**\*Основні поняття (якість, вимога, дефекти і причини їх виникнення, severity, priority), цілі тестування.**

**Якість**

Якість програмного забезпечення — характеристика програмного забезпечення, ступінь відповідності ПЗ до вимог. При цьому вимоги можуть трактуватись по-різному, що породжує декілька незалежних визначень терміну. Частіше за все, використовують визначення ISO 9001, згідно з яким якість — це «ступінь відповідності наявних характеристик вимогам».

**Вимоги**

Вимоги до програмного забезпечення — набір вимог щодо властивостей, якості та функцій програмного забезпечення, що буде розроблено, або знаходиться у розробці. Вимоги визначаються в процесі аналізу вимог та фіксуються в специфікації вимог, діаграмах прецедентів та інших артефактах процесу аналізу та розробки вимог.

Види вимог за рівнями

Бізнес-вимоги — визначають призначення ПЗ, можуть описуватися в документі про бачення (англ. vision) та документі про межі проекту (англ. scope).

Вимоги користувача — визначають набір завдань користувача, які повинна вирішувати програма, а також сценарії їхнього вирішення в системі. Ці вимоги можуть мати вигляд тверджень, варіантів використання, історій користувача, сценаріїв взаємодії.

Функціональні вимоги — визначають «що» повинен робити програмний продукт. Ці вимоги описуються в документі Специфікації програмного забезпечення (англ. SRS).

*Види вимог за характером*

Функціональний характер — вимоги до поведінки системи

Бізнес-вимоги

Вимоги користувача

Функціональні вимоги

Нефункціональний характер — вимоги до характеру поведінки системи

Бізнес-правила — визначають обмеження, що витікають з предметної області.

Системні вимоги — вимоги до програмних інтерфейсів, надійності, обладнанню.

Атрибути якості

Зовнішні системи та інтерфейси

Обмеження

Джерела вимог

Законодавство

Вимоги стандартів

Бізнес-процеси

Очікування на бачення користувачів системи

**Severity и priority**

Серьезность (Severity) **-** это атрибут, характеризующий влияние дефекта на работоспособность приложения.

Приоритет (Priority) **-** это атрибут, указывающий на очередность выполнения задачи или устранения дефекта. Можно сказать, что это инструмент менеджера по планированию работ. Чем выше приоритет, тем быстрее нужно исправить дефект.

При управлении пожеланиями и дефектами нас интересуют следующие вопросы:

есть ли замечания к поведению программы, на которые необходимо обратить внимание, то есть есть ли какие-то важные дефекты или доработки? Отвечает на этот вопрос признак важности, то есть **severity**.

в каком порядке нам нужно реализовывать доработки или исправлять ошибки, чтобы соответствовать ожиданиям заказчика или пользователей продукта? Отвечает на этот вопрос признак приоритета, то есть **priority**.

**Цілі тестування**

Цілі тестування:

\* Перевірити взаємодії між об'єктами.

\* Перевірити належну інтеграцію всіх компонентів програмного забезпечення.

\* Щоб переконатися, що всі вимоги були правильно виконані.

\* Щоб визначити та переконатись що дефекти будуть розглянуті до розгортання програмного забезпечення.

\* Переконатись, що всі дефекти виправлені, повторно перевірені та закриті.

Раціональний уніфікований процес пропонує ітеративний підхід, а це означає, що тестування відбувається протягом всього проекту. Це дозволяє виявляти дефекти якомога раніше, що радикально знижує вартість виправлення дефекту. Тести проводяться за чотирма вимірами якості:надійності, функціональності, продуктивності додатків і продуктивності системи. Для кожного з цих вимірів критеріїв якості, процес описує як пройти життєвий цикл планування, проектування, виконання і оцінки тесту.

**\*Підходи до тестування**

Тестування білої скриньки

У термінології професіоналів тестування (програмного й деякого апаратного забезпечення), фрази «тестування білого ящика» і «тестування чорного ящика» відносяться до того, чи має розробник тестів доступ до вихідного коду ПЗ, що тестується, або ж тестування виконується через інтерфейс користувача або прикладний програмний інтерфейс, наданий модулем, що тестується. При тестуванні білого ящика (англ. white-box testing, також говорять — прозорого ящика), розробник тесту має доступ до вихідного коду й може писати код, що пов'язаний з бібліотеками ПЗ, що тестується. Це типово для юніт-тестування (англ. unit testing), при якому тестуються тільки окремі частини системи. Воно забезпечує те, що компоненти конструкції — працездатні й стійкі до певного ступеня.

White-Box Testing (или так называемое Structural testing): один из видов который использует внутреннюю часть данной системы для разработки тестовых случаев основанных на внутренней структуре. Данные для тестирования определяются путем изучения логики системы, без учета системных требований. Преимущество «белого ящика» состоит в том, что основное внимание уделяется коду системы.

Тестування чорної скриньки

При тестуванні чорного ящика (англ. black-box testing), тестер має доступ до ПЗ тільки через ті ж інтерфейси, що й замовник або користувач, або через зовнішні інтерфейси, що дозволяють іншому комп'ютеру або іншому процесу підключитися до системи для тестування. Наприклад, модуль, що тестується, може віртуально натискати клавіші або кнопки миші в програмі, що тестується, за допомогою механізму взаємодії процесів, із упевненістю в тім, що ці події викликають той же відгук, що й реальні натискання клавіш і кнопок миші.

Black-Box Testing (или так называемое Functional testing): один из видов тестирования разрабатываемый на основе функциональности системы, то есть, тестеры знают информацию корректных и некорректных данных которые подаются на вход системы и сравнивают их с выходными данными, но ничего не знаю о том как работает сама система. Одним из основных преимуществ «черного ящика» в том, что тестирование, направленное на то, что бы проверить правильную функциональную работу системы.

**Gray-Box Testing** (или так называемое Functional and Structural testing)- это сочетание Black-Box и White-Box тестирования. Команда тестирования исследует требования спецификаций и общается с разработчиками, что бы понять внутреннюю структуру системы. Преимущество этого метода состоит в том, что многие «кейсы» смогут быть убранными.

**\*Види тестування (UI testing, Acceptance Testing, Smoke Testing, Regression Testing, Functional Testing).**

UI testing - тестування користувальницького інтерфейсу

Acceptance - это тестирование что проводится, что бы узнать удовлетворяет ли система критериям достижения определенного уровня или приема заказчиком. Проводится на основе обычных «тест кейсов».

Smoke Testing (англ. Smoke testing, димове тестування) в тестуванні програмного забезпечення означає мінімальний набір тестів на явні помилки. Димовий тест зазвичай виконується самим програмістом; програму, що не пройшла такий тест, не має сенсу віддавати на більш глибоке тестування.

Regression (или регрессионное тестирование) - это тестирование участков кода с учетом изменений, внесенных в ходе, исправления багов или друг нюансов. Этот метод включают повторные прохождение тестов, и просмотр новых багов которые могли появится изменения кода.

Functional (или функциональное тестирование) **–** это тестирование в котором проверяется правильность выполнения работы функциональных требований. В свою очередь требования дают понять как система должна работать.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Load (ил нагрузочное тестирование)- это тестирование системы под действием нагрузки, чтобы посмотреть, в какой момент система снижает время отклика или нет.

Stress (или стресс тестирование)- это тестирование проводится для проверки того, насколько система в целом работоспособны в условиях высокой нагрузки(стресса), также оценить способность системы к восстановлению после работы в стрессовом режиме.

Performance (или тестирование производительности) – термин тестирование производительности часто используется наравне с Stress(Стресс) и Load(Нагрузочное) тестированием. Особой разницы между ними нет и все они настроены на то, что бы проверить производительность системы при добавлении нагрузок.

Integration (интеграционное) – это тестирование комбинированных частей приложения, чтобы определить, правильно ли они функционируют вместе. Этот вид тестирования является особенно актуальный для: клиент/серверных и распределенных систем.

Security (или тестирование безопасности)- это тестирование, предназначенное для проверки защиты системы от несанкционированного внутреннего или внешнего доступа, нанесения умышленное повреждения и т.д.

Installation (инсталяции) – это тестирование которое применяется для проверки корректной установки системы. В него могут входить и удаления и обновления этой же системы.

Sanity Testing - Узконаправленное тестирование достаточное для доказательства того, что конкретная функция работает согласно заявленным в спецификации требованиям.

Ad-hoc (інтуїтивне тестування, ad hoc testing) — тестування без тест плану та документації, що базується на методиці передбачення помилки та власному досвіді тестера.

Види тестової документації

План Тестування (Test Plan) — це документ, що описує весь обсяг робіт з тестування, починаючи з опису об'єкта, стратегії, розкладу, критеріїв початку і закінчення тестування, до необхідного в процесі роботи обладнання, спеціальних знань, а також оцінки ризиків з варіантами їх вирішення.

Хороший тест план должен как минимум отвечать на следующие вопросы:

- Что надо тестировать?

описание объекта тестирования: системы, приложения, оборудование

- Что будете тестировать?

список функций и описание тестируемой системы и её компонент

- Как будете тестировать?

стратегия тестирования, а именно: виды тестирования и их применение по отношению к тестируемому объекту

- Когда будете тестировать?

последовательность проведения работ: подготовка (Test Preparation), тестирование (Testing), анализ результатов (Test Result Analisys) 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

требовния 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)

Ответив в своем тест плане на вышеперечисленные вопросы, можно считать, что у вас на руках уже есть хороший черновик документа по планированию тестирования. Далее, чтобы документ приобрел более менее серьезный вид, предлагаю дополнить его следующими пунктами:

Окружение тестируемой системы

Необходимое для тестирования оборудование и программные средства

Риски и их разрешение

Тест дизайн (Test Design) — це етап процесу тестування програмного забезпечення, на якому проектуються і створюються тестові випадки (тест кейси), відповідно до визначених раніше критеріями якості та цілями тестування.

Тестовий випадок (Test Case) — це документ, що описує сукупність кроків, конкретних умов і параметрів, необхідних для перевірки реалізації тестованої функції або її частини.

Баг/Дефект Репорт (Bug Report) — це документ, що описує ситуацію або послідовність дій (Steps), що призвела до некоректної роботи об'єкта тестування (Misbehavior), із зазначенням причин та очікуваного результату (Expected Result).

Тестове Покриття (Test Coverage) — це одна з метрик оцінки якості тестування, що представляє із себе щільність покриття тестами вимог або коду, що виконується.

Знає основи і принципи роботи мережі та мережевих протоколів

Транспортний рівень

SPX

TCP

UDP

Прикладний рівень

DHCP

DNS

FTP

HTTP

HTTPS

IMAP

NTP

POP3

SMTP

**CVS (від англ. Concurrent Versions System)** - система контролю версій.

CVS (Concurrent Versions System, «Система Одновременных Версий») — программный продукт, относящийся к разряду систем управления версиями (англ. version control system). Хранит историю изменений определённого набора файлов, как правило, исходного кода программного обеспечения, и облегчает совместную работу группы людей (часто — программистов) над одним проектом. CVS популярна в мире открытого ПО. Система распространяется на условиях лицензии GNU GPL.

**Team Foundation Server (сокр. TFS)** — продукт корпорации Microsoft, представляющий собой комплексное решение, объединяющее в себе систему управления версиями, сбор данных, построение отчетов, отслеживание статусов и изменений по проекту и предназначенное для совместной работы над проектами по разработке программного обеспечения.

**Microsoft Visual SourceSafe (Visual SourceSafe, VSS)** — программный продукт компании Майкрософт, файл-серверная система управления версиями, предназначенная для небольших команд разработчиков. VSS позволяет хранить в общем хранилище файлы, разделяемые несколькими пользователями, для каждого файла хранится история версий.

VSS входил в состав пакета Microsoft Visual Studio и был интегрирован с продуктами этого пакета. Доступен только для платформы Windows. Версию для Unix поддерживает компания MainSoft.

**SecureCRT** — коммерческий SSH и Telnet клиент и эмулятор терминала от компании VanDyke Software.

**Возможности:**

Графический пользовательский интерфейс с поддержкой вкладок и конфигурируемых сеансов.

**Process Explorer** — бесплатная программа для Microsoft Windows, созданная Sysinternals, и затем приобретённая Microsoft Corporation.

Process Explorer — приложение для мониторинга процессов в системе. Используя эту программу можно не только отследить какой-либо процесс, но и уточнить такие данные, как используемые этим процессом файлы и папки.

Выводимая Process Explorer информация значительно подробнее, чем у Диспетчера задач Windows. В числе наиболее интересных возможностей приложения — возможность быстро уточнить какому процессу принадлежит окно на рабочем столе.

Process Explorer работает под Windows XP и выше, включая 64-битные версии.

**Возможности:**

Иерархическое отображение процессов.

Возможность идентификации системных процессов (например, является ли процесс svchost.exe системным или "левым")

Отображает иконку и компанию производителя каждого процесса.

Изменяемый диапазон измерений загрузки CPU и графические индикаторы.

Возможность «заморозить» любой процесс.

Возможность управления (запуск, пауза, остановка) потоками (нитями) процесса.

Возможность вывести окно, принадлежащее тому или иному процессу поверх остальных.

Возможность закрытия древа процессов.

Возможность в реальном режиме времени менять приоритет и то, на каком ядре процессора будет выполняться тот или иной процесс.

Возможность проверки сертификата файла процесса.

Возможность заменять системный диспетчер задач по тем же горячим клавишам.

Для каждого объекта, имеющего ACL, отображается вкладка «Безопасность» (с версии 12.04)

HTTP Analyzer - программа-сниффер, позволяющая вам производить мониторинг и захватывать HTTP/HTTPS трафик в реальном времени. Позволяет отслеживать и отображать широкий диапазон информации, включая заголовки, содержание, cookies, строки запросов, отправку данных, потоки запросов и ответов, перенаправления URL и многое другое. Программа дает возможность удобно просматривать потоковые запросы. Кроме этого, данная программа обеспечивает кэширование информации и очистку сессии, информацию о статусе HTTP кода и имеет несколько настроек по фильтрации.

По заявлению авторов ее используют в крупных компаниях, таких как Microsoft, Cisco, AOL и Google - это сложно проверить, но будем думать что это правда.

Программа доступна в двух версиях - в качестве отдельного полнофункционального приложения и в качестве дополнения для Internet Explorer. Кроме IE, программа поддерживает работу с браузерами Opera, Firefox и другими.

Основные возможности программы HTTP Analyzer:

Интеграция с Internet Explorer и Firefox. Больше не требуется переключения между окнами - HTTP Analyzer показывает HTTP / HTTPS трафик и веб-страницы в одном окне.

Поддержка HTTPS. Показывает вам не зашифрованные данные, передаваемые через HTTPS / SSL соединения, с таким же уровнем детализации, как HTTP.

Просмотр широкого диапазона данных.

Отображение временных диаграмм страниц в режиме реального времени и уровней запросов.

Встроенная поддержка Flash Remoting.

Автоматизация интерфейса. Автоматизированная библиотека HTTP Analyzer поставляется в виде компонентов COM.

Удобный просмотр потоковых запросов.

Очистка cookie и кеша браузера.

Анализ интересующих вас веб-сайтов.

Помощь в изучении HTTP-протокола.

Отладка веб-приложений.

Повышение продуктивности веб-разработки.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Verification - постоянно выполняемый аналитический процесс проверки того, что разработка находится на правильном пути: каждый этап разработки является корректным, необходимым (не лишним) и удовлетворяет потребности следующего этапа. Проводится на всех этапах разработки. На каждом этапе следует убеждаться, что сделали именно то, что планировали, и это соответствует общей логике разработке. «Мы создаем систему правильно».

Validation (проверка правильности) — процесс проверки того, что реализованная система удовлетворяет предъявленным требованиям и работает так, как предполагалось. «Мы создаем правильную систему».