Анализ требований к программным системам

Проверка требований

Кулаков Кирилл Александрович

Тестирование требований

- Важность тестирования требований состоит в том, что хорошие требования позволяют:
  - Достичь общего понимания между заказчиком и разработчиком
  - Определить рамки проекта
  - Более точно определить финансовые и временные характеристики проекта.
  - Обезопасить заказчика от риска получить продукт, в котором он не сможет работать
  - Обезопасить разработчика от риска попасть в ситуацию «неконтролируемого размытия границ», которое может привести к непредвиденным затратам ресурсов сверх начальных ожиданий.

Характеристики хорошего требования

- Каждое требование должно быть:
  - **Завершённым** (complete). Все важные аспекты должны быть включены. Ничто не должно быть оставлено «для будущего определения» (TBD – to be defined).
  - **Непротиворечивым** (consistent). Требование не должно содержать противоречий как внутри себя, так и с другими требованиями.
  - **Корректным** (correct). 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** (unambiguous). Требование не должно допускать разночтений.
  - **Проверяемым** (verifiable). Требование должно быть сформулировано так, чтобы существовали способы однозначной проверки – выполнено требование или нет.

Характеристики хорошего набора требований

- Наборы требований должны быть:
  - **Модифицируемыми** (modifiable). Структура и стиль набора требований должны быть такими, чтобы набор требований можно было легко модифицировать. Должна отсутствовать избыточность. Должно быть построено корректное содержание всего документа.
  - **Прослеживаемыми** (traceable). У каждого требования должен быть уникальный идентификатор, по которому на это требование можно сослаться.
  - **Проранжированными по важности, стабильности и срочности** (ranked for importance, stability and priority). Для каждого требования должен быть указан уровень его важности (насколько оно важно для заказчика), стабильности (насколько высока вероятность, что это требование ещё будет изменено в процессе обсуждения деталей проекта) и срочности (как быстро требование должно быть реализовано).

Проблемы с требованиями

- Проблема незавершенности (неполноты)
- Проблема «To Be Defined»
- Проблема противоречивости
- Проблема некорректности
- Проблема двусмысленности
- Проблема непроверяемости
- Проблема немодифицируемости
- Проблема непрослеживаемости
- Проблема непроранжированности

Проблема незавершённости (неполноты)

- Хорошо, когда вся важная информация присутствует. Только вот вопрос – а как можно найти то, чего нет, но должно быть? Как догадаться, что что-то отсутствует?
- Следует обратить внимание на такие типичные случаи.
- Отсутствуют нефункциональные требования или нефункциональные составляющие требования. Мы знаем, что система должна делать. А про то, как (как быстро, как безопасно) в лучшем случае узнаём только в самом конце.
- Итак, мы уточнили какие-то требования, и заказчик высказал свои предпочтения. Например, ему нужно, чтобы приложение работало «надёжно и быстро».
- Думаем, как мы сможем проверить эти требования? Проверить, что приложение работает быстро. А как проверить? И что проверять?
- Вывод: кроме самого нефункционального требования, нам обязательно нужны критерии – как это требование проверить (и измерить). Чем важнее требование для заказчика, тем точнее должны быть эти критерии.

Проблема незавершённости (неполноты)

- Рекомендуется задавать общие вопросы.
- Их преимущества:
  - Они универсальные.
  - Они не навязывают решение.
  - Они не загоняют заказчика в ситуацию, когда ему приходится выбирать из имеющихся вариантов.

Проблема «To Be Defined»

- Ещё одной проблемой чрезмерной общности утверждений является т.н. TBD («to be defined», «будет определено»).

  Мы можем успокоиться (на время), только если знаем, кто и когда должен определить эти TBD. И почему они не определены сейчас.

- Также бывает, что стороны, ответственные за закрытие TBD, просто забывают об этом. Вывод? Нужно напоминать.

Проблемы противоречивости

- Противоречия внутри одного требования.
- Противоречия между двумя и более требованиями.
- Противоречия между таблицами и текстом.
- Противоречия между картинкой и текстом.
- Противоречия между требованием и прототипом.

Проблемы некорректности

- Ошибки могут быть вызваны:
  - опечатками, последствиями «copy-paste»;
  - остатками устаревших требований;
  - наличием «озолочения» («gold plating»);
  - наличием технически невыполнимых требований;
  - наличием неаргументированных требований к дизайну и архитектуре.

Проблемы двусмысленности

- Если что-то можно понять несколькими способами, можно быть уверенным, что разные люди поймут это по-разному.
- Например. В требовании сказано, что «функциональность X» является опциональной.
  - Что думает отдел разработки? «Чудесно. Опционально – значит необязательно. Можно не реализовывать.»
  - Что думает отдел маркетинга? «Ага. Мы выпустим две версии продукта. В более дорогой будет эта функциональность.»
  - Что думает заказчик? «Я за те же деньги получу ещё и вот эту функциональность».
- Вывод? Следует писать требования так, чтобы исключить возможность их двоякого понимания.

Проблемы непроверяемости

- Для начала следует отметить, что все вышеперечисленные проблемы ведут в том числе к тому, что мы не можем проверить, удовлетворяет ли продукт требованию.
- Как понять, что требование непроверяемо? Попробовать придумать несколько тестов для его проверки. Если тесты не придумываются – вот она, проблема.
- А бывает ли «просто непроверямое требование»? Да.
- Например: «В приложении должно быть ноль ошибок». «Приложение должно поддерживать все версии всех операционных систем».

Проблемы немодифицируемости

- После каждого обновления требований понадобится тратить недели чтобы выловить все появившиеся противоречия

Проблемы непрослеживаемости

- Если требования не пронумерованы, не имеют чёткого оглавления, не имеют работающих перекрёстными ссылок – это хаос. А в хаосе ошибки плодятся с удивительной скоростью.

Проблемы непроранжированности

- Если требования не проранжированы по важности, стабильности и срочности, мы рискуем уделить основное внимание не тому, что на самом деле важно для заказчика.

Проверка требований

- Если мы не можем проверить требование – это проблема.
- В конечном итоге именно тестировщики отвечают за то, чтобы требование было проверено.
- Если мы не можем проверить требование по объективным причинам, мы уточняем его до тех пор, пока оно не станет проверяемым. В зависимости от ситуации, мы расспрашиваем заказчика, разработчиков, своих более опытных коллег и т.д.

Проверка требований (техники)

- Одна из наиболее распространённых техник работы с требованиями – взаимный **перепросмотр**.
- Суть взаимного перепросмотра требований проста: после того, как один человек создал требование, другой человек это требование проверяет.
- Обычно, выделяют три уровня перепросмотра:
  - **Неформальный перепросмотр**. Двое коллег просто обмениваются листиками (файликами) и правят найденные ошибки, которые потом обсуждаются за чашкой чая или в любое другое относительно свободное время.
  - **Технический перепросмотр**. Это немного более формализованный процесс, требующий подготовки, выделенного времени, участия некоторой группы специалистов (желательно, из различных областей).
  - **Формальная инспекция**. Проводится редко и в случае очень больших проектов и крайней необходимости. Описывается специальными стандартами, требует соблюдения широкого спектра правил и протоколирования результатов.

Проверка требований (техники)

- Существует три простые техники работы с требованиями: задавать вопросы, писать тест кейсы и рисовать рисунки.
- Самый простой и не требующий большого опыта способ – **задавать как можно больше вопросов**. Получая разнообразные ответы от разных участников проекта (как наших коллег так и представителей заказчика), мы расширяем, углубляем и уточняем своё представление о том, что и как должно работать.
- Второй способ – **создавать тест-кейсы.** Когда вы видите требование, спросите себя: «Как я буду его тестировать? Какие тесты очевидно покажут, что это требование реализовано в ПС правильно?» Если с придумыванием таких тестов вы испытываете сложность – это тревожный звоночек: скорее всего, в требовании есть проблемы.
- Третий способ – **рисовать**. Чтобы увидеть общую картину требований целиком, очень удобно использовать рисунки, схемы, диаграммы и т.д.

Мнемоника CIRCUS MATTA

- CIRCUS MATTA — мнемоника для ревью пользовательских историй.
- Истории должны обладать качествами:
  - Completeness — полнота
  - Independent — независимость
  - Realisable — реализуемость
  - Consistency — консистентность
  - Unambiguity — однозначность

Мнемоника CIRCUS MATTA

- Specific — специфика заказчика
- Measurable — измеримая
- Acceptable — приемлемая
- Testable — тестируемая
- Traceable — трассируемая (можно проставить взаимосвязи)
- Achievable — достижимая