

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технологический университет «СТАНКИН»**

**(ФГБОУ ВО МГТУ «СТАНКИН»)**

|  |  |
| --- | --- |
| **Институт**  информационных технологий | **Кафедра**  информационных систем |

**Реферат № 4**

по дисциплине «**Основы программной инженерии**»

на тему: «Разработка требований к программному проекту»

|  |  |  |
| --- | --- | --- |
| **Студент** группа ИДБ–22–06 |  | **Мустафаева П.М.** |
|  | подпись |  |
| **Руководитель**  к.э.н., доцент |  | **Ахмедов Э.Р.** |
|  | подпись |  |

**1. Почему необходимо определение требований к разрабатываемым ПП?  
Приведите схему структурирования уровней требований.**

Под требованиями понимаются потребности людей (заказчиков, пользователей, разработчиков), заинтересованных в создании ПП, которые должны быть надлежащим образом представлены и оформлены в виде технического задания или иного документа.

Определение требований – это важный этап разработки программного обеспечения, потому что:

* Обеспечивает соответствие ожиданиям заказчика – позволяет четко определить, какие функции должен выполнять ПП.
* Снижает затраты и риски – уменьшает вероятность изменений на поздних этапах разработки, что сокращает издержки.
* Повышает качество продукта – четко сформулированные требования помогают создать функциональное, удобное и надежное ПО.
* Облегчает тестирование и поддержку – требования служат основой для тест-кейсов и документации.
* Оптимизирует взаимодействие между участниками проекта – разработчики, аналитики, тестировщики и заказчики работают в одном направлении.

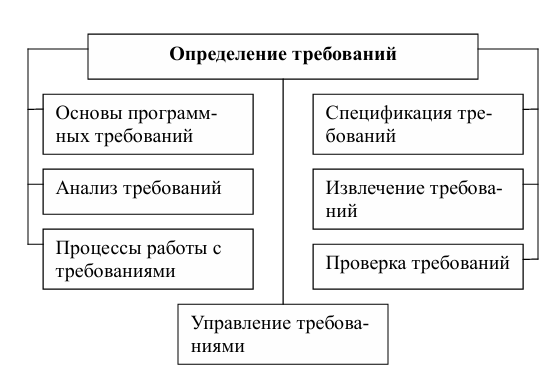


Рис. 1 Область знаний этапа «Определение требований»

Бизнес-требования: определяют цели и задачи проекта, выгоды от его реализации, основные ограничения.

Пользовательские требования: описывают сценарии взаимодействия пользователя с системой, требования к интерфейсу, ограничения на использование.

Функциональные требования: определяют, какие функции должна выполнять система, входные и выходные данные, логику обработки.

Нефункциональные требования: включают требования к производительности, безопасности, масштабируемости, надежности.

Системные требования: описывают аппаратные и программные зависимости, интеграцию с другими системами.

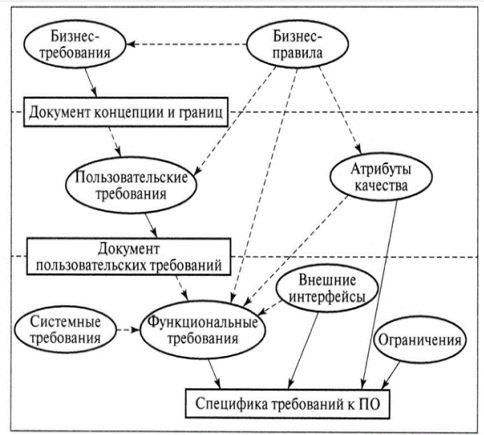


Рис. 2 Группы требований к ПП

**2. Каковы основные разделы разработки требований в соответствии  
с ядром знаний SWEBOK? Дайте классификацию требований.**

Ядро знаний SWEBOK является основополагающим научно-техническим документом, который отображает мнение многих зарубежных и отечественных специалистов в области программной инженерии и согласуется с современными регламентированными процессами ЖЦ ПО стандарта ISO/IEC 12207. В этом ядре знаний содержится описание 10 областей, каждая из которых представлена согласно принятой всеми участниками создания этого ядра общей схемы описания, включающей определение понятийного аппарата, методов и средств, а также инструментов поддержки инженерной деятельности. В каждой области описывается определенный запас знаний, который должен практически использоваться в соответствующих процессах ЖЦ.

Ядро знаний SWEBOK (Software Engineering Body of Knowledge) определяет процесс разработки требований как инженерию требований, которая включает следующие основные разделы:

**Инженерия требований к ПО** – это дисциплина анализа и документирования требований к ПО, которая заключается в преобразовании предложенных заказчиком требований к системе в описании требований к ПО и их валидация. Она базируется на модели процесса определения требований и действующих лицах, обеспечивающих управление и формирование требований, а также на методах достижения показателей качества.

**Выявление требований** – это процесс извлечения информации из разных источников (договоров, материалов аналитиков по декомпозиции задач и функций системы и др.), проведения технических мероприятий (собеседований, собраний и др.) для формирования отдельных требований к продукту и к процессу разработки. Исполнитель должен согласовать требования с заказчиком.

**Анализ требований** – процесс изучения потребностей и целей пользователей, классификация и преобразование их к требованиям к системе, аппаратуре и ПО, установление и разрешение конфликтов между требованиями, определение приоритетов, границ системы и принципов взаимодействия со средой функционирования. Требования могут быть функциональные и нефункциональные, которые определяют соответственно внешние и внутренние характеристики системы.

**Спецификация требований к ПО** – процесс формализованного описания функциональных и нефункциональных требований, требований к характеристикам качества в соответствии со стандартом качества ISO/IEC 9126-94, которые будут отрабатываться на этапах ЖЦ ПО. В спецификации требований отражается структура ПО, требования к функциям, качеству и документации, а также задается в общих чертах архитектура системы и ПО, алгоритмы, логика управления и структура данных. Специфицируются также системные требования, нефункциональные требования и требования к взаимодействию с другими компонентами и платформами (БД, СУБД, сеть и др.).

**Валидация требований** – это проверка изложенных в спецификации требований, выполняющаяся для того, чтобы путем отслеживания источников требований убедиться, что они определяют именно данную систему. Заказчик и разработчик ПО проводят экспертизу сформированного варианта требований с тем чтобы разработчик мог далее проводить проектирование ПО. Один из методов валидации – прототипирование, т.е. быстрая отработка отдельных требований на конкретном инструменте и исследование масштабов изменения требований, измерение объема функциональности и стоимости, а также создание моделей оценки зрелости требований.

**Верификация требований** – это процесс проверки правильности спецификаций требований на их соответствие, непротиворечивость, полноту и выполнимость, а также на соответствие стандартам. В результате проверки требований делается согласованный выходной документ, устанавливающий полноту и корректность требований к ПО, а также возможность продолжить проектирование ПО.

**Управление требованиями** – это руководство процессами формирования требований на всех этапах ЖЦ и включает управление изменениями и атрибутами требований, а также проведение мониторинга – восстановления источника требований. Управление изменениями возникает после того, когда ПО начинает работать в заданной среде и обнаруживаются ошибки в трактовке требований либо в невыполнении некоторого отдельного требования и т.п.

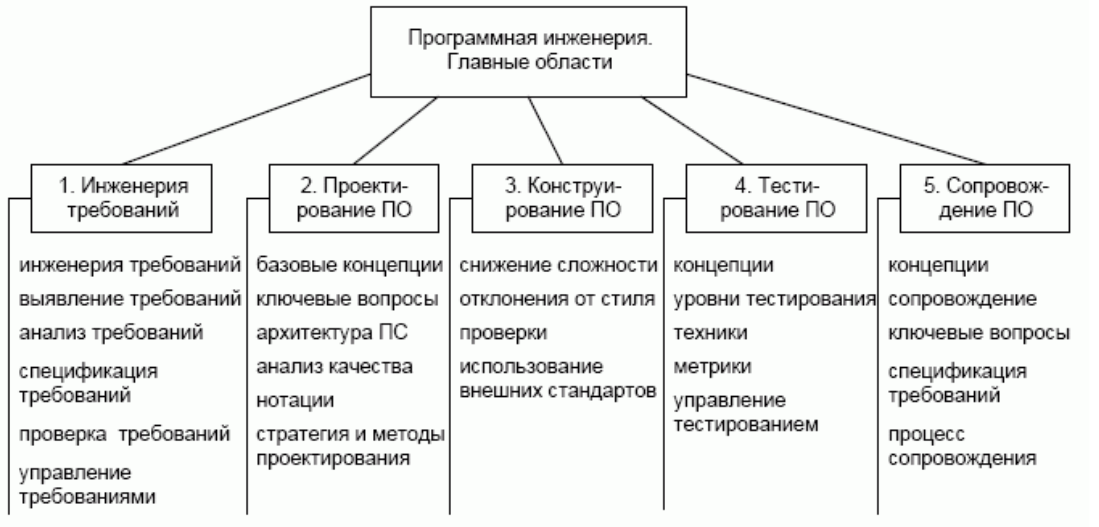


Рис. 3 Основные области знаний SWEBOK

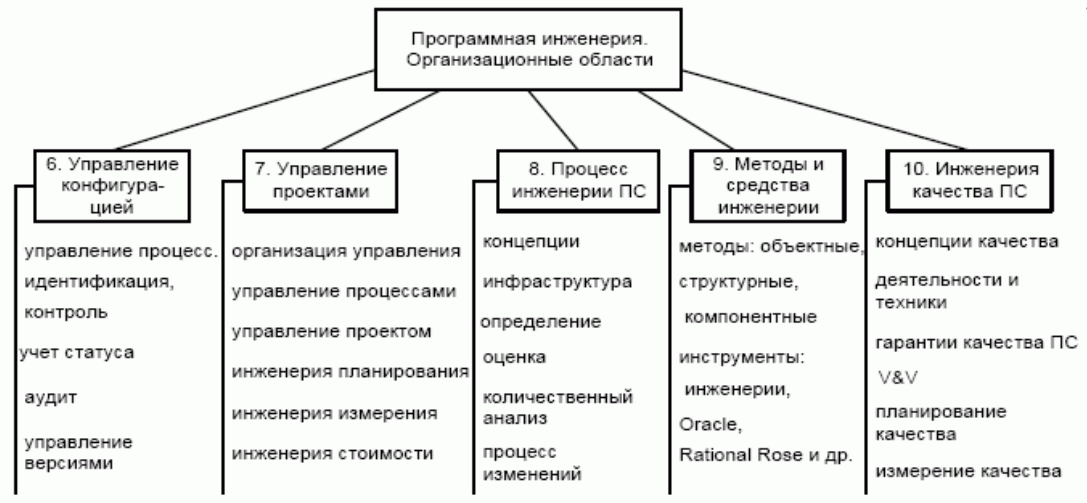


Рис. 4 Организационные области знаний SWEBOK

**Классификация требований**

1. По уровню абстракции:

* Бизнес-требования – цели и задачи проекта.
* Пользовательские требования – сценарии взаимодействия пользователей с системой.
* Системные требования – описание ограничений и технических условий.

1. По функциональности:

* Функциональные требования – определяют, какие функции должна выполнять система.
* Нефункциональные требования – характеристики системы (производительность, безопасность, надежность).

1. По изменчивости:

* Жесткие требования – обязательные к выполнению.
* Гибкие требования – желательные, но не критичные.

1. По источнику происхождения:

* Явные требования – формализованные заказчиком.
* Неявные требования – подразумеваемые требования, которые очевидны, но не всегда явно указаны.

**3. Каково назначение функциональных и нефункциональных требований?**

Функциональные требования описывают "поведение" системы и информацию, с которой система будет работать. Они описывают возможности системы в терминах поведения или выполняемых операций

Цель использования данной группы требований (как следует из названия) заключается в регламентации возможностей и соответствующем им поведении разрабатываемого программного продукта.

Функциональные требования должны отвечать на вопрос - каким образом должны быть алгоритмизированы процессы информационного продукта, чтобы взаимодействие между пользователем и системой удовлетворяло потребности стэйкхолдеров (разнообразные организации или группы лиц, которые как-то связаны или могут быть заинтересованы в деятельности и успехе развития отдельно взятой компании)?

Именно с помощью функциональных требований, в большинстве случаев, определяются рамки работ по процессам, сопровождающим цикл создания программных продуктов. Данный тип требований устанавливает:

Цели разрабатываемого функционала;

* Задачи, которые должны быть выполнены для достижения поставленной цели;
* Сервисы, поддерживающие выполнение задач;
* Многое другое, касающееся непосредственно функциональных возможностей и особенностей программного продукта.

Примеры функциональных требований

* Регистрация пользователей: Система должна предоставлять возможность пользователям регистрироваться, вводя свои данные, такие как имя, адрес электронной почты и пароль. Это позволяет создать учетную запись и получить доступ к персонализированным функциям системы.
* Авторизация и аутентификация: Система должна проверять учетные данные пользователей при входе и предоставлять доступ только авторизованным пользователям. Это обеспечивает безопасность и конфиденциальность данных пользователей.
* Управление заказами: Система должна позволять пользователям создавать, изменять и удалять заказы. Это включает в себя возможность добавления товаров в корзину, оформления заказа и отслеживания его статуса.
* Поиск и фильтрация товаров: Система должна предоставлять возможность поиска товаров по различным критериям, таким как категория, цена и рейтинг. Это помогает пользователям быстро находить нужные товары и принимать решения о покупке.
* Отправка уведомлений: Система должна отправлять уведомления пользователям о статусе их заказов и других важных событиях. Это может включать в себя уведомления о подтверждении заказа, его отправке и доставке.

Нефункциональные требования описывают, как система должна выполнять свои функции. Они включают в себя характеристики, которые влияют на производительность, надежность, безопасность и удобство использования системы. Эти требования часто называют качественными атрибутами системы. Нефункциональные требования помогают определить, насколько хорошо система будет выполнять свои функции и удовлетворять потребности пользователей.

Примеры нефункциональных требований

* Производительность: Система должна обрабатывать не менее 1000 запросов в секунду при пиковых нагрузках. Это обеспечивает быструю и эффективную работу системы даже при высокой нагрузке.
* Надежность: Система должна обеспечивать доступность не менее 99.9% времени в течение года. Это означает, что система должна быть доступна для пользователей практически всегда, за исключением минимальных периодов обслуживания или непредвиденных сбоев.
* Безопасность: Система должна использовать шифрование данных при передаче и хранении, а также обеспечивать защиту от несанкционированного доступа. Это помогает защитить конфиденциальные данные пользователей и предотвратить утечки информации.
* Удобство использования: Интерфейс системы должен быть интуитивно понятным и доступным для пользователей с различным уровнем технической подготовки. Это помогает пользователям быстро освоить систему и эффективно использовать ее функции.
* Масштабируемость: Система должна поддерживать возможность увеличения нагрузки без значительного снижения производительности. Это позволяет системе расти и адаптироваться к увеличению числа пользователей и объема данных.

**4. Какие нефункциональные требования вы знаете? Перечислите количественные показатели нефункциональных требований.**

Основные виды нефункциональных требований:

* производительность – нагрузка, с которой справляется система, выражается в количестве одновременно работающих пользователей и/или в числе запросов в единицу времени, обычно rps, request per second.
* доступность – степень работоспособности и доступности решения для использования, часто выражается в процентах времени;
* совместимость – степень успешности взаимодействия решения с другими окружающими компонентами (бизнес-процессами, информационными системами, аппаратным обеспечением и пр.);
* функциональность – степень соответствия функций решения потребностям пользователей, включая пригодность, точность, совместимость, т.е. корреляция с требованиями стейкхолдеров;
* ремонтопригодность – легкость изменения решения или его компонента, чтобы улучшить их, исправить ошибки или адаптировать к изменениям окружающей среды;
* эффективность работы – способность решения или его компонента выполнять свои целевые функции с минимальным потреблением ресурсов в контексте или временном периоде, например, в условиях пиковой нагрузки;
* надежность – способность решения или его компонента выполнять требуемые функции в определенных условиях в течение конкретного периода времени, например, наработка на отказ, измеряемая в часах и равная среднему времени работы устройства до сбоя;
* масштабируемость – способность решения расти или развиваться по мере роста объемов данных и количества пользователей;
* расширяемость – возможность добавления в решение новых функциональных возможностей.
* переносимость – легкость переноса решения или его компонента из одной среды в другую, например, миграция на новую версию операционной системы или аппаратной платформы;
* безопасность – защита данных и компонентов решения от случайного или злонамеренного доступа, неправомерного использования, разрушения или раскрытия.
* удобство использования – легкость взаимодействия пользователя с решением, включая простоту ежедневной работы и обучения;
* сертификация – соответствие отдельным стандартам или отраслевым соглашениям;
* соответствие – нормативные, финансовые или правовые ограничения в зависимости от контекста и юрисдикции;
* локализация – адаптация текстовых и графических компонентов интерфейса, а также представления данных к языкам, законам, валютам и особенностям культуры пользователей в определенной местности или отрасли;
* соглашения об уровне обслуживания между поставщиком и пользователем решения. На практике это свойство тесно связано с доступностью.

Некоторые количественные показатели нефункциональных требований:

* Скорость. Количество выполненных транзакций в секунду, время реакции на действия пользователя, время обновления экрана.
* Размер. Килобайты, количество модулей памяти.
* Простота эксплуатации. Время обучения персонала, количество статей в справочной системе.
* Надежность. Средняя продолжительность времени между двумя последовательными проявлениями ошибок в системе, вероятность выхода системы из строя, коэффициент готовности системы.
* Устойчивость к сбоям. Время восстановления системы после сбоя, процент событий, приводящих к сбоям, вероятность порчи данных при сбоях.
* Переносимость. Процент машинно-зависимых операторов, количество машинно-зависимых подсистем.

**5. Какие способы записи спецификаций требований вам известны?**

Способы записи спецификаций требований:

* Структурированный естественный язык – использование стандартных форм и шаблонов для написания спецификации.
* Языки описания программ – использование специальных структурированных языков, подобных языкам программирования, где спецификация требований строится на основе выбранной операционной модели системы.
* Графические нотации – графический язык, использующий для описания функциональных требований диаграммы и блок-схемы, дополненные текстовыми пояснениями. Наиболее известный пример такого графического языка —диаграммы структурного анализа и проектирования ПО (SADT).
* Математические спецификации – это системы нотаций, основанные на математических концепциях, таких, как теория конечных автоматов или теория множеств. Это формализованная однозначная и лишенная двусмысленности запись системных требований. Однако многие заказчики ПО не понимают формальных спецификаций, вследствие чего возникают определенные проблемы при заключении контрактов на разработку программных продуктов.

**6. Охарактеризуйте процесс разработки требований.**

Разработка требований - это процесс, включающий в себя мероприятия, необходимые для создания и утверждения документа, содержащего спецификацию системных требований. Различают четыре основных этапа процесса разработки требовании:

* анализ технической осуществимости создания системы;
* формирование и анализ требований;
* специфицирование требований и создание соответствующей документации;
* аттестация этих требований.



Рис. 5 Процесс разработки требований

Разработка требований начинается с анализа технической осуществимости создания системы. На этом этапе проводится исследование возможностей реализации проекта с учетом существующих технологий, ограничений и ресурсов. Оценивается, можно ли реализовать систему в заданных условиях, изучаются доступные технологические решения, оцениваются потенциальные риски, стоимость разработки и дальнейшего сопровождения. Также анализируются возможные ограничения, такие как законодательные нормы, аппаратные и программные требования.

Следующим этапом является формирование и анализ требований. Здесь происходит сбор информации от заказчиков, пользователей и других заинтересованных сторон для определения потребностей, которые система должна удовлетворять. Требования классифицируются на функциональные и нефункциональные, выявляются зависимости между ними, проводится их уточнение и приоритизация. Анализ требований включает выявление возможных конфликтов между ожиданиями различных групп пользователей, а также проверку их полноты, непротиворечивости и выполнимости.

После этого выполняется специфицирование требований и создание соответствующей документации. На основе проведенного анализа составляется подробное описание требований, включающее детализированные характеристики системы, сценарии использования и возможные ограничения. Документ спецификации требований оформляется в соответствии с установленными стандартами, такими как IEEE 830, и может содержать текстовые описания, диаграммы UML, таблицы или другие графические модели, облегчающие восприятие информации.

Заключительным этапом является аттестация требований. Она включает в себя проверку корректности, полноты и согласованности сформулированных требований. На этом этапе заказчики и разработчики совместно анализируют документ, выявляют возможные недочеты, проводят ревизию требований и утверждают их. Также могут выполняться дополнительные валидационные процедуры, такие как моделирование системы, создание прототипов или экспертные оценки. После успешного прохождения аттестации требования считаются утвержденными и используются в дальнейших этапах разработки программного продукта.

**7. Что такое анализ осуществимости требований?**

Анализ осуществимости требований – это процесс оценки того, насколько реально выполнить заданные требования с точки зрения различных ограничений: технических, финансовых, временных, юридических и организационных.

Разработка требований – это процесс, включающий мероприятия, необходимые для создания и утверждения документа, содержащего спецификацию системных требований. Для новых программных систем процесс разработки требований должен начинаться с анализа осуществимости. Началом такого анализа является общее описание системы и ее назначения, а результатом анализа – отчет, в котором должна быть четкая рекомендация, продолжать или нет процесс разработки требований проектируемой системы. Другими словами, анализ осуществимости должен осветить следующие вопросы.

Отвечает ли система общим и бизнес-целям организации-заказчика и организации-разработчика?

Можно ли реализовать систему, используя существующие на данный момент технологии и не выходя за пределы заданной стоимости?

Можно ли объединить систему с другими системами, которые уже эксплуатируются?

Критическим является вопрос, будет ли система соответствовать целям организации. Если система не соответствует этим целям, она не представляет никакой ценности для организации. В то же время многие организации разрабатывают системы, не соответствующие их целям, либо не совсем ясно понимая эти цели, либо под влиянием политических или общественных факторов.

Выполнение анализа осуществимости включает сбор и анализ информации о будущей системе и написание соответствующего отчета. Сначала следует определить, какая именно информация необходима, чтобы ответить на поставленные выше вопросы. Например, эту информацию можно получить, ответив на следующее:

1. Что произойдет с организацией, если система не будет введена в эксплуатацию?

2. Какие текущие проблемы существуют в организации и как новая система поможет их решить?

3. Каким образом система будет способствовать целям бизнеса?

4. Требует ли разработка системы технологии, которая до этого не использовалась в организации?

Далее необходимо определить источники информации. Это могут быть менеджеры отделов, где система будет использоваться, разработчики программного обеспечения, знакомые с типом будущей системы, технологи, конечные пользователи и т.д.

После обработки собранной информации готовится отчет по анализу осуществимости создания системы. В нем должны быть даны рекомендации относительно продолжения разработки системы. Могут быть предложены изменения бюджета и графика работ по созданию системы или предъявлены более высокие требования к системе.

**8. Каким образом осуществляется извлечение требований?**

Извлечение требований – это процесс выявления, сбора и документирования потребностей пользователей и других заинтересованных сторон, которые система должна удовлетворять. Этот процесс является одним из ключевых этапов разработки требований и осуществляется различными методами.

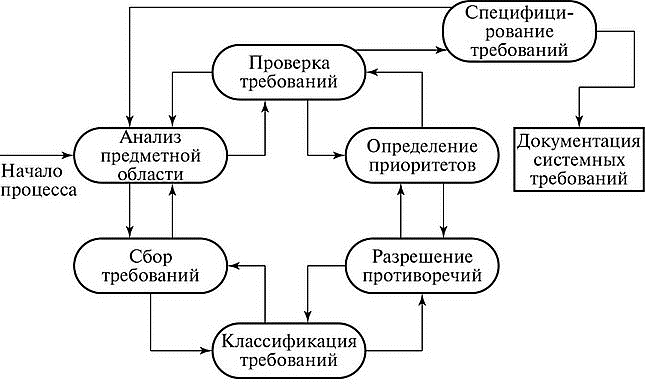
Извлечение требований начинается с идентификации всех заинтересованных сторон, включая заказчиков, конечных пользователей, разработчиков, бизнес-аналитиков и регуляторов. Важно учитывать ожидания каждой группы, так как они могут существенно различаться. Далее проводится сбор информации о существующих процессах, проблемах, которые должна решить новая система, и возможных ограничениях.

Существует множество практик и подходов, позволяющих добиться действительно стройной системы требований, отвечающих реальным потребностям и приоритетам заказчиков. Среди них можно выделить следующие:

* Интервьюрирование – традиционный подход извлечения требований; информация должна быть проанализирована и трансформирована в требования, таким образом, информация от пользователя является «входом» в процессы сбора требований, а сами требования – «выходом» этих процессов;
* Сценарии – контекст для сбора пользовательских требований, определяющий ответы на вопросы «что если» и «как это делается» в отношении бизнес-процессов, реализуемых пользователями;
* Прототипы – отличный инструмент для уточнения и/или детализации требований; существуют разные подходы к прототипированию – от «бумажных» моделей до пилотных подсистем, реализуемых как самостоятельные (в терминах управления ресурсами) проекты или бета-версии продуктов; часто прототипы постепенно трансформируются в результаты проекта и используются для проверки и утверждения требований;
* «Разъясняющие встречи» – термин, пришедший из общей практики менеджмента и базирующийся на идеях сотрудничества заинтересованных лиц для совместного анализа путей решения проблем, определения и предупреждения рисков и т.п. В отличие от «обычного» «мозгового штурма», как исключительной формы обсуждения тех или иных задач (часто в критические моменты работ над проектом), «запланированный мозговой штурм» – особая форма встреч участников проекта и заинтересованных лиц со стороны заказчика, посвященная обсуждению тех вопросов, ответы на которые не могут быть определены в результате обычных интервью и которые требуют вовлечения большего количества лиц, чем просто пары «пользователь-аналитик»;
* Наблюдение – подразумевает непосредственное присутствие аналитиков и инженеров рядом с пользователем в процессе выполнения последним его работ по обеспечению функционирования бизнес-процессов: в определенной степени можно провести аналогию с практикой присутствия представителя заказчика в проектной группе исполнителя; данная техника является достаточно затратной, но, в то же время, очень эффективной, а иногда – просто незаменимой, особенно, если речь идет о достаточно сложных и взаимосвязанных бизнес-процессах.

**9. Охарактеризуйте процесс формирования и анализа требований.**

Обобщенная модель процесса формирования и анализа требований показана на рис. 6. Каждая организация использует собственный вариант этой модели, зависящий от «местных факторов»: опыта работы коллектива разработчиков, типа разрабатываемой системы, используемых стандартов и т.д.

Рис. 6 Процесс формирования и анализа требований

Процесс формирования и анализа требований проходит через ряд этапов:

1. Анализ предметной области. Прежде чем собирать и формулировать требования, аналитики должны глубоко изучить предметную область, в которой будет использоваться система. Это включает изучение существующих бизнес-процессов, выявление текущих проблем и ограничений, понимание ключевых задач, которые должна решать система. Аналитики анализируют документацию, проводят интервью с экспертами, исследуют существующие программные решения, если таковые имеются. На этом этапе важно разобраться в терминологии, используемой в предметной области, чтобы исключить возможные недопонимания при описании требований.

2. Сбор требований. На этом этапе аналитики взаимодействуют с ключевыми заинтересованными сторонами (заказчиками, пользователями, регуляторами и разработчиками) для выявления их потребностей и ожиданий. Сбор требований осуществляется различными методами: интервью, опросы, мозговые штурмы, анализ существующих систем и документов, наблюдение за работой пользователей. Параллельно продолжается анализ предметной области для выявления скрытых потребностей, которые пользователи могут не осознавать, но которые важны для успешного функционирования системы.

3. Классификация требований. На этом этапе бесформенный набор требований преобразуется в логически связанные группы требований по категориям (функциональные требования, нефункциональные требования, бизнес-требования и ограничения). Такое разделение помогает упорядочить требования, выявить недостающие элементы и исключить дублирование.

4. Разрешение противоречий. Часто различные заинтересованные стороны предъявляют противоречивые требования, которые необходимо согласовать. Например, пользователи могут хотеть максимально простой интерфейс, а бизнес-заказчики – множество дополнительных функций, усложняющих систему. На этом этапе аналитики выявляют конфликты между требованиями и проводят обсуждения с заинтересованными сторонами, чтобы прийти к компромиссу. Используются методы переговоров, приоритетизации и анализа альтернативных решений.

5. Назначение приоритетов. Не все требования равнозначны: одни из них являются критическими для работы системы, а другие желательными, но не обязательными. На этом этапе требования оцениваются по степени важности и срочности. Приоритеты могут определяться по разным критериям: бизнес-ценность, техническая сложность, зависимость от других требований. Часто применяются методологии, такие как **MoSCoW** (Must-have, Should-have, Could-have, Won’t-have), где требования делятся на обязательные, желательные, возможные и отложенные.

6. Проверка требований. На завершающем этапе аналитики проверяют требования на полноту, непротиворечивость и выполнимость. Анализируется, охватывают ли они все аспекты системы, не содержат ли логических ошибок или двусмысленностей. Проверка может проводиться с помощью моделирования, создания прототипов, экспертной оценки или тестирования требований с помощью различных сценариев. После успешной проверки требования фиксируются в документации и передаются в разработку.

Процесс формирования и анализа требований циклический, с обратной связью от одного этапа к другому. Цикл начинается с анализа предметной облас­ти и заканчивается проверкой требований. Понимание требований предметной области увеличивается в каждом цикле процесса формирования требований.

**10. Определите этапы разработки спецификаций требований.**

Процесс разработки спецификаций требований проходит через несколько ключевых этапов, каждый из которых направлен на создание точного, полного и согласованного описания требований к системе.

1. Сбор и анализ требований

На этом этапе происходит выявление всех требований к системе путем взаимодействия с заказчиками, пользователями и другими заинтересованными сторонами. Используются методы интервью, анкетирования, наблюдения, анализа документации и существующих систем. Затем проводится анализ требований: проверяется их полнота, непротиворечивость и осуществимость.

2. Классификация и структурирование требований

После сбора требований они группируются по категориям, что позволяет лучше организовать их и упростить дальнейшую работу.

3. Формализация и документирование

На этом этапе все собранные и проанализированные требования фиксируются в документе спецификации требований (SRS). Документ должен быть написан в четкой и однозначной форме, чтобы избежать разночтений. Для лучшего понимания могут использоваться таблицы, диаграммы UML, схемы бизнес-процессов и другие визуальные инструменты.

4. Верификация и согласование требований

Разработанный документ проверяется на полноту, логичность и согласованность. Аналитики, заказчики и технические специалисты оценивают спецификацию, выявляют возможные недочеты, противоречия и устраняют их. Часто на этом этапе проводится рецензирование документации, а также тестирование требований с использованием прототипов или сценариев.

5. Утверждение и передача в разработку

После согласования требований документ спецификации утверждается заказчиком и передается в работу разработчикам. На основе спецификации создаются технические задания, проектная документация и планы тестирования.

Этот процесс может быть итеративным: при изменении требований или выявлении новых деталей спецификация дополняется и проходит повторные циклы верификации и утверждения.

**11. Что такое аттестация требований?**

Аттестация должна продемонстрировать, что требования действительно определяют ту систему, которую хочет иметь заказчик. Проверка требований важна, так как ошибки в спецификации требований могут привести к переделке системы и большим затратам, если будут обнаружены во время процесса разработки системы или после введения ее в эксплуатацию.

Во время процесса аттестации должны быть выполнены различные типы проверок требований:

* Проверка правильности требований. Пользователь может считать, что система необ­ходима для выполнения некоторых определенных функций. Однако дальнейшие размышления и анализ могут привести к необходимости введения дополнительных или новых функций. Системы предназначены для разных пользователей с различными потребностями, и поэтому набор требований будет представлять собой некоторый компромисс между требованиями пользователей системы.
* Проверка на непротиворечивость. Спецификация требований не должна содержать про­тиворечий. Это означает, что в требованиях не должно быть противоречащих друг другу ограничений или различных описаний одной и той же системной функции.
* Проверка на полноту. Спецификация требований должна содержать требования, которые определяют все системные функции и ограничения, налагаемые на систему.
* Проверка на выполнимость. На основе знания существующих технологий требования должны быть проверены на возможность их реального выполнения. Здесь также проверяются возможности финансирования и график разработки системы.

**12. Что понимается под итеративной природой работы с требованиями?**

Итеративная природа работы с требованиями означает, что процесс их разработки, анализа и уточнения не является линейным, а проходит в нескольких циклах (итерациях). На каждом этапе могут вноситься изменения, дополнения или корректировки, что позволяет постепенно уточнять и улучшать требования к системе.

Основные аспекты итеративного подхода:

1. Постепенное уточнение требований – на ранних этапах разработки требования могут быть неполными или неточными, но в ходе работы над проектом они детализируются, дополняются и адаптируются.

2. Гибкость и адаптация – по мере появления новой информации или изменения бизнес-условий требования могут корректироваться, чтобы лучше соответствовать реальным потребностям пользователей и возможностям технологий.

3. Многократная проверка и валидация – после каждой итерации требования пересматриваются, тестируются на согласованность, выполнимость и соответствие ожиданиям заказчика.

4. Связь с гибкими методологиями (Agile, Scrum) – в Agile-подходе требования разрабатываются, тестируются и уточняются в каждом спринте, что позволяет оперативно реагировать на изменения.

5. Раннее выявление ошибок – итеративный процесс помогает обнаружить и исправить противоречия, неточности или неоправданные ожидания на ранних стадиях, снижая риски и затраты на последующие исправления.

**13. Охарактеризуйте процесс управления изменениями требований.**

Процесс управления изменениями состоит из трех основных этапов:

1. Анализ проблем изменения спецификации. Этот этап включает выявление несоответствий, пробелов или ошибок в текущих требованиях, а также учет предложений об изменениях, поступающих от заинтересованных сторон. Причины таких предложений могут быть разными: изменение бизнес-процессов, выявление новых потребностей пользователей, технические ограничения или обнаруженные на этапе тестирования недочеты. Важно не только зафиксировать необходимость изменений, но и определить их обоснованность и возможные последствия.

2. Анализ изменений и расчет их стоимости. На этом этапе проводится оценка их влияния на систему в целом, включая архитектуру, функциональность, производительность и интеграцию с другими модулями. Разрабатывается прогноз возможных рисков, сроков реализации и требуемых ресурсов. Оценка включает как техническую, так и экономическую составляющие: определяются затраты на разработку, тестирование и возможное обучение пользователей новым возможностям системы. Этот этап также предусматривает проведение экспертной оценки и согласование изменений с ключевыми участниками проекта.

3. Реализация изменений. Если изменения утверждаются, начинается их реализация: обновляется спецификация требований, вносятся корректировки в проектную документацию и код системы. После внесения изменений проводится тестирование, направленное на проверку корректности их работы и отсутствия побочных эффектов. В случае успешного тестирования изменения вводятся в эксплуатацию, а процесс их реализации фиксируется в системе управления изменениями.

****

Рис. 7 Управление изменениями требований

**14. Приведите классификацию изменяемых требований.**

Изменяемые требования – это требования, которые отображают изменения, сделанные во время разработки системы или после ввода ее в эксплуатацию.

Изменяемые требования можно классифицировать по нескольким критериям:

Непостоянные требования – требования, которые изменяются из-за изменений в окружении системы.

Неожиданно возникающие требования – требования, которые появляются во время разработки системы. В процессе проектирования может возникнуть необходимость добавления новых требований.

Непрямые требования – требования, которые являются результатом внедрения компьютерной системы, способной изменить организационные процессы и показать новые способы работы, которые приведут к новым системным требованиям.

Вторичные требования – требования, которые зависят от особенностей данной системы или от бизнес-проблем организации.

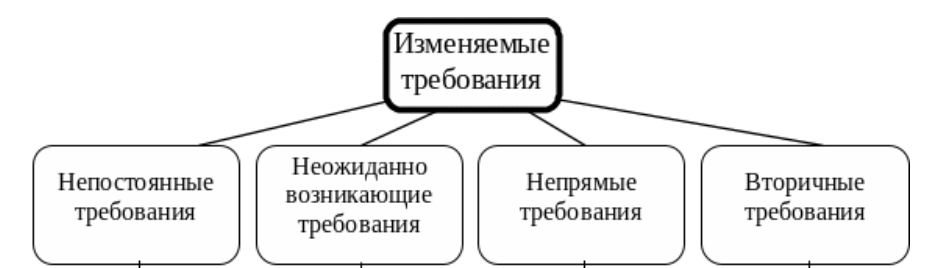


Рис. 8 Классификация изменяемых требований

**15. Что предполагает трассировка требований?**

Трассировка требований – это способность соотнести какой-либо элемент проекта с другим связанным с проектом элементом, особенно с тем, который имеет отношение к техническим требованиям проекта.

Цели трассировки требований:

* повышение качества анализа влияния изменений на продукт;
* повышение качества самого продукта за счёт того, что мы понимаем взаимосвязи и как изменения влияют на продукт;
* сокращение трудозатрат.

Этапы трассировки требований:

1. Вертикальная трассировка, т.е. от общего/главного к частному/детальному. Это принцип иерархии, когда наверху есть главные бизнес-цели проекта, ниже – бизнес-правила, пользовательские требования, варианты использования (use case), системные требования. Так, сверху вниз мы можем проследить связи верхнеуровневых требований к самым детальным. И если мы меняем детальное требование, то можем посмотреть, укладывается ли наше новое требование в верхнеуровневые. И, наоборот, когда мы меняем верхнеуровневое требование, через трассировку мы найдём все детальные требования, на которые оно повлияет.

2. Горизонтальная трассировка, или кроссовая трассировка, когда мы делаем связи одного уровня. Это пример соцсети, когда через соцсети наши друзья все друг с другом так или иначе связаны. Например, если в документации у нас много use case, то они могут пересекаться друг с другом, и быть связаны. Когда мы меняем один use case, благодаря кроссовой трассировке мы можем найти, какие другие use case поменяются у нас в проекте, таким образом отследить влияние на них и учесть в них изменения.