Раздаточный материал к теме 2 - Тестирование документации и требований

**Требования (requirements)** − это подробное изложение функционального наполнения системы. Требования должны быть независимы от внутренней архитектуры приложения, т.е. должны описывать то, что необходимо заказчику без деталей реализации (what, not how).

**Уровни требований**

|  |  |  |
| --- | --- | --- |
| Уровень | Требования  *В чём (как) выражаются* | Что описывают |
| 1 | Бизнес требования  *Общее видение и обзорная документация* | Зачем вообще нужен ПП и что с его помощью будет делаться. Например: "Нам нужен инструмент, извлекающих бизнес-информацию из различных источников и представляющий её в виде диаграмм и таблиц" |
| 2 | Пользовательские требования  *Use Cases* | Задачи, которые пользователь может выполнять с помощью ПП, и реакцию ПП. Например: "Когда пользователь заходит в систему, должен появляться текст приветствия" |
| 3 | Функциональные и нефункциональные требования  *Требования к ПО* | Функциональные требования описывают **поведение системы**, например: "В процессе инсталляции программа должна проверять остаток свободного места на всех несъёмных накопителях".  Нефункциональные требования описывают специфические **свойства системы**, такие как: требования к удобству использования, безопасности, надёжности, расширяемости и т.д. Например: "При одновременной работе с системой 1000 пользователей, минимальное время между возникновением сбоя должно быть более или равно десяти тысячам часов". |

**Необходимые свойства хорошего требования.** Требование должно быть:

* Завершённым. Все важные аспекты должны быть включены. Ничто не должно быть оставлено для "будущего определения" (2BD).
* Непротиворечивым. Требование не должно содержать противоречий как внутри себя, так и с другими требованиями.
* Корректным. Требование должно чётко указывать на то, что должно выполнять приложение. Недопустимо при написании требования предполагать, что что-то окажется очевидным. Каждый человек понимает это "очевидное" по-своему, и в итоге система получится не такой, как задумывалось.
* Недвусмысленным. Требование не должно допускать разночтений.
* Проверяемым. Требование должно быть сформулировано так, чтобы существовали способы однозначной проверки − выполнено требование или нет. Например, требование "Приложение должно быстро запускаться" непроверяемо. Что значит − быстро? За секунду, за минуту, за час? В то же время требование "Приложение должно запускаться не более, чем за пять секунд на компьютере с процессором 3GHz, оперативной памятью 1Gb под управлением ОС WinXP" вполне проверяемо.

**Наборы требований должны быть:**

* Модифицируемыми. Структура и стиль набора требований должны быть такими, чтобы набор требований можно было легко модифицировать. Избыточность должна отсутствовать. Должно быть построено корректное содержание всего документа.
* Прослеживаемыми. У каждого требования должен быть уникальный идентификатор, по которому на это требование можно сослаться.
* Проранжированными по важности. Для каждого требования должен быть указан уровень его важности.

**Типичные проблемы с требованиями:**

* По завершённости. Упущены нефункциональные составляющие требования. Присутствуют скрытые предположения. Присутствуют слишком общие утверждения.
* По непротиворечивости. Требования противоречат явным образом друг другу. Части требований противоречат друг другу. Требование противоречит здравому смыслу, например: "Разослать сообщения всем удалённым пользователям".
* По корректности. Опечатки. Проблемы, вызванные использованием Copy-Paste. Требования, приводящие к избыточным расходам на разработку, при этом не дающие соответствующей отдачи. Технически невыполнимые требования. Случаи требований к дизайну, а не (не)функциональным особенностям ПП.
* По недвусмысленности. Требование должно однозначно трактоваться. Например: "Функция NNN является опциональной". Разработчик может посчитать, что эту функцию можно не реализовывать. Пользователь будет уверен, что эта функция идёт как "бонус". Отдел продаж может посчитать, что эту функцию можно включить в список функций расширенного пакета, и взять за это больше денег. Пример плохого требования: "Проверить, что почта отправлена должным образом". Что такое "должный образ"?
* По проверяемости. "В приложении должно быть ноль ошибок" ­­– это гарантировать невозможно. "Приложение должно быть дружественным к пользователю". Невозможно написать тест, проверяющий такое требование.
* По ранжированности. "Все требования одинаково важны!" Это не так. Есть небольшой процент критических требований, средний процент требований средней важности и большой процент мелких требований в стиле "Эта полосочка должна быть зелёной". В приложении на сотни мегабайт цвет одной полосочки, как правило, мало на что влияет.

**Тестовый план** – базовый документ, описывающий перечень тестируемых компонент, критические факторы и риски тестирования, ресурсы и график тестирования, тестовую стратегию и виды тестирования.

**Хороший план должен обладать следующими свойствами:**

* Тестовый план должен быть полным, корректным и недвусмысленным;
* Он должен соответствовать принятому стандарту;
* Тестовый подход должен быть реально осуществимым;
* Цели тестирования должны быть определены;
* Тестовое окружение, конфигурация и оборудование должны быть детально описаны;
* Критерии остановки или прекращения тестирования должны быть определены;
* Критерии качества должны быть реалистичны;
* Все продукты, подлежащие доставке, должны быть определены;
* Стратегия и виды тестирования должны быть определены;
* Ресурсы как людские, так и необходимое оборудование выделены и описаны;
* График тестирования составлен, реалистичен и выполним.

**Test Case (тестовый сценарий)** – описывает сценарии приемочного, критического и расширенного тестов, ожидаемые результаты и результат выполнения каждого сценария с возможными комментариями.

**Характеристики хорошего теста:**

* Все требования проверены и охвачены тестами;
* Все test cases имеют уникальные идентификаторы и сгруппированы по логическим признакам;
* Test cases очевидны, корректны, ясны и сопоставимы с требованиями;
* Набор тестов не должен быть избыточным, слишком простым или слишком сложным;
* Тест должен быть лучшим в своей категории.

**Баг** - это невыполнение требования, связанного с предполагаемым или установленным использованием.

**Основные состояния дефекта (части его жизненного цикла)**

* Submitted (зафиксирован) – тестировщик записывает дефект в систему. Чтобы записать ошибку, должна быть выпущена версия программного продукта, на которую записывается баг.
* Assigned (назначен) – ведущий разработчик просматривает отчет об ошибке и назначает разработчика для устранения. Во время назначения дефекта, разработчику, ответственному за устранение ошибки, автоматически создается задача.
* Fixed (исправлен) – разработчик сделал необходимые изменения, чтобы устранить ошибку. В одном из следующих билдов (как правило, в следующем), можно будет проверить, действительно ли ошибка устранена.
* Verified (проверен) – тестировщик проверяет. устранена ли действительно ошибка (то есть, повторяется ли она снова согласно указанным шагам воспроизведения или нет), и если не повторяется – закрывает ее.
* Reopened (повторно обнаружен) – тестировщик проверяет, устранена ли действительно ошибка (то есть, повторяется ли она снова согласно указанным шагам воспроизведения или нет), и если повторяется – снова открывает ее. После чего, ошибка попадает на доработку к разработчику.
* Declined (отклонён) – ведущий разработчик, менеджер или координатор проекта могут отклонить ошибку по следующим причинам:
  + тестировщик записал дефект ошибочно, не разобравшись до конца, и записанный дефект не является реальной ошибкой;
  + баг дублирует записанную ранее, точно такую же, ошибку;
  + ошибка не воспроизводится по шагам, описанным в отчете об ошибке;

Могут быть и другие причины: например, заказчик решает не устранять проблему, даже если она действительно имеет место.

* Deferred (отложен) – баг откладывается на определенное время, что значит, что он не будет устраняться в данный момент.

Практическое задание к теме 2 - Тестирование документации и требований

Перед вами список требований к условному программному продукту "Текстовый редактор". Найдите в нём ошибки, укажите типы этих ошибок, сформулируйте предложения по исправлению ошибок.

R001. Интерфейс

R001-1. Приложение должно быть максимально похожим на notepad.

Проблема в недвусмысленности фразы notepad?

Ранжированность – среднее требование

Приложение должно быть похожим на Microsoft Windows notepad версии 1803.

R001-2. Цвета фона и текста должны задаваться в меню.

Проблема в незавершённости. В каком меню?

Ранжированность – среднее требование

Цвета фона и текста должны задаваться на панели управления во вкладке «Формат».

R001-2. Шрифты должны выбираться пользователями.

Требование R001-2 уже существует.

Проблема в незавершённости. Где выбираться?

Ранжированность – среднее требование

Шрифты должны выбираться пользователями на панели управления во вкладке «Формат».

R001-3. Кнопки должны утопать при наведении на них мышью.

Проблема в недвусмысленности. Утопать?

Ранжированность – мелкое требование

Кнопки должны изменять свой цвет на серый при наведении на них мышью.

R002. Функциональность

R002-1. Приложение должно уметь открывать, редактировать и сохранять файлы.

Проблема в незавершённости. Какие файлы?

Ранжированность – критическое требование

Приложение должно уметь открывать, редактировать и сохранять файлы с расширением .txt.

R002-2. Опционально должна быть возможность выбора кодировки символов.

Проблема в недвусмысленности.

Ранжированность – мелкое требование

В расширенном пакете должна быть возможность выбора кодировки символов.

R002-3. Приложение должно автоматически сохранять файл через заданный интервал времени, если файл не был сохранён.

Проблема в проверяемости. Заданный интервал?

Ранжированность – среднее требование

Приложение должно автоматически сохранять файл через каждую минуту, если файл не был сохранён.

R003. Общие требования

R003-1. Приложение должно быть написано на C#.

Проблема в недвусмысленности. C# какой версии?

Ранжированность – критическое требование

Приложение должно быть написано под .NET Core 3.1 на C# 8.0.

R003-2. Приложение должно быть удобным в использовании.

Проблема в проверяемости. Невозможно написать тест, проверяющий такое требование.

Ранжированность – среднее требование

Приложение должно содержать панель инструментов.

Тест к теме 2 - Тестирование документации и требований

1. Дайте определение требований.
2. Опишите три уровня требований − в чём они выражаются и что описывают?
3. Перечислите как можно больше свойств хороших требований.
4. Какими свойствами должны обладать наборы требований?
5. Какие проблемы чаще всего возникают при работе с требованиями? В чём их суть?
6. Что такое тестовый план?
7. Каковы основные свойства хорошего тестового плана?
8. Что такое тестовый сценарий (test case)?
9. Каковы характеристики хорошего тестового сценария?
10. Перечислите и опишите этапы жизненного цикла дефекта.
11. Что такое баг?