УПРАВЛЕНИЕ ТРЕБОВАНИЯМИ В ИТ-ПРОЕКТАХ

к. ф.-м. наук, Рабовская М.Я.

Agile Manifesto

Ценности

- **Люди и взаимодействие** важнее процессов и инструментов;
- **Работающий продукт** важнее исчерпывающей документации;
- **Сотрудничество с заказчиком** важнее согласования условий контракта;
- **Готовность к изменениям** важнее следования первоначальному плану.

Таким образом, не отрицая важности того, что справа, всё-таки больше ценится то, что слева.

Основные принципы

- Наивысшим приоритетом является удовлетворение потребностей заказчика, благодаря регулярной и ранней поставке ценного программного обеспечения;
- Изменение требований приветствуется, даже на поздних стадиях разработки;
- Работающий продукт следует выпускать как можно чаще, с периодичностью от пары недель до пары месяцев;
- На протяжении всего проекта разработчики и представители бизнеса должны ежедневно работать вместе; Над проектом должны работать мотивированные профессионалы.
- Чтобы работа была сделана, создайте условия, обеспечьте поддержку и полностью доверьтесь им;
- Непосредственное общение является наиболее практичным и эффективным способом обмена информацией как с самой командой, так и внутри команды;
- Работающий продукт — основной показатель прогресса;
- Инвесторы, разработчики и пользователи должны иметь возможность поддерживать постоянный ритм бесконечно;
- Постоянное внимание к техническому совершенству и качеству проектирования повышает гибкость проекта;
- Простота — искусство минимизации лишней работы — крайне необходима;
- Самые лучшие требования, архитектурные и технические решения рождаются у самоорганизующихся команд;
- Команда должна систематически анализировать возможные способы улучшения эффективности и соответственно корректировать стиль своей работы.

ГОСТ 34.602-89
ГОСТ 19.201-78

Техническое задание на создание автоматизированной системы по ГОСТ 34.602-89

Дата введения с 01.01.1990г.

Комплекс стандартов на автоматизированные системы 34.602-89

Техническое задание на создание программного обеспечения по ГОСТ 19.201 -78

Виды и свойства требований

- *Функциональные* требования являются детальным описанием поведения и сервисов системы, ее функционала. Они определяют то, что система должна уметь делать.
- *Нефункциональные* требования не являются описанием функций системы. Этот вид требований описывает такие характеристики системы, как надежность, особенности поставки (наличие инсталлятора, документации), определенный уровень качества (например, для новой Java-машины это будет означать, что она удовлетворяет набору тестов, поддерживаемому компанией Sun). Сюда же могут относиться требования на средства и процесс разработки системы, требования к переносимости, соответствию стандартам и т.д. Требования этого вида часто относятся ко всей системе в целом. На практике, особенно начинающие специалисты, часто забывают про некоторые важные нефункциональные требования.

Сформулируем ряд важных свойств требований.

- *Ясность, недвусмысленность* — однозначность понимания требований заказчиком и разработчиками. Часто этого трудно достичь, поскольку конечная формализация требований, выполненная с точки зрения потребностей дальнейшей разработки, трудна для восприятия заказчиком или специалистом предметной области, которые должны проинспектировать правильность формализации.
- *Полнота и непротиворечивость*.
- *Необходимый уровень детализации*. Требования должны обладать ясно осознаваемым уровнем детализации, стилем описания, способом формализации: либо это описание свойств предметной области, для которой предназначается ПО, либо это техническое задание, которое прилагается к контракту, либо это проектная спецификация, которая должна быть уточнена в дальнейшем, при детальном проектировании. Либо это еще что-нибудь. Важно также ясно видеть и понимать тех, для кого данное описание требований предназначено, иначе не избежать недопонимания и последующих за этим трудностей. Ведь в разработке ПО задействовано много различных специалистов – инженеров, программистов, тестировщиков, представителей заказчика, возможно, будущих пользователей – и все они имеют разное образование, профессиональные навыки и специализацию, часто говорят на разных языках. Здесь также важно, чтобы требования были максимально абстрактны и независимы от реализации.
- *Прослеживаемость* — важно видеть то или иное требование в различных моделях, документах, наконец, в коде системы. А то часто возникают вопросы типа – "Кто знает, почему мы решили, что такой-то модуль должен работать следующим образом ….?". Прослеживаемость функциональных требований достигается путем их дробления на отдельные, элементарные требования, присвоение им идентификаторов и создание трассировочной модели, которая в идеале должна протягиваться до программного кода. Хочется например, знать, где нужно изменить код, если данное требование изменилось. На практике полная формальная прослеживаемость труднодостижима, поскольку логика и структура реализации системы могут сильно не совпадать с таковыми для модели требований. В итоге одно требование оказывается сильно "размазано" по коду, а тот или иной участок кода может влиять на много требований. Но стремиться к прослеживаемости необходимо, разумно совмещая формальные и неформальные подходы.
- *Тестируемость и проверяемость* — необходимо, чтобы существовали способы оттестировать и проверить данное требование. Причем, важны оба аспекта, поскольку часто проверить-то заказчик может, а вот тестировать данное требование очень трудно или невозможно в виду ограниченности доступа (например, по соображениям безопасности) к окружению системы для команды разработчика. Итак, необходимы процедуры проверки – выполнение тестов, проведение инспекций, проведение формальной верификации части требований и пр. Нужно также определять "планку" качества (чем выше качество, тем оно дороже стоит!), а также критерии полноты проверок, чтобы выполняющие их и руководители проекта четко осознавали, что именно проверено, а что еще нет.
- *Модифицируемость*. Определяет процедуры внесения изменений в требования.

Варианты формализации требований

- *Неформальная постановка требований в переписке по электронной почте*. Хорошо работает в небольших проектах, при вовлеченности заказчика в разработку (например, команда выполняет субподряд). Хорошо также при таком стиле, когда есть взаимопонимание между заказчиком и командой, то есть лишние формальности не требуются. Однако, электронные письма в такой ситуации часто оказываются важными документами – важно уметь вести деловую переписку, подводить итоги, хранить важные письма и пользоваться ими при разногласиях. Важно также вовремя понять, когда такой способ перестает работать и необходимы более формальные подходы.
- *Требования в виде документа* – описание предметной области и ее свойств, техническое задание как приложение к контракту, функциональная спецификация для разработчиков и т.д.
- *Требования в виде графа с зависимостями* в одном из средств поддержки требований (IBM Rational RequisitePro, DOORS, Borland CaliberRM и нек. др.). Такое представление удобно при частом изменении требований, при отслеживании выполнения требований, при организации "привязки" к требованиям задач, людей, тестов, кода. Важно также, чтобы была возможность легко создавать такие графы из текстовых документов, и наоборот, создавать презентационные документы по таким графам.
- *Формальная модель требований* для верификации, модельно-ориентированного тестирования и т.д.
- Итак, каждый способ представления требований должен отвечать на следующие вопросы: кто потребитель, пользователь этого представления, как именно, с какой целью это представление используется.

Некоторые ошибки при документировании требований 1

- Описание возможных решений вместо требований.
- Нечеткие требования, которые не допускают однозначную проверку, оставляют недосказанности, имеют оттенок советов, обсуждений, рекомендаций: "Возможно, что имеет смысл реализовать также…..", "и т.д.".
- Игнорирование аудитории, для которой предназначено представление требований. Например, если спецификацию составляет инженер заказчика, то часто встречается переизбыток информации об оборудовании, с которым должна работать программная система, отсутствует глоссарий терминов и определений основных понятий, используются многочисленные синонимы и т. д. Или допущен слишком большой уклон в сторону программирования, что делает данную спецификацию непонятной всем непрограммистам.
- Пропуск важных аспектов, связанных с нефункциональными требованиями, в частности, информации об окружении системы, о сроках готовности других систем, с которыми должна взаимодействовать данная. Последнее случается, например, когда данная программная система является частью более крупного проекта. Типичны проблемы при создании программно-аппаратных систем, когда аппаратура не успевает вовремя и ПО невозможно тестировать, а в сроках и требованиях это не предусмотрено….

Некоторые ошибки при документировании требований 2

- Теперь о неточностях: нельзя смешивать понятия модулей и подсистем, поскольку подсистема — это та же система, только маленькая. Подсистема, как и система, включает в себя все виды обеспечения, все те же общие требования, а модуль есть всего лишь «Программа или функционально завершенный фрагмент программы, предназначенный для хранения, трансляции, объединения с другими программными модулями и загрузки в оперативную память [из п. 15 Таблицы 1 ГОСТ 19781-90]»

Цикл работы с требованиями

- *Выделение требований (requirements elicitation)*, нацеленное на выявление всех возможных источников требований и ограничений на работу системы и извлечение требований из этих источников.
- *Анализ требований (requirements analysis)*, целью которого является обнаружение и устранение противоречий и неоднозначностей в требованиях, их уточнение и систематизация.
- *Описание требований (requirements specification)*. В результате этой деятельности требования должны быть оформлены в виде структурированного набора документов и моделей, который может систематически анализироваться, оцениваться с разных позиций и в итоге должен быть утвержден как официальная формулировка требований к системе.
- *Валидация требований (requirements validation)*, которая решает задачу оценки понятности сформулированных требований и их характеристик, необходимых, чтобы разрабатывать ПО на их основе, в первую очередь, непротиворечивости и полноты, а также соответствия корпоративным стандартам на техническую документацию.

Некоторые принципы разработки ТЗ

- Грамотный исполнитель в этих условиях может выбрать т.н. спиральную модель разработки ПО, в рамках которой ТЗ, фактически, разрабатывается на каждом новом витке спирали и описывает те изменения, которые должны произойти в следующей версии программного продукта.
- «Законодательно» ТЗ разрабатывается самим заказчиком только в том случае, если он представляет Министерство обороны или иное силовое ведомство;
- Тендерную документацию разрабатывает именно тот подрядчик, который заведомо должен выиграть конкурс. Простите, но это жизненные реалии...

Шаблонное построение фраз

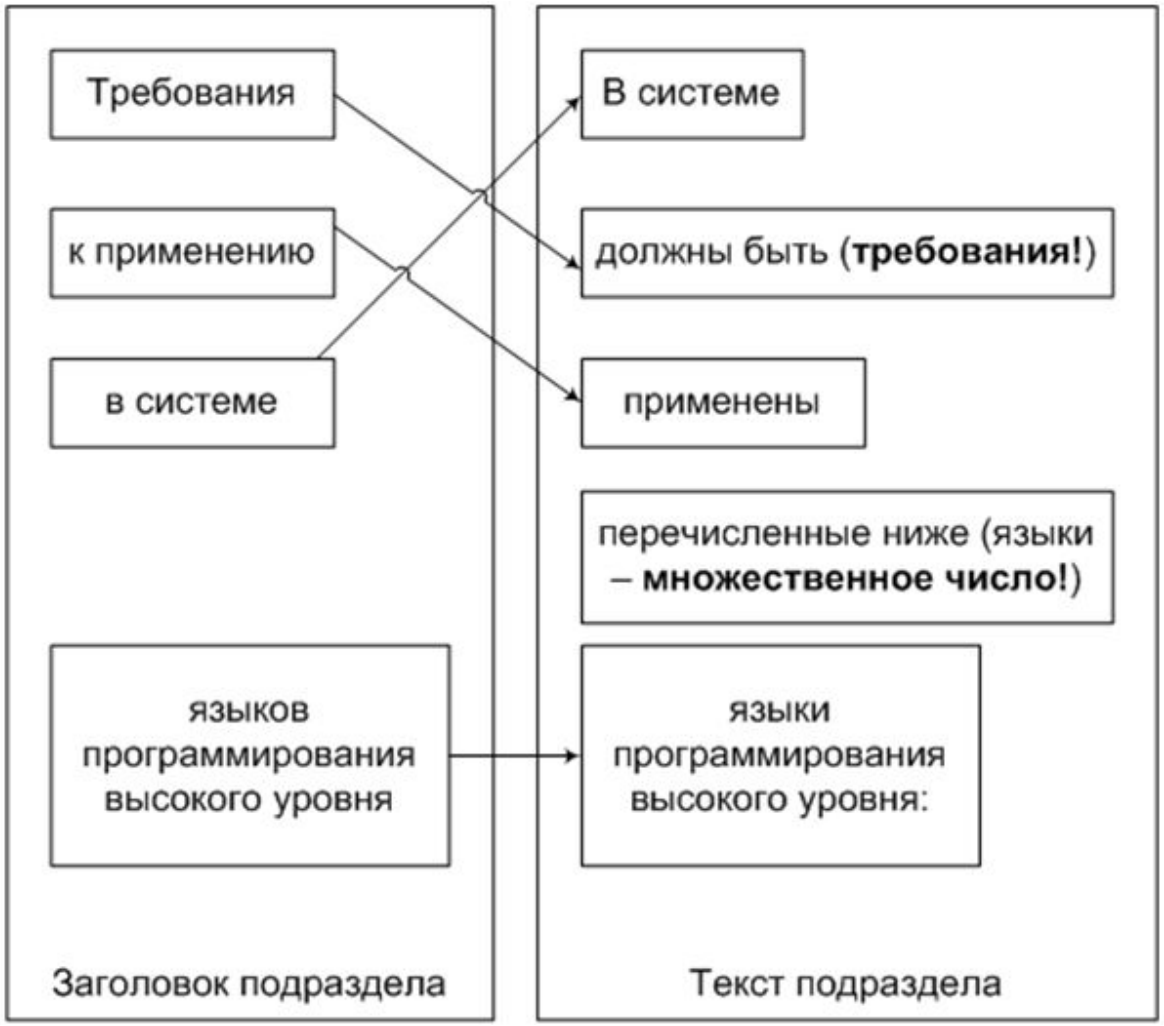

Схема на практике

- Еще один пример - Требования к численности и квалификации персонала системы и режиму его работы. Настоятельно рекомендуется не забывать о детализации, за детализацию разделов технического задания никто никого еще не наказывал. Итак, пишем пункт технического задания:
- 2.3.4. Требования к численности и квалификации персонала системы и режиму его работы
- (детализируем - создаем подпункты)
- 2.3.4.1. Требования к численности персонала
- (правильно формулируем текст подпункта - отвечаем на вопрос, каким требованиям должна удовлетворять численность персонала)
- Численность персонала (требования-то предъявляются к численности!) **должна удовлетворять требованиям**:
- быть достаточной для реализации автоматизированных функций системы во всех режимах работы системы;
- обеспечивать полную занятость персонала при реализации автоматизированных функций системы и т.д.
- 2.3.4.2. Требования к квалификации персонала
- Квалификация персонала (требования предъявляются именно к квалификации!) **должна обеспечивать** эффективное функционирование технических и программных средств системы во всех режимах работы системы.

Уточнение требований

- 2.3.4.1. Требования к численности персонала
- Численность персонала должна удовлетворять требованиям:
- быть достаточной для реализации автоматизированных функций системы во всех режимах работы системы;
- обеспечивать полную занятость персонала при реализации автоматизированных функций системы и т.д.
- Лапша полнейшая, но формально все верно. Рекомендуется к применению, если невозможно конкретизировать какой-либо пункт технического задания. Если Большой Босс будет неудовлетворен, можно вежливо попросить его уточнить требования Заказчика по данному пункту, дать более точную информацию. Это право техписа, не контактирующего непосредственно с Заказчиком.

Объекты в ТЗ

- Наименование изделия - преобразователь энергии солнечного излучения в энергию человеческого разума (**далее по тексту - Изделие**).
- И, в тексте - Изделие, Изделие, Изделие...
- Тоже самое относится к программным изделиям и автоматизированным системам. Наименование АС - автоматизированная система раздачи грубых кормов для крупного рогатого скота (**далее по тексту - Система**).
- И, в тексте - Система, Система, Система... Программа, Программа, Программа...
- Итог - догнали и завалили сразу двух зайцев. Склонять-спрягать целую кучу слов не потребуется, да и читать построенное таким образом техническое задание будет проще.
- Ниже - типовые перечни штампов, долго и успешно применяемых при разработке технических заданий (по основным разделам, выделено жирным):
- назначение системы - система **предназначена для** решения перечисленных ниже задач:
  - **задачи** такой-то (первой);
  - **задачи** сякой-то (второй);
  - и так далее.
- цели создания системы - **целями создания системы являются**:
  - **увеличение** скорости...;
  - **повышение** точности...;
  - **уменьшение** издержек...;
  - **снижение** потребления...;
  - **улучшение** показателей...;
  - и так далее

Шаблоны формулировок

- требования к функциям (задачам), выполняемым системой - система **должна обеспечивать возможность выполнения** перечисленных ниже **функций**:
  - в **рамках** первой **задачи** - выполнение функции такой-то, такой-то и еще какой-то;
  - в **рамках** второй **задачи** - выполнение функции такой-то и пр.
- Если функция автоматизированная, тогда именно **обеспечивать возможность выполнения** указанной функции. Пользователь может убрать стопор - мельница начнет молоть муку. Но пользователь может стопор и не убрать. В **указанном** случае мельница (система) будет находиться в режиме ожидания.
- Если функция автоматическая, тогда система должна именно **обеспечивать выполнение** функции. Функция автоматического резервирования базы данных запускается программными средствами системы (без участия персонала) по заданному расписанию и сливает базу данных на резервный носитель.
- По части рамок задач. Задачи **решаются**, а функции **выполняются**. Чтобы **решить** задачу, надо **выполнить** ряд функций, процедур или операций. Иными словами - задача есть более крупный структурный элемент. Пример.
- В **рамках задачи** (или **для решения задачи**) ведения базы данных программные средства системы **должны обеспечивать выполнение** перечисленных ниже **функций**:
- **автоматизированной** функции добавления записей в таблицы базы данных;
- автоматизированной функции удаления записей из таблиц базы данных;
- автоматизированной функции сортировки записей в таблицах базы данных;
- ...;
- **функции автоматического** резервирования базы данных.
- И из предыдущего подраздела:
- **должны быть**...;
- **должна удовлетворять требованиям**..
- В результате применения штампов тескт технического задания становится унифицированным и формализованным. **Никаких прикрас**. И Заказчик-**мужчина никуда не уйдет** от вас, милые девушки-техписы, поскольку требования технического задания будут для него **прозрачны**.

Перечни и нумерация разделов

- Случай первый.
- В **рамках задачи** (или **для решения задачи**) ведения базы данных программные средства системы **должны обеспечивать выполнение** перечисленных ниже **функций**:
- **автоматизированной функции** добавления записей в таблицы базы данных;
- автоматизированной функции удаления записей из таблиц базы данных;
- автоматизированной функции сортировки записей в таблицах базы данных...;
- Случай второй.
- 4.3.2.1. В **рамках задачи** (или **для решения задачи**) ведения базы данных программные средства системы **должны обеспечивать выполнение** перечисленных ниже **функций**:
- **автоматизированной функции** добавления записей в таблицы базы данных;
- автоматизированной функции удаления записей из таблиц базы данных;
- автоматизированной функции сортировки записей в таблицах базы данных...;
- Отличия, казалось бы, невелики. Но!
- В первом случае, в документе «Программа и методики испытаний», придется написать «методика проверки выполнения системой автоматизированной функции добавления записей в таблицы базы данных».
- Во втором случае, всего-навсего - «методика проверки выполнения п. 4.3.2.1(1) технического задания».
- В Протоколе испытаний, в первом случае - «требования технического задания к выполнению автоматизированной функции добавления записей в таблицы базы данных выполнены».
- Во втором случае - «требования п. 4.3.2.1(1) технического задания выполнены».

Пример. 2.2. Назначение системы

- Система должна обеспечивать выполнение (возможность выполнения) перечисленных ниже задач:
- задачи сбора данных с каких-то, допустим, датчиков;
- задачи обработки, хранения, отображения и пр. информации в центре сбора.
- Вот и все. Немного фантазии, и раздел готов:
- Система должна обладать иерархической структурой и включать в себя перечисленные ниже уровни иерархии:
- 1-й уровень - **уровень сбора данных**;
- 2-й уровень - **уровень консолидации данных** (централизованная обработка, хранение и пр.)