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.

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

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. Также одной из задач при стрессовом тестировании может быть оценка деградации производительности, таким образом цели стрессового тестирования могут пересекаться с целями тестирования производительности.

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.

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:

- 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».

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.

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.)

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.

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.

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

В чем разница между regression testing и re-testing?

- Re-testing — проверяется исправление багов;

- Regression testing — проверяется то, что исправление багов, а также любые изменения в коде приложения, не повлияли на другие модули ПО и не вызвало новых багов.

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.

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.

Литература:

1. https://www.edsd.ru/ru/uslugi/testirovanie_po
2. Тестирование. Фундаментальная теория / https://qa-uide.ru/forums/topic/teorija_testirovanija/
3. Обзор частых вопросов на собеседованиях и ответы на них / https://habr.com/ru/post/257529/