# Требования

# Стандартное определение понятия «требование»

- Стандарт IEEE
  - ***Требования к программной системе* – это:**
  - Функциональность, необходимая пользователю для решения проблемы или достижения цели.
  - Функциональность, которая должна быть получена (достигнута) системой или ее компонентами для соответствия контракту, стандарту, спецификации или другим формальным документам.
  - Документальное представление пп. 1 – 2

- Стандарт ISO 12207
  - ***Разработчик должен установить и документально оформить следующие требования к программным средствам:***
  - функциональные и технические требования, включая производительность, физические характеристики и окружающие условия, под которые должен быть, создан программный объект;
  - требования к внешним интерфейсам программного объекта;
  - квалификационные требования;
  - требования безопасности, включая требования, относящиеся к методам эксплуатации, сопровождения, воздействию окружающей среды и травмобезопасности персонала;
  - и т.д.

- Требования
  - Цели разработки
  - Участники разработки
  - Критерии качества требований
  - Классификация требований
  - Приоритеты требований
  - Разработка требований

# Цели разработки требований

- Обеспечение наиболее полного и точного отражения условий или возможностей, необходимых заказчику для решения его проблем и достижения бизнес-целей
- Снижение затрат на разработку, обслуживание и поддержку сложного заказного программного обеспечения

# Участники разработки требований

**Основные задачи разработки требований**:

- создание базовой версии требований к продукту в оговоренные сроки
- достижение управляемости процессом разработки требований и формирование атмосферы взаимного доверия среди заинтересованных лиц:
  - заказчик должен быть уверен, что разработчики будут взаимодействовать с ним для создания нужной системы, даже если перед началом работы над проектом он не успел продумать все требования;
  - заказчик должен быть уверен, что границы проекта не выйдут из-под контроля, поскольку решения относительно границ принимает он;
  - менеджер проекта должен быть уверен, что команда разработчиков получила достойного делового партнера, который совместно с разработчиками готов отвечать за качество продукта;
  - аналитик требований должен быть уверен в том, что сможет управлять изменениями проекта, минимизируя хаос

# Участники разработки требований

- **Заказчики**, которые финансируют проект или приобретают продукт для решения каких-либо бизнес-задач
- **Пользователи**, которые взаимодействуют напрямую или косвенно с приложением (подкласс заказчиков)
- **Аналитики требований**, которые пишут требования и передают их разработчикам
- **Разработчики**, которые создают, разворачивают и поддерживают продукт.
- **Тестировщики**, которые определяют соответствие поведения программного обеспечения желаемому
- **Технические писатели**, которые отвечают за создание руководства пользователей, тренировочных материалов и справочной системы
- **Менеджер проекта**, который планирует процесс и руководит командой разработчиков вплоть до успешного выпуска продукта
- **Сотрудники правового отдела**, которые следят, чтобы продукт не противоречил всем действующим законам и постановлениям
- **Сотрудники отдела продаж и маркетинга, выездной службы поддержки и другие**, кому придется работать с продуктом и его

# Права и обязанности заказчика

- Права
  - Иметь дело с аналитиком, который разговаривает на языке его предметной области.
  - Иметь дело с аналитиком, хорошо изучившим его бизнес и цели, для которых создается система.
  - Потребовать, чтобы аналитик преобразовал требования, предоставленные им устно, в письменную спецификацию требований к программному продукту.
  - Получить от аналитика подробный отчет обо всех рабочих продуктах, созданных в процессе формулирования требований.
  - На уважительное и профессиональное отношение со стороны аналитиков и разработчиков.
  - Знать о вариантах и альтернативах требований и их реализации.
  - Описать характеристики, упрощающие работу с продуктом.
  - Изменить требования или разрешить использование имеющихся программных компонентов.
  - Получить исчерпывающие сведения о сумме затрат, ожидаемом эффекте и необходимых компромиссах, которые возникают в связи с изменениями в ПО.
  - Потребовать, чтобы система функциональностью и качеством удовлетворяла его требованиям.

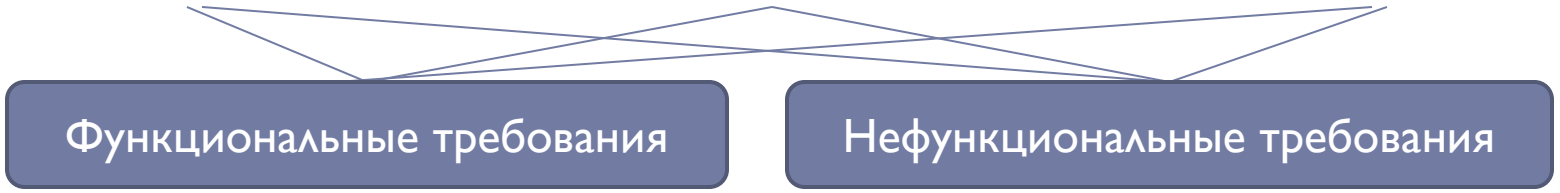
- Обязанности
  - Ознакомить аналитиков и разработчиков с особенностями своего бизнеса.
  - Потратить столько времени, сколько необходимо, на объяснение требований.
  - Точно и конкретно описать требования к системе.
  - Принимать своевременные решения.
  - Уважать определенную разработчиком оценку стоимости и возможность реализации его требований.
  - Определять приоритеты требований.
  - Просматривать документы с требованиями и оценивать прототипы.
  - Своевременно сообщать об изменениях требований.
  - Поддерживать принятый в организации-разработчике порядок контроля изменений.
  - Уважительно относиться к методам, с помощью которых аналитики создают требования.

# Обязанности аналитика требований

- Определить бизнес-требования к проекту
- Определить классы пользователей продукта и выбрать соответствующих представителей каждого класса – сторонников продукта, а также заручиться их поддержкой и согласовать обязанности
- Выявить требования к проекту, используя для этого способы сбора информации, рекомендуемые настоящим Положением
- Вести анализ требований для обеспечения их надлежащего качества.
- Обеспечить создание спецификации требований к продукту
- Определять необходимость представления требований с помощью моделей графического анализа, таблиц, математических уравнений, раскадровок и прототипов
- Управлять проверкой требований
- Обеспечить расстановку приоритетов требований
- Обеспечить выпуск базовой версии требований
- Управлять требованиями после выпуска базовой версии требований

# Классификация требований

- Уровни требований
  - Бизнес требования
  - Требования пользователей
  - Системные требования



# Уровни требований

- *Бизнес-требования* содержат высокоуровневые задачи и цели организации-разработчика или заказчиков системы. Как правило, их высказывают те, кто финансируют проект, покупатели системы, менеджер реальных пользователей или отдел маркетинга. Эти требования объясняют, почему организации нужна такая система, или, иначе говоря, описывают цели, которые организация намерена достичь с ее помощью. Бизнес-требования записываются в соответствующем разделе Спецификации требований к программному обеспечению, хотя могут записываться и в форме документа об образе и границах проекта, который еще иногда называют уставом проекта или документом рыночных требований

# Требования пользователей

- *Требования пользователей* описывают цели и задачи, которые пользователям позволит решить система. К отличным способам представления этого вида требований относятся варианты использования, сценарии и таблицы «событие – отклик». Таким образом, требования пользователей определяют, что клиенты смогут делать с помощью системы.

# Системные требования

- *Системные требования* определяют функциональность и характеристики системы, которую должны построить разработчики, для того чтобы пользователи смогли выполнить свои задачи (в рамках бизнес-требований)
- Термином системные требования обозначают высокоуровневые требования к продуктам, которые содержат многие подсистемы, то есть системам (IEEE, 1998c). Говоря о системе, мы подразумеваем программное обеспечение или подсистемы программного обеспечения и оборудования. Люди – часть системы, поэтому определенные функции системы могут распространяться и на людей

# Функциональные требования

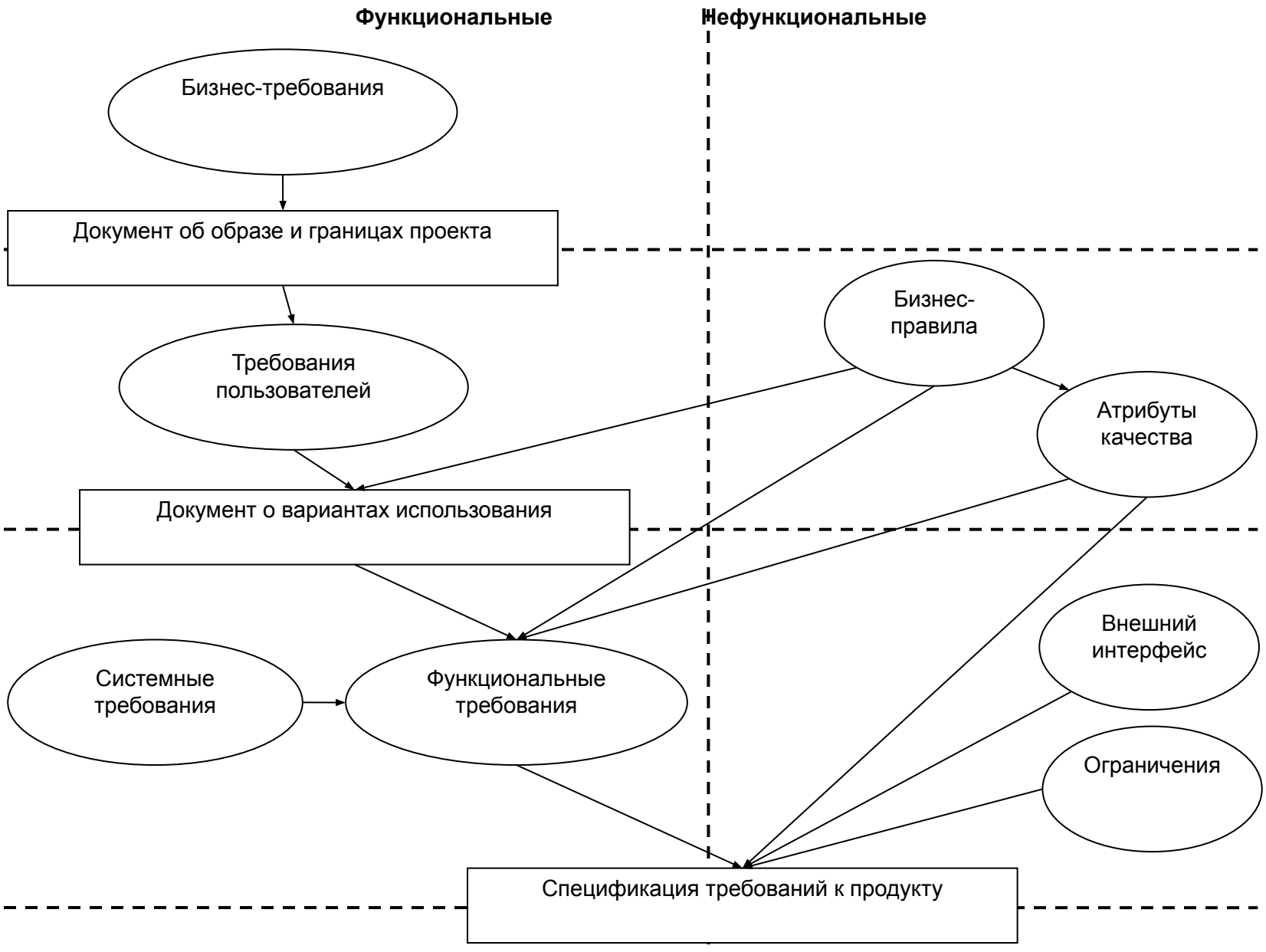
- *Функциональные требования* определяют функции, которые выполняет система, и зависят от потребностей пользователей и типа решаемой задачи. Функциональные пользовательские требования описывают функции в обобщенном виде. Выполняя детализацию этих требований, разработчики формируют более подробное и точное описание сервисов системы – *функциональные системные требования*.
  - Особое внимание при документировании требований нужно уделить их точному описанию. Неточности в описании будут интерпретироваться пользователями и разработчиками по-разному. Такое положение приведет к разработке новых требований или изменению существующих требований, а, значит, к внесению изменений в систему и ее удорожанию.
  - *Спецификация требований*, содержащая пользовательские и системные требования должна быть комплексной и непротиворечивой. В ней должны быть определены все функции системы, и не должно быть несовместимых и взаимоисключающих определений функций.

# Нефункциональные требования

*Нефункциональные требования* определяют характеристики и ограничения системы и не связаны непосредственно с функциональными требованиями. Они формируются на основе имеющихся атрибутов качества, требований к внешнему интерфейсу и ограничений.

Нефункциональные требования делится на:

- нефункциональные *требования к продукту*
- нефункциональные *требования к процессу*
- *внешние нефункциональные требования*




Взаимосвязи разных типов информации для требований

# Бизнес-правила

- *Бизнес-правила* включают корпоративные политики, правительственные постановления, промышленные стандарты и вычислительные алгоритмы. Бизнес-правила не являются требованиями к программному обеспечению, потому что они находятся снаружи границ любой системы программного обеспечения. Однако они часто налагают ограничения, определяя, кто может выполнять конкретные варианты использования, или диктовать, какими функциями должна обладать система, подчиняющаяся соответствующим правилам. Иногда бизнес-правила становятся источником атрибутов качества, которые реализуются в функциональности
- Бизнес-правила – один из основных источников функциональных требований к программному обеспечению, поскольку они определяют возможности, которыми должна обладать система для выполнения правил

# Классификация бизнес-правил

- **Факты** – это всего лишь верные утверждения о бизнесе, зачастую описывающие связи и отношения между важными бизнес-терминами. Факты также называют инвариантами – неизменными истинами о сущности данных и их атрибутах. Примеры фактов: на каждый товар нанесен уникальный штрих-код; оплачивается доставка каждого заказа; каждый элемент заказа содержит данные о товаре и его цене.
- **Ограничения** – определяют, какие операции может выполнять система и ее пользователи. Вот некоторые слова и фразы, которые часто применяются при описании ограничивающего бизнес-правила: *должен*, *не должен*, *не может* и *только*. Скорее всего, в организации есть политики безопасности, определяющие порядок доступа к информационным системам. Обычно они констатируют, какие пароли следует применять, как часто их надо менять, можно ли применять старые пароли и т.п. Все эти ограничения, касающиеся доступа к приложению, можно считать бизнес-правилами.
- **Активаторы операций.** Правило, при определенных условиях приводящее к выполнению каких-либо действий, называется активатором операции. Человек может выполнять эти действия самостоятельно. Как вариант, правило может управлять некоторыми программными функциями, благодаря которым приложение при выполнении определенных условий реализует нужную модель поведения. Условия, определяющие выполнение операции, иногда представляют собой сложную комбинацию значений «истина» и «ложь», выполняющихся для нескольких отдельных условий. Выражение вида «*Если <некоторое условие верно или наступило определенное событие>, то <что-то произойдет>*», – это признак, который описывает активатор операции.

# Классификация бизнес-правил

- **Выводы.** Вывод, иногда его еще называют предположительным знанием, – это правило, устанавливающее новые реалии на основе достоверности определенных условий. Вывод создает новый факт на основе других фактов или вычислений. Выводы зачастую записывают в формате «*если – то*», применяемом также при записи бизнес-правил, активирующих операции; тем не менее, раздел «*то*» вывода заключает в себе факт или предположение, а не действие. Вот несколько примеров выводов: если платеж не поступил в течение 30 календарных дней с момента отправки счета, счет считается просроченным; если поставщик не может поставить заказанный товар в течение пяти дней с момента получения заказа, заказ считается невыполненным.
- **Вычисления.** Компьютеры осуществляют вычисления, и поэтому один из классов бизнес-правил определяет вычисления, выполняемые с использованием математических формул и алгоритмов. Многие вычисления выполняются по внешним для предприятия правилам, например по формулам удержания подоходного налога. В отличие от активирующих операции бизнес-правил, для реализации которых иногда приходится создавать специфические функциональные требования к программному обеспечению, правила вычислений в той форме, в которой они выражены, можно непосредственно рассматривать в качестве требований к программному обеспечению. Бизнес-правила для вычислений можно представлять в текстовой форме, в символьной форме, например в виде математических выражений, однако представление таких правил в виде таблицы гораздо понятнее, чем длинный список сложных текстовых выражений.

# Нефункциональные требования к продукту

- *Нефункциональные требования к продукту* определяют его эксплуатационные качества, т.е. определяют, то насколько хорошо будет работать система. Часто такие характеристики называются ***атрибутами или факторами качества программ***. Основная сложность заключается в том, что атрибуты качества трудно определить (выявить), их невозможно измерить, и они сильно влияют на реализацию системы.
- Атрибуты качества представляют собой дополнительное описание функций продукта, выраженное через описание его характеристик, важных для пользователей или разработчиков. К таким характеристикам относятся легкость и простота использования, легкость перемещения, целостность, эффективность и устойчивость к сбоям.
- Существует большое число атрибутов качества.
- Например, стандарт ISO 9126 предлагает оценивать программную продукцию по шести *характеристикам качества*, рекомендуя использовать 21 *показатель* (*подхарактеристику*) *качества*. Этот же стандарт советует учитывать, что представления о качестве для разных групп заинтересованных лиц отличаются, приводя в качестве примера представления о качестве пользователей, разработчиков и руководителей проекта.

# Классификация атрибутов качества

- *Способность к взаимодействию*
- *Надежность*
- *Устойчивость к сбоям*
- *Удобство и простота использования*

- Атрибуты, важные для разработчиков и специалистов по техническому обслуживанию
  - *Легкость в эксплуатации*

# Атрибуты, важные для пользователей

- Под *доступностью* понимается запланированное время доступности (up time), в течение которого система действительно доступна для использования и полностью работоспособна. Формально доступность равна среднему времени до сбоя (mean time to failure, MTTF) системы, деленному на сумму среднего времени до сбоя и ожидаемого времени до восстановления системы после сбоя.
- *Эффективностью* называется показатель того, насколько эффективно система использует производительность процессора, место на диске, память или полосу пропускания соединения.
- *Гибкость*. Этот атрибут также называют расширяемостью, дополняемостью, наращиваемостью или растяжимостью. Гибкость показывает, с какой легкостью в продукт удается добавить новые возможности.
- *Целостность*, которая включает в себя и безопасность, связана с блокировкой неавторизированного доступа к системным функциям, предотвращением потери информации, антивирусной защитой программного обеспечения и защитой конфиденциальности и безопасности данных, введенных в систему.
- *Способность к взаимодействию* показывает, каким образом система обменивается данными или сервисами с другими системами. Чтобы оценить способность к взаимодействию, вам необходимо знать, какие приложения клиенты будут применять совместно с вашим продуктом и обмен какими данными предполагается.

# Атрибуты, важные для пользователей

- *Надежностью* называется вероятность работы программного обеспечения без сбоев в течение определенного периода времени. Иногда одной из характеристик надежности считают устойчивость к сбоям. Для измерения надежности программного обеспечения используют такие показатели, как процент успешно завершенных операций и средний период времени работы системы до сбоя.
- Под *устойчивостью к сбоям*, которую еще иногда называют отказоустойчивостью (fault tolerance), понимают уровень, до которого система продолжает корректно выполнять свои функции, несмотря на неверный ввод данных, недостатки подключенных программных компонентов или компонентов оборудования или неожиданные условия работы.
- *Удобство и простота использования.* Этот атрибут связан с массой факторов, которые составляют основу того, что пользователи часто описывают как дружелюбие к пользователю. Удобство и простота использования измеряется усилиями, требуемыми для подготовки ввода данных, эксплуатации и вывода конечной информации.

# Атрибуты, важные для разработчиков и специалистов по техническому обслуживанию

- ***Легкость в эксплуатации.*** Этот атрибут показывает, насколько удобно исправлять ошибки или модифицировать программное обеспечение. Легкость в эксплуатации зависит от того, насколько просто разобраться в работе программного обеспечения, изменять его и тестировать, и тесно связано с гибкостью и тестируемостью. Этот показатель крайне важен для продуктов, которые подвергаются частым изменениям, и тех, что создаются быстро (и, возможно, с экономией на качестве).
- ***Мобильность.*** Мерой ее измерения можно считать усилия, необходимые для перемещения программного обеспечения из одной операционной среды в другую. Зачастую к мобильности относят и возможность интернационализации и локализации продукта.
- ***Возможность повторного использования.*** Постоянная задача разработки программного обеспечения – возможность повторного использования – показывает усилия, необходимые для преобразования программных компонентов с целью их дальнейшего применения в других приложениях. Затраты на разработку программного обеспечения с возможностью повторного использования значительно выше, чем на создание компонента, который будет работать только в одном приложении. Оно должно быть модульным, хорошо задокументированным, не зависеть от конкретных приложения и операционной среды, а также обладать некоторыми универсальными возможностями.
- ***Тестируемость.*** Этот атрибут также называют проверяемостью или верифицируемостью, он показывает легкость, с которой программные компоненты или интегрированный продукт можно проверить на предмет дефектов.

# Взаимосвязь атрибутов качества

- Большинство атрибутов качества противоречат друг другу и для определенных их комбинаций компромиссы неизбежны. Пользователи и разработчики должны решить, какие атрибуты важнее других и при принятии решений соблюдать эти приоритеты.
- В таблице, приведенной ниже, отражены взаимосвязи наиболее распространенных атрибутов качества. Знак «+» в ячейке означает, что увеличение величины атрибута в соответствующей строке позитивно влияет на атрибут в соответствующем столбце. Знак «-» в ячейке означает, что увеличение величины атрибута в этой строке негативно влияет на атрибут в соответствующем столбце. Пустая ячейка означает, что атрибут в этой строке оказывает незначительное влияние на атрибут в столбце.

# Позитивные и негативные взаимосвязи атрибутов качества

| | Доступность | Эффективность | Гибкость | Целостность | Способность к взаимодействию | Легкость в эксплуатации | Мобильность | Надежность | Возможность повторного использования | Устойчивость к сбоям | Тестируемость | Удобство и простота использования |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступность | | | | | | | | + | | + | | |
| Эффективность | | | - | | - | - | - | - | | - | - | - |
| Гибкость | | - | | - | | + | + | + | | | + | |
| Целостность | | - | | | - | | | | - | | - | - |
| Способность к взаимодействию | | - | + | - | | | + | | | | | |
| Легкость в эксплуатации | + | - | + | | | | | + | | | + | |
| Мобильность | | - | + | | + | - | | | + | | + | - |
| Надежность | + | - | + | | | + | | | | + | + | + |
| Возможность повторного использования | | - | + | - | + | + | + | - | | | + | |
| Устойчивость к сбоям | + | - | | | | | | + | | | | + |
| Тестируемость | + | - | + | | | + | | + | | | | + |
| Удобство и простота использования | | - | | | | | | | | + | - | |

# Нефункциональные требования к процессу

- *Нефункциональные требования к процессу* зависят от политики и организационных процедур заказчика и разработчика. Они определяют ограничения, связанные с использованием определенных технологий и стандартов разработки, ограничения реализации, например, использования определенного языка программирования и методов проектирования, требования к документации, срокам изготовления программного продукта и т.д.

# Внешние нефункциональные требования

- *Внешние нефункциональные требования* учитывают факторы, внешние по отношению к системе и процессу ее разработки. Они определяют взаимодействие проектируемой системы с другими системами, требования по квалификации персонала, юридические требования, логистические требования, требования среды, этические, экологические и т.п. требования.

# Критерии качества требований

- Критерии качества отдельных положений спецификации требований
- Критерии качества спецификации требований в целом

# Критерии качества отдельных положений спецификации требований

***Полнота***

- Требование является полным тогда и только тогда, когда оно содержит всю информацию, необходимую для разработки соответствующей функциональности, которую следует реализовать в продукте.
- Если в процессе разработки требований не хватает каких-либо данных, необходимо использовать пометку «НО» (необходимо определить) на полях как стандартный флаг для выделения такого места. Все пробелы в каждом фрагменте требований должны быть восполнены, прежде чем спецификация требований будет окончательно утверждена.

***Корректность***

- Требование является корректным тогда и только тогда, когда оно представляет что-либо, требуемое от создаваемого продукта.

***Осуществимость***

- Требование является осуществимым тогда и только тогда, когда оно реализуемо при известных условиях и ограничениях создаваемого продукта и операционной среды, в том числе и при оговоренных сроках и объеме финансирования.

***Необходимость***

- Требование является необходимым тогда и только тогда, когда оно отражает возможность, которая действительно необходима пользователям или которая нужна для соответствия внешним системным требованиям или стандартам. Кроме того, требование должно исходить от лица, которое имеет полномочия на формулирование требований.

# Критерии качества отдельных положений спецификации требований

***Упорядоченность по важности и стабильности***

- Все требования должны быть упорядочены по их важности для заказчика и стабильности. Этот процесс упорядочения особенно важен для управления масштабом. Если ресурсы недостаточны, чтобы в пределах выделенного времени и бюджета реализовать все требования, полезно знать, какие требования являются не столь уж обязательными, а какие заказчик считает критическими.

***Недвусмысленность***

- Требование является недвусмысленным тогда и только тогда, когда его можно однозначно интерпретировать. Если формулировка требований может по-разному интерпретироваться разработчиками, пользователями и другими участниками проекта, вполне может оказаться, что построенная система будет полностью отличаться от той, которую представлял себе заказчик.

***Проверяемость***

- Требования должны поддаваться проверке (быть верифицируемыми). Требование в целом считается верифицируемым тогда и только тогда, когда каждое из составляющих его элементарных требований является верифицируемым. Элементарное требование считается верифицируемым тогда и только тогда, когда существует конечный финансово-эффективный процесс, с помощью которого человек или машина могут определить, что разработанная программная система действительно удовлетворяет данному требованию. Практическая задача состоит в таком определении требований, чтобы можно было впоследствии протестировать их и выяснить, действительно ли они выполняются.

# Критерии качества спецификации требований в целом

***Полнота***

- Набор требований является полным тогда и только тогда, когда он описывает все важные требования, интересующие пользователей, в том числе требования, связанные с функциональными возможностями, производительностью, ограничениями проектирования, атрибутами или внешними интерфейсами.

***Непротиворечивость***

- Набор требований является непротиворечивым тогда и только тогда, когда ни одно его подмножество, состоящее из отдельных требований, не противоречит другим подмножествам. Конфликты могут иметь различную форму и проявляться на различных уровнях детализации.

***Способность к модификации***

- Набор требований является модифицируемым, когда его структура и стиль таковы, что любое изменение требований можно произвести просто, полно и согласованно, не нарушая существующей структуры и стиля всего подмножества. Для этого требуется, чтобы пакет требований имел минимальную избыточность и был хорошо организован, с соответствующим содержанием, указателями и возможностью перекрестных ссылок.

***Трассируемость***

- Набор требований является трассируемым, когда ясно происхождение каждого из составляющих его элементарных требований и существует механизм, который делает возможным обращение к этим требованиям при дальнейших действиях по разработке. Возможность трассировки имеет огромное значение. Разработчики могут использовать ее как для достижения лучшего понимания проекта, так и для обеспечения более высокой степени уверенности, что все требования выполняются.

# Назначение приоритетов требований

- Приоритеты – это способ разрешения борьбы между конкурирующими требованиями за ограниченные ресурсы. Определение относительного приоритета каждой возможности позволяет так планировать разработку, чтобы обеспечивать наибольшую ценность при наименьших затратах. Определение приоритетов наиболее критично для работы в очень строгих временных рамках.
- Менеджер проекта должен сбалансировать желаемый объем проекта и ограничения, определяемые сроком, бюджетом, людскими ресурсами и качеством. Один из способов достижения этого – убрать (или отложить до более поздней версии) требования с низким приоритетом, когда принимаются новые, более важные требования или изменяются другие условия проекта.
- Участие в определении приоритетов требований – одна из обязанностей клиента в отношениях «клиент – разработчик». Обсуждение приоритетов помогает не только определить последовательность реализации требований, но и прояснять ожидания клиентов.
- И клиенты, и разработчики должны внести вклад в определение приоритетов требований. Клиентам больше всего нужны функции, наиболее ценные для бизнеса или удобства работы. Однако, когда разработчики обрисуют затраты, трудоемкость, технический риск или компромиссы, связанные с каждым требованием, клиенты могут передумать и прийти к выводу, что это требование не так важно, как они считали изначально. Разработчики же иногда решают на ранних стадиях реализовать некоторые функции с низким приоритетом из-за их влияния на архитектуру системы.
- Оценивая приоритеты, учитывайте связи и взаимосвязи между различными требованиями, а также их соответствие бизнес-целям проекта.

# Шкала приоритетов

Один из способов оценки приоритетов предлагает учитывать два измерения: важность и срочность. Каждое требование считается важным либо не важным и срочным либо не срочным. Получаются четыре комбинации для определения шкалы приоритетов:

- *Требования с высоким приоритетом* (high priority) – и важные (пользователям нужны функции), и срочные (они необходимы уже в следующем выпуске). Некоторые требования приходится включать в эту категорию согласно контрактным или юридическим обязательствам либо из-за непреодолимых бизнес-причин.
- *Требования со средним приоритетом* (medium priority) – важные (пользователям нужны функции), но не срочные (они могут ждать следующего выпуска).
- *Требования с низким приоритетом* (low priority) – не важные (пользователи при необходимости могут обойтись без этой функций) и не срочные (пользователи могут ждать, причем вечно).
- Требования в четвертой клетке кажутся срочными, но в действительности – они не важны. Не тратьте время на работу над ними – они не сделают продукт более ценным.

# Аналитические и математические методики определения приоритетов

- Аналитические и математические методики основываются на оценке относительной ценности и относительной стоимости каждого требования.
- В соответствии с этими методиками требования с самым высоким приоритетом – это те, что обеспечивают большую ценность продукта при меньшей стоимости.

- Quality Function Deployment (QFD), всесторонний метод определения относительной ценности для клиента
- Подход, заимствованный из Total Quality Management (TQM), позволяющий оценить каждое требование по нескольким весомым критериям успеха проекта и подсчитать количество баллов для назначения приоритетов требований.
- Однако в силу сложности лишь немногие организации разработчики ПО применяют QFD или TQM

# Матрица определения приоритетов

- Используется для оценки относительных приоритетов наборов вариантов использования, функций или функциональных требований
- Эта матрица заимствует из QFD концепцию обоснования ценности для пользователя - учитывается как выгода для пользователя, если требование реализовано в продукте, так и неудобство, если реализация отсутствует
- Затраты также оцениваются двояко: как стоимость реализации требования и как возможный риск срыва реализации.
- Все показатели: Выгода – B (benefit), Урон – L (loss), Стоимость – C (cost) и Риск – R (risk) оцениваются в баллах по шкале 1-10. Двойная оценка ценности и затрат позволяет снизить влияние субъективных факторов.
- Обобщенная оценка Ценности – W (worth) и Затрат – O (outlay) вычисляется как $W = \alpha_1 B + \alpha_2 L,$ $O = \beta_1 C + \beta_2 R,$
- где α и β – весовые коэффициенты, определяющие предпочтения в парных оценках. Причем должно соблюдаться условие $\alpha_1 + \alpha_2 = \beta_1 + \beta_2 = 1$
- При прочих равных условиях требования с наибольшим значением Ценность/Затраты должны иметь наивысший приоритет. Поэтому оценка приоритета P определяется как $P = \frac{W}{K_n O}$
- где коэффициент $K_n$ введен из условия нормировки (значение приоритета лежит в диапазоне 0-1) и равен 10.

| Вес | $\alpha_1$ | $\alpha_2$ | $\beta_1$ | $\beta_2$ | |
|---|---|---|---|---|---|
| Требование | Выгода | Урон | Стоимость | Риск | Приоритет |

# Ограничения применения матрицы

- Применять эту матрицу определения приоритетов к элементам, не имеющим наивысшей ценности.
- Не нужно включать в этот анализ элементы, которые реализуют основные бизнес-функции продукта, или которые являются основными отличительными чертами продукта, а также те, которые необходимы для соответствия юридическим нормам. Если нет возможности назначить этим элементам со временем более низкий приоритет при изменении условий, то необходимо реализовать их в продукте как можно быстрее. Определив элементы, которыми обязательно должен обладать выпускаемый продукт, для оценки оставшихся можно применить рассматриваемую матрицу.

Обычно в процессе назначения приоритетов участвуют:

- менеджер проекта, который ведет процесс, разрешает конфликты и при необходимости адаптирует данные, поступающие от других участников;
- представители клиента – сторонники продукта или маркетологи, предоставляющие информацию о сильных и слабых сторонах продукта;
- представители разработчиков, например руководители команд, сообщающие данные о стоимости и риске.

# Методика оценки

- В таблицу вносятся все функции, варианты использования или требования, для которых определяются приоритеты. Все элементы должны принадлежать к одному уровню абстракции – нельзя смешивать, например, функциональные требования с функциями продукта. Если определенные функции логически связаны (например, вы реализуете функцию В только при наличии функции А), в анализ включайте только ведущую функцию. Поскольку получить приемлемые экспертные оценки для сотен объектов весьма сложно, при наличии большого числа оцениваемых элементов следует предварительно сгруппировать связанные элементы и включить в перечень оцениваемых элементов обобщенные, групповые элементы. При необходимости вы можете провести второй раунд анализа, на более детальном уровне.
- Попросите представителей клиентов оценить относительную выгоду, которую каждый элемент дает клиенту или бизнесу, по шкале от 1 до 10 баллов: 1 балл означает, что никто не находит его полезным, а 10 – что этот элемент крайне ценен. Эти оценки выгоды по существу отражают связь оцениваемых элементов с бизнес-требованиями к продукту.
- Аналогично попросите представителей клиентов оценить относительный урон, который потерпит клиент или бизнес, если элемент не будет включен в продукт. Здесь 1 балл означает, что никто не расстроится, если этого элемента не будет; 10 показывает серьезный урон. Оценивая урон, учитывайте, насколько расстроятся клиенты, если определенный элемент не будет реализован
- Попросите разработчиков оценить относительную стоимость реализации каждого элемента, опять-таки по шкале от 1 (легко и быстро) до 10 (трудоемко и дорого). При оценке стоимости разработчики должны учитывать сложность реализации, объем требуемой работы над интерфейсом пользователя, потенциальную возможность повторного использования существующего кода, объем необходимого тестирования и документации и т.д.
- Подобным же образом, разработчики должны оценить относительную степень технического или другого риска, связанного с каждым элементом. Оценка в 1 балл означает, подобное команда разработчиков уже реализовывала и имеет технологические наработки, 10 баллов означает серьезную озабоченность самой возможностью реализации, нехваткой сотрудников с необходимым опытом или использованием неиспытанных или незнакомых средств и технологий.
- После получения всех оценок определяется значения приоритета по указанной выше формуле.
- Полученный список следует отсортировать по уменьшению подсчитанного приоритета (при равенстве приоритетов – по убыванию выгоды или ценности).

- Элементы вверху списка характеризуются наиболее благоприятным сочетанием ценности, стоимости и риска и поэтому – при равенстве остальных факторов – должны иметь наивысший приоритет.
- Элементы, имеющие низкие рейтинги и выгоды, и урона, увеличивают затраты, но имеют малую ценность. Поэтому являются потенциальными кандидатами на исключение.
- Точность этого метода ограничена возможностью команды оценивать выгоду, урон, стоимость и риск для каждого элемента. Поэтому используйте подсчитанную последовательность приоритетов только как ориентир. Представители клиентов и разработчиков должны проверить итоговую таблицу, чтобы прийти к соглашению о рейтингах и результирующей последовательности приоритетов.
- Изначально весовые коэффициенты α и β, определяющие предпочтения в парных оценках, равны 0.5, т.е. компоненты оценок имеют одинаковый вес. Однако, используя наборы готовых требований из предыдущих проектов, эти коэффициенты можно настроить. Для этого необходимо их изменить так, чтобы подсчитанная последовательность приоритетов хорошо сочеталась с вашей постфактум-оценкой того, насколько в действительности важны требования в вашем калибровочном наборе.

# Согласование оценок

- Заинтересованные в проекте лица обычно по-разному оценивают рассматриваемые элементы. При малом количестве оцениваемых элементов и небольшом разбросе оценок согласованные оценки могут быть получены в ходе простого обсуждения. В противном случае необходима формальная процедура согласования оценок. В качестве такой оценки может выступать взвешенная оценка:

$$R_w = \sum_{i=1}^{N} k_i R_i, \quad \sum_{i=1}^{N} k_i = 1$$

где $k_i$ – вес i-го эксперта, а $R_i$ – его оценка.

- При назначении весов экспертов следует исходить из их влияния на принятие решений, касающихся проекта. При назначении весов экспертам, оценивающим показатели выгоды и урона, можно исходить из выявленных классов пользователей. Имеющиеся классы необходимо проранжировать по степени их влияния на принятие решений, касающихся проекта, и определить их веса как показано в таблице

# Определение весов экспертов

Например, эксперты принадлежат к классам заказчиков, привилегированных и непривилегированных пользователей.

| Группа экспертов | Ранг группы | Кол-во экспертов в группе | Общий ранг группы | Вес группы | Вес эксперта |
|---|---|---|---|---|---|
| Привилегированные пользователи | 3 | 2 | 6 | 0.46 | 0.23 |
| Заказчики | 2 | 2 | 4 | 0.31 | 0.15 |
| Непривилегированные пользователи | 1 | 3 | 3 | 0.23 | 0.08 |
| | | Суммарный ранг: | **13** | | |

**Примечания:**

1. Несмотря на кажущийся приоритет заказчиков, привилегированные пользователи имеют более высокий ранг, поскольку их работа с продуктом определяет, способствует ли он достижению заявленных бизнес-целей или нет.
2. *Общий ранг группы* определяется как произведение *ранга группы* на *количество экспертов в группе*.
3. *Вес группы* определяется как отношение *общего ранга группы* к *суммарному рангу*.
4. *Вес эксперта* определяется как частное от деления *веса группы* на *количество экспертов в группе*.

При назначении весов экспертам, оценивающим показатели стоимости и риска, можно исходить из аналогичных соображений, выделив соответствующие группы экспертов. Причем наивысший ранг также следует присваивать тем группам, от которых в наибольшей степени зависит успешность реализации проекта.

# Обеспечение ресурсами

- Обеспечение ресурсами процесса разработки требований осуществляется на основе утвержденного Плана-графика разработки требований

# Обучение

- Не следует предполагать, что участники проекта интуитивно знают, как сотрудничать при формулировании требований. Всех участников необходимо информировать о их правах и обязанностях с тем, чтобы организовать взаимодействие наиболее эффективно.
- Более того, не все разработчики программного обеспечения знают теорию разработки требований, хотя на определенном этапе профессиональной деятельности многие из них осваивают обязанности аналитика требований и работают с клиентами, собирая, анализируя и документируя требования. Обучение позволяет повысить профессиональные навыки сотрудников, выполняющих роли аналитиков, но не может компенсировать нехватку навыков межличностного общения и отсутствие интереса к делу.
- Процесс формулирования требований весьма важен, и поэтому все участники проекта должны понимать концепцию и приемы формулирования требований. Эффективный способ создать команду – собрать участников проекта на однодневный семинар, где и обсудить все процедуры разработки требований. Стороны смогут глубже понять проблемы, стоящие перед остальными, а также то, что необходимо участникам друг от друга для успеха проекта. Аналогичным образом разработчиков следует просветить о концепциях и терминологии предметной области.

# Обучение

**Обучение аналитиков требований**

- Всем членам команды, которые будут исполнять функции аналитиков, необходимо научиться приемам и методам разработки требований. Однако этого недостаточно. Поскольку именно на аналитиков требований возлагается организация и проведение разработки требований, для них дополнительно необходимо организовать семинары по межличностному общению, организации работы групп, ведению переговоров и разрешению конфликтов. Полезно также организовать семинары по соответствующим предметным областям.

**Ознакомление пользователей и менеджеров с требованиями**

- Пользователи, которые будут принимать участие в разработке ПО, должны пройти непродолжительный тренинг (один-два дня), чтоб научиться формулировать требования. Он полезен и для менеджеров по разработке и по работе с клиентами. Подобное обучение поможет понять особое значение выявления требований, суть процесса их разработки, а также опасность пренебрежения ими.

**Ознакомление разработчиков с концепциями предметной области**

- Чтобы помочь разработчикам в общих чертах понять предметную область, перед каждой разработкой требований необходимо провести семинар, на котором познакомить их с бизнесом клиента, терминологией и назначением создаваемого продукта. Это уменьшит вероятность путаницы, непонимания и доработок. Можно также на время проекта назначить каждому разработчику «личного пользователя», который будет разъяснять профессиональные термины и бизнес-концепции.

**Создание бизнес-словаря**

- Словарь со специализированными терминами из предметной области снизит вероятность непонимания. Включите в него синонимы, термины, имеющие несколько значений, и термины, имеющие в предметной области и повседневной жизни разные значения. Таким словарем желательно снабдить всех участников проекта.

# Особенности разработки требований к программным системам

- Разработка требований – это первый из основных процессов создания программных систем. Этот процесс состоит из следующих основных этапов