**Требование (requirement48) —** описание того, какие функции и с соблюдением каких условий должно выполнять приложение в процессе решения полезной для пользователя задачи.

**Источники и пути выявления требований:**

* Интервью
* Работа с фокусными группами
* Анкетирование
* Семинары и мозговой штурм
* Наблюдение
* Прототипирование
* Анализ документов
* Моделирование процессов и взаимодействий
* Самостоятельное описание

**Уровни требований:**

* Бизнес требования
* Пользовательские
* Функциональные

**Требования к требованиям**

**Завершённость (**completeness85). Требование является полным и закончен-ным с точки зрения представления в нём всей необходимой информации, ничто не пропущено по соображениям «это и так всем понятно».

**Атомарность, единичность (**atomicity86). Требование является атомарным, если его нельзя разбить на отдельные требования без потери завершённости и оно описывает одну и только одну ситуацию.

**Непротиворечивость, последовательность (**consistency87). Требование не должно содержать внутренних противоречий и противоречий другим требованиям и документам.

**Недвусмысленность (**unambiguousness88, clearness). Требование должно быть описано без использования жаргона, неочевидных аббревиатур и расплывча-тых формулировок, должно допускать только однозначное объективное понимание и быть атомарным в плане невозможности различной трактовки сочетания отдель-ных фраз.

**Выполнимость (**feasibility89). Требование должно быть технологически вы-полнимым и реализуемым в рамках бюджета и сроков разработки проекта.

**Обязательность, нужность (**obligatoriness90) и **актуальность** (up-to-date). Если требование не является обязательным к реализации, оно должно быть просто исключено из набора требований. Если требование нужное, но «не очень важное», для указания этого факта используется указание приоритета (см. «проранжирован-ность по…»). Также исключены (или переработаны) должны быть требования, утра-тившие актуальность.

**Прослеживаемость (**traceability91, 92). Прослеживаемость бывает вертикаль-ной (vertical traceability93) и горизонтальной (horizontal traceability94). Вертикальная позволяет соотносить между собой требования на различных уровнях требований, горизонтальная позволяет соотносить требование с тест-планом, тест-кейсами, ар-хитектурными решениями и т.д.

**Модифицируемость (**modifiability97). Это свойство характеризует простоту внесения изменений в отдельные требования и в набор требований. Можно гово-рить о наличии модифицируемости в том случае, если при доработке требований искомую информацию легко найти, а её изменение не приводит к нарушению иных описанных в этом перечне свойств.

**Проранжированность по важности, стабильности, срочности (**ranked98 for importance, stability, priority). Важность характеризует зависимость успеха про-екта от успеха реализации требования. Стабильность характеризует вероятность того, что в обозримом будущем в требование не будет внесено никаких изменений. Срочность определяет распределение во времени усилий проектной команды по реализации того или иного требования.

**Корректность (**correctness99) и **проверяемость** (verifiability100). Фактически эти свойства вытекают из соблюдения всех вышеперечисленных (или можно сказать, что они не выполняются, если нарушено хотя бы одно из вышеперечисленных). В дополнение можно отметить, что проверяемость подразумевает возможность создания объективного тест-кейса (тест-кейсов), однозначно показывающего, что требование реализовано верно и поведение приложения в точности соответствует требованию.

**Техники тестирования требований**

* Взаимный просмотр (беглый, технический, Формальная инспекция)
* Вопросы
* Тест-кейсы и чек-листы
* Исследование поведения системы
* Рисунки
* Прототипирование

**Уровни тестирования**

* Модульное, компонентное, Unit
* Интеграционное
* Системное
* Пользовательское

**Методы или типы тестирования**

***По запуску кода*** на исполнение:

* Статическое тестирование — без запуска.
* Динамическое тестирование — с запуском.

***По доступу к коду и архитектуре*** приложения:

* Метод белого ящика — доступ к коду есть.
* Метод чёрного ящика — доступа к коду нет.
* Метод серого ящика — к части кода доступ есть, к части — нет.

***По степени автоматизации:***

* Ручное тестирование — тест-кейсы выполняет человек.
* Автоматизированное тестирование — тест-кейсы частично или полностью выполняет специальное инструментальное средство.

***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.

***По позитивности*** тестовых сценариев:

* Позитивное
* Негативное

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

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

**Виды тестирования.**

***По степени формализации*** (наличия документации):

* Тестирование на основе тест-кейсов
* Исследовательское тестирование
* Интуитивное тестирование (ad-hoc)

***По объекту тестирования:***

**Функциональное**:

* Функциональное тестирование
* Тестирование пользовательского интерфейса (GUI)
* Взаимодействия (API)

**Нефункциональное:**

***Тестирование на отказ и восстановление***

**Тестирование производительности** (performance testing207) — исследование показателей скорости реакции приложения на внешние воздействия при различной по характеру и интенсивности нагрузке. В рамках тестирования производительности выделяют следующие подвиды:

* + ***Нагрузочное тестирование*** (load testing208, capacity testing209) — исследование способности приложения сохранять заданные показатели качества при нагрузке в допустимых пределах и некотором превышении этих пределов (определение «запаса прочности»).
  + ***Стрессовое тестирование*** (stress testing212) — исследование поведения приложения при нештатных изменениях нагрузки, значительно превышающих расчётный уровень, или в ситуациях недоступности значительной части необходимых приложению ресурсов.
  + ***Тестирование масштабируемости*** (scalability testing210) — исследование способности приложения увеличивать показатели производительности в соответствии с увеличением количества доступных приложению ресурсов.
* ***Объёмное тестирование*** (volume testing211) — исследование производительности приложения при обработке различных (как правило, больших) объёмов данных.
* ***Конкурентное тестирование*** (concurrency testing214) — исследование поведения приложения в ситуации, когда ему приходится обрабатывать большое количество одновременно поступающих запросов, что вызывает конкуренцию между запросами за ресурсы (базу данных, память, канал передачи данных, дисковую подсистему и т.д.).
* ***Тестирование стабильности.*** Представляет собой проверку нашего приложения при длительной работе.

**Тестирование удобства использования**(usability174 testing) — тестирование, направленное на исследование того, насколько конечному пользователю понятно, как работать с продуктом.

**Инсталляционное тестирование** (installation testing, installability testing164) — тестирование, направленное на выявление дефектов, влияющих на протекание стадии инсталляции (установки) приложения.

**Тестирование удобства использования** (usability174 testing) — тестирова-ние, направленное на исследование того, насколько конечному пользова-телю понятно, как работать с продуктом (understandability175, learnability176, op-erability177), а также на то, насколько ему нравится использовать продукт (attractiveness178).

**Тестирование доступности** (accessibility testing179) — тестирование, направ-ленное на исследование пригодности продукта к использованию людьми с ограниченными возможностями (слабым зрением и т.д.).

**Тестирование безопасности** (security testing184) — тестирование, направ-ленное на проверку способности приложения противостоять злонамеренным попыткам получения доступа к данным или функциям, права на доступ к ко-торым у злоумышленника нет.

**Тестирование интернационализации** (internationalization testing, i18n testing, globalization186 testing, localizability187 testing) — тестирование, направленное на проверку готовности продукта к работе с использованием различных языков и с учётом различных национальных и культурных особенностей.

**Тестирование локализации** (localization testing188, l10n) — тестирование, направленное на проверку корректности и качества адаптации продукта к ис-пользованию на том или ином языке с учётом национальных и культурных особенностей.

**Тестирование совместимости** (compatibility testing, interoperability testing189) — тестирование, направленное на проверку способности приложения работать в указанном окружении.

**Тестирование данных** (data quality196 testing) и баз данных (database integrity testing197) — два близких по смыслу вида тестирования, направленных на исследование таких характеристик данных, как полнота, непротиворечивость, целостность, структурированность и т.д.

**Тестирование использования ресурсов** (resource utilization testing198, efficiency testing199, storage testing200) — совокупность видов тестирования, проверяющих эффективность использования приложением доступных ему ресурсов и зависимость результатов работы приложения от количества доступных ему ресурсов.

**Тестирование надёжности** (reliability testing204) — тестирование способности приложения выполнять свои функции в заданных условиях на протяжении заданного времени или заданного количества операций.

**Тестирование восстанавливаемости** (recoverability testing205) — тестирова-ние способности приложения восстанавливать свои функции и заданный уровень производительности, а также восстанавливать данные в случае воз-никновения критической ситуации, приводящей к временной (частичной) утрате работоспособности приложения.

**Тестирование отказоустойчивости** (failover testing206) — тестирование, заключающееся в эмуляции или реальном создании критических ситуаций с целью проверки способности приложения задействовать соответствующие механизмы, предотвращающие нарушение работоспособности, производи-тельности и повреждения данных.

**Связанные с изменениями виды тестирования:**

* **Дымовое тестирование (Smoke Testing, Build Verification**

**Testing)** — проверка самой важной, самой ключевой функциональности, неработоспособность которой делает бессмысленной саму идею использования приложения. Первый прогон программы (после написания или после внесения существенных изменений). Как правило, используется для определения, готова ли программа для проведения более обширного тестирования.

* **Санитарное тестирование или проверка согласованности/исправности (Sanity Testing)** — это узконаправленное тестирование достаточное для доказательства того, что конкретная функция работает согласно заявленным в спецификации требованиям.
* **Регрессионное тестирование** (regression testing165) — тестирование, направленное на проверку того факта, что в ранее работоспособной функци-ональности не появились ошибки, вызванные изменениями в приложении ский человеко-месяц»166 писал: «Фундаментальная проблема при сопровож-дении программ состоит в том, что исправление одной ошибки с большой вероятностью (20–50 %) влечёт появление новой».
* **Повторное тестирование** (re-testing167, confirmation testing) — выполнение тест-кейсов, которые ранее обнаружили дефекты, с целью подтверждения устранения дефектов.

Типичные общие **сценарии используются** в том случае, когда не существует явных предпосылок к реализации иной стратегии. Такие сценарии могут видоизменяться и комбинироваться (например, весь «типичный общий сценарий 1» можно повторять на всех шагах «типичного общего сценария 2»).

Типичный общий сценарий 1:

1) Дымовое тестирование{76}.

2) Тестирование критического пути{77}.

3) Расширенное тестирование{78}. o Типичный общий сценарий 2:

1) Модульное тестирование{74}.

2) Интеграционное тестирование{74}.

3) Системное тестирование{75}. o Типичный общий сценарий 3:

1) Альфа-тестирование{81}.

2) Бета-тестирование{81}.

3) Гамма-тестирование{81}.

**Чек-листы, тест-кейсы, наборы тест-кейсов**

**Чек-лист** (checklist280) — набор идей [тест-кейсов]. Последнее слово не зря взято в скобки281, т.к. в общем случае чек-лист — это просто набор идей: идей по тестированию, идей по разработке, идей по планированию и управлению — любыхидей.

**Тест-кейс** (test case285) — набор входных данных, условий выполнения и ожидаемых результатов, разработанный с целью проверки того или иного свойства или поведения программного средства.



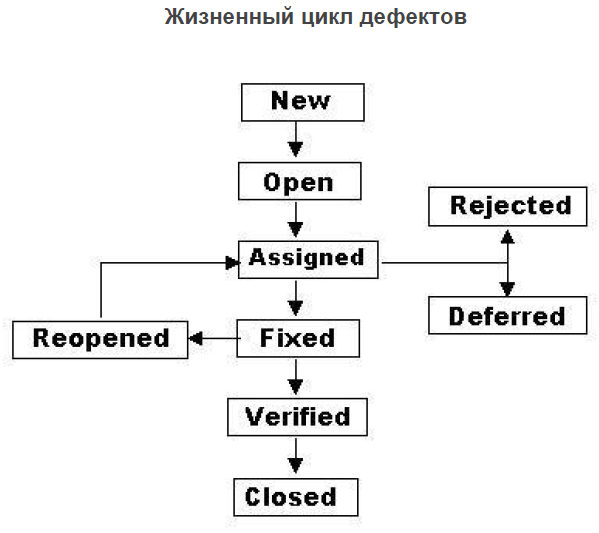
**Набор тест-кейсов** (test case suite301, test suite, test set) — совокупность тест-кейсов, выбранных с некоторой общей целью или по некоторому общему признаку. Иногда в такой совокупности результаты завершения одного тест-кейса становятся входным состоянием приложения для следующего тест-кейса.

В общем случае наборы тест-кейсов можно разделить на свободные (поря-док выполнения тест-кейсов не важен) и последовательные (порядок выполнения тест-кейсов важен).

**Ошибка (**error307, mistake) — действие человека, приводящее к некоррект- ным результатам.

**Дефект (**defect308, bug, problem, fault) — недостаток в компоненте или си- стеме, способный привести к ситуации сбоя или отказа.

**Сбой (**interruption309) или **отказ** (failure310) — отклонение поведения си- стемы от ожидаемого.



**Серьезность (**severity) показывает степень ущерба, который наносится проекту существованием дефекта.

В общем случае выделяют следующие градации важности:

* Критическая (critical) — существование дефекта приводит к масштабным последствиям катастрофического характера, например: потеря данных, раскрытие конфиденциальной информации, нарушение ключевой функциональности приложения и т.д.
* Высокая (major) — существование дефекта приносит ощутимые неудобства многим пользователям в рамках их типичной деятельности, например: недоступность вставки из буфера обмена, неработоспособность общепринятых клавиатурных комбинаций, необходимость перезапуска приложения при выполнении типичных сценариев работы.
* Средняя (medium) — существование дефекта слабо влияет на типичные сценарии работы пользователей, и/или существует обходной путь достижения цели, например: диалоговое окно не закрывается автоматически после нажатия кнопок «OK»/«Cancel», при распечатке нескольких документов подряд не сохраняется значение поля «Двусторонняя печать», перепутаны направления сортировок по некоему полю таблицы.
* Низкая (minor) — существование дефекта редко обнаруживается незначительным процентом пользователей и (почти) не влияет на их работу, например: опечатка в глубоко вложенном пункте меню настроек, некое окно сразу при отображении расположено неудобно (нужно перетянуть его в удобное место), неточно отображается время до завершения операции копирования файлов.
* Тривиальная (trivial)

**Тест-план** (test plan327) — документ, описывающий и регламентирующий перечень работ по тестированию, а также соответствующие техники и под-ходы, стратегию, области ответственности, ресурсы, расписание и ключе-вые даты.