Глава 1: Требования к программному обеспечению

Из SWEBOK

Содержание

АКРОНИМЫ

ЦРУ	Конфиденциальность, целостность и доступность
ДАГ	Направленный ациклический граф
ФШМ	Измерение функционального размера
ВХОД	Международный совет по системной инженерии
UML	Единый язык моделирования
SysML	Язык моделирования систем

ВВЕДЕНИЕ

Область знаний «Требования к программному обеспечению» (KA) связана с выявлением, анализом, спецификацией и проверкой требований к программному обеспечению, а также управлением требованиями в течение всего жизненного цикла программного продукта. Среди исследователей и отраслевых практиков широко признано, что программные проекты становятся критически уязвимыми, когда действия, связанные с требованиями, выполняются плохо.

Требования к программному обеспечению выражают потребности и ограничения, накладываемые на программный продукт, которые способствуют решению некоторой реальной проблемы.

Термин «разработка требований» широко используется в данной области для обозначения систематической обработки требований. Из соображений согласованности термин «инженерия» не будет использоваться в этом KA, кроме как для разработки программного обеспечения как такового. По той же причине термин «инженер требований», который встречается в некоторой литературе, также не будет использоваться. Вместо этого будет использоваться термин «инженер-программист» или, в некоторых конкретных случаях, «специалист по требованиям», причем в последнем случае рассматриваемая роль обычно выполняется лицом, отличным от инженера-программиста. Однако это не означает, что инженер-программист не может выполнять эту функцию.

Риск, присущий предлагаемой разбивке, заключается в том, что можно сделать вывод о водопадоподобном процессе. Чтобы предотвратить это, тема 2, Процесс требований, предназначена для предоставления общего обзора процесса требований путем определения ресурсов и ограничений, в которых работает процесс и которые действуют для его настройки.

Альтернативная декомпозиция может использовать структуру, основанную на продукте (системные требования, требования к программному обеспечению, прототипы, варианты использования и т. д.). Разбивка на основе процессов отражает тот факт, что процесс требований, если он хочет быть успешным, должен рассматриваться как процесс, включающий сложные, тесно связанные действия (как последовательные, так и параллельные), а не как дискретное, однократно выполняемое действие. в начале проекта разработки программного обеспечения.

KA требований к программному обеспечению тесно связан с KA проектирования программного обеспечения, тестирования программного обеспечения, обслуживания программного обеспечения, управления конфигурацией программного обеспечения, управления разработкой программного обеспечения, процесса разработки программного обеспечения, моделей и методов разработки программного обеспечения и качества программного обеспечения.

РАЗБИВКА ТЕМ ПО ТРЕБОВАНИЯМ К ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ

Разбивка тем для требований к программному обеспечению KA показана на рисунке 1.1.

1 Основные требования к программному обеспечению

[1, c4, c4c1, c10c1, c10c4] [2, c1, c6, c12]

1.1 Определение требования к программному обеспечению

По своей сути требование к программному обеспечению — это свойство, которое должно быть продемонстрировано чем-то, чтобы решить какую-то проблему в реальном мире. Он может быть направлен на автоматизацию части задачи для кого-то по поддержке бизнес-процессов организации, исправление недостатков существующего программного обеспечения или управление устройством — и это лишь некоторые из многих проблем, для которых возможны программные решения. Способы функционирования пользователей, бизнес-процессов и устройств обычно сложны. Таким образом, в более широком смысле требования к конкретному программному обеспечению обычно представляют собой сложную комбинацию от разных людей на разных уровнях организации, которые так или иначе вовлечены или связаны с этой функцией из среды, в которой будет работать программное обеспечение.

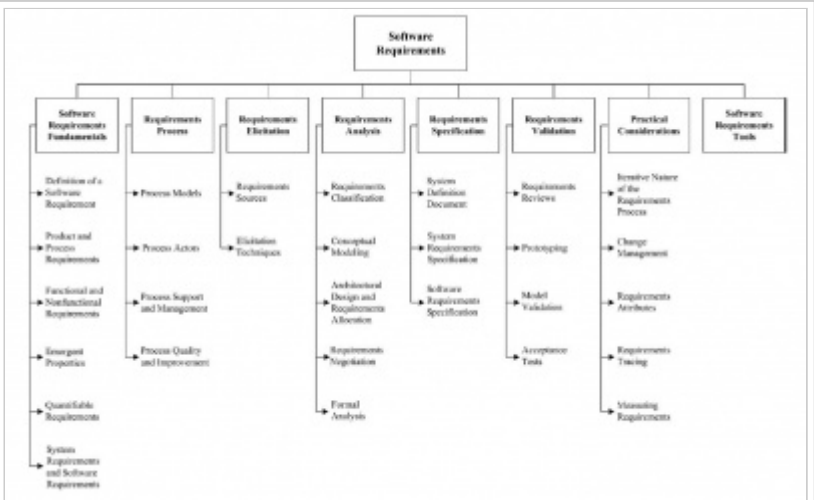

Рисунок 1.1: Разбивка тем для требований к программному обеспечению KA

Существенным свойством всех требований к программному обеспечению является то, что они поддаются проверке как отдельные функции в качестве функциональных требований или на системном уровне в качестве нефункциональных требований. Проверка определенных требований к программному обеспечению может быть сложной или дорогостоящей. Например, проверка требований к пропускной способности колл-центра может потребовать разработки программного обеспечения для моделирования. Требования к программному обеспечению, тестирование программного обеспечения и персонал по обеспечению качества должны гарантировать, что требования могут быть проверены в рамках имеющихся ограничений ресурсов.

1.2 Требования к продукту и процессу

Требование к продукту — это потребность или ограничение в отношении разрабатываемого программного обеспечения (например, «Программное обеспечение должно проверять, что студент соответствует всем предварительным требованиям, прежде чем он или она зарегистрируется на курс»).

Требование к процессу, по сути, является ограничением на разработку программного обеспечения (например, «Программное обеспечение должно быть разработано с использованием процесса RUP»).

Некоторые требования к программному обеспечению генерируют неявные требования к процессам. Одним из примеров является выбор метода проверки. Другим может быть использование особенно строгих методов анализа (таких как методы формальной спецификации) для уменьшения ошибок, которые могут привести к недостаточной надежности. Технологические требования также могут быть наложены непосредственно организацией-разработчиком, их заказчиком или третьей стороной, например органом по регулированию безопасности.

1.3 Функциональные и нефункциональные требования

Функциональные требования описывают функции, которые должно выполнять программное обеспечение; например, форматирование текста или модулирование сигнала. Их иногда называют возможностями или функциями. Функциональное требование также может быть описано как требование, для которого можно написать конечный набор тестовых шагов для проверки его поведения.

Нефункциональные требования ограничивают решение. Нефункциональные требования иногда называют ограничениями или требованиями к качеству. Их можно дополнительно классифицировать в зависимости от того, являются ли они требованиями к производительности, требованиями к ремонтопригодности, требованиями безопасности, требованиями надежности, требованиями безопасности, требованиями функциональной совместимости или одним из многих других типов требований к программному обеспечению (см. Модели и характеристики качества в KA качества программного обеспечения).

1.4 Возникающие свойства

Некоторые требования представляют собой эмерджентные свойства программного обеспечения, то есть требования, которые не могут быть удовлетворены одним компонентом, но зависят от того, как взаимодействуют все программные компоненты. Требуемая пропускная способность колл-центра, например, будет зависеть от того, как телефонная система, информационная система и операторы взаимодействуют в реальных условиях работы. Возникающие свойства в решающей степени зависят от архитектуры системы.

1.5 Количественные требования

Требования к программному обеспечению должны быть сформулированы как можно более четко и недвусмысленно и, при необходимости, в количественном выражении. Важно избегать расплывчатых и непроверяемых требований, интерпретация которых зависит от субъективного суждения («программное обеспечение должно быть надежным»; «программное обеспечение должно быть удобным для пользователя»). Это особенно важно для нефункциональных требований. Ниже приведены два примера количественных требований: программное обеспечение колл-центра должно увеличить пропускную способность центра на 20%; и система должна иметь вероятность возникновения фатальной ошибки в течение любого часа работы менее $1 * 10^{-8}$. Требования к пропускной способности находятся на очень высоком уровне и должны быть использованы для получения ряда подробных требований. Требование надежности будет жестко ограничивать архитектуру системы.

1.6 Системные требования и требования к программному обеспечению

В этой теме "система" означает

> взаимодействующая комбинация элементов для достижения определенной цели. К ним относятся аппаратное обеспечение, программное обеспечение, встроенное программное обеспечение, люди, информация, методы, средства, услуги и другие элементы поддержки,

как определено Международным советом по программному обеспечению и системной инженерии (INCOSE) [3].

Системные требования — это требования к системе в целом. В системе, содержащей программные компоненты, требования к программному обеспечению выводятся из системных требований.

Этот KA определяет «пользовательские требования» ограниченным образом, как требования клиентов системы или конечных пользователей. Системные требования, напротив, включают требования пользователей, требования других заинтересованных сторон (таких как регулирующие органы) и требования без идентифицируемого человеческого источника.

2 Требования Процесс

[1 , c4c4] [2 , c1-4, c6, c22, c23]

Этот раздел знакомит с процессом разработки требований к программному обеспечению, ориентирует оставшиеся пять тем и показывает, как процесс требований согласуется с общим процессом разработки программного обеспечения.

2.1 Модели процессов

Цель этого раздела — дать понимание того, что процесс требований

- не дискретное внешнее действие жизненного цикла программного обеспечения, а скорее процесс, инициированный в начале проекта, который продолжает совершенствоваться на протяжении всего жизненного цикла;
- идентифицирует требования к программному обеспечению как элементы конфигурации и управляет ими, используя те же методы управления конфигурацией программного обеспечения, что и другие продукты процессов жизненного цикла программного обеспечения;
- должны быть адаптированы к организации и контексту проекта.

В частности, эта тема связана с тем, как действия по выявлению, анализу, спецификации и проверке настраиваются для различных типов проектов и ограничений. Тема также включает действия, которые вносят вклад в процесс требований, такие как маркетинг и технико-экономическое обоснование.

2.2 Участники процесса

В этом разделе представлены роли людей, которые участвуют в процессе требований. Этот процесс по своей сути является междисциплинарным, и специалист по требованиям должен быть посредником между областью заинтересованной стороны и областью разработки программного обеспечения. Помимо специалиста по требованиям, часто задействовано много людей, каждый из которых заинтересован в программном обеспечении. Заинтересованные стороны будут различаться в зависимости от проекта, но всегда будут включать пользователей/операторов и клиентов (которые не обязательно должны быть одними и теми же).

Типичные примеры заинтересованных сторон программного обеспечения включают (но не ограничиваются ими) следующее:

- Пользователи: в эту группу входят те, кто будет работать с программным обеспечением. Часто это гетерогенная группа, в которую входят люди с разными ролями и требованиями.
- Клиенты: в эту группу входят те, кто заказал программное обеспечение или представляет целевой рынок программного обеспечения.
- Аналитики рынка: продукт для массового рынка не будет иметь заказчика, поэтому часто требуются специалисты по маркетингу, чтобы определить, что нужно рынку, и действовать в качестве доверенных клиентов.
- Регуляторы. Многие области приложений, такие как банковское дело и общественный транспорт, регулируются. Программное обеспечение в этих доменах должно соответствовать требованиям регулирующих органов.
- Инженеры-программисты: эти лица имеют законный интерес в получении прибыли от разработки программного обеспечения, например, путем повторного использования компонентов в других продуктах или из них. Если в этом сценарии у клиента определенного продукта есть особые требования, которые ставят под угрозу возможность повторного использования компонентов, инженеры-программисты должны тщательно взвесить свою собственную ставку по сравнению с ставкой поставщика.

покупатель. Конкретные требования, особенно ограничения, могут иметь большое влияние на стоимость проекта или его реализацию, поскольку они либо хорошо, либо плохо соответствуют набору навыков инженеров. Следует определить важные компромиссы между такими требованиями.

Невозможно полностью удовлетворить требования всех заинтересованных сторон, и работа инженера-программиста заключается в согласовании компромиссов, приемлемых как для основных заинтересованных сторон, так и в рамках бюджетных, технических, нормативных и других ограничений. Необходимым условием для этого является выявление всех заинтересованных сторон, анализ характера их «заинтересованности» и выявление их требований.

2.3 Поддержка процесса и управление

В этом разделе представлены ресурсы управления проектом, необходимые и потребляемые процессом требований. Он устанавливает контекст для первой темы (инициация и определение объема) KA управления программной инженерией. Его основная цель состоит в том, чтобы установить связь между действиями процесса, определенными в 2.1, и вопросами затрат, человеческих ресурсов, обучения и инструментов.

2.4 Качество процесса и улучшение

Эта тема связана с оценкой качества и улучшением процесса требований. Его цель состоит в том, чтобы подчеркнуть ключевую роль, которую играет процесс требований с точки зрения стоимости и своевременности программного продукта, а также удовлетворенности потребителя им. Это поможет сориентировать процесс требований со стандартами качества и моделями улучшения процессов для программного обеспечения и систем. Качество и улучшение процесса тесно связаны как с KA качества программного обеспечения, так и с KA процесса разработки программного обеспечения, включая

- охват процесса требований стандартами и моделями улучшения процессов;
- меры по обработке требований и бенчмаркинг;
- планирование и внедрение улучшений;
- безопасность/улучшение ЦРУ/планирование и реализация

3 Выявление требований

[1 , c4c5] [2 , c5, c6, c9]

Выявление требований связано с происхождением требований к программному обеспечению и тем, как инженер-программист может их собрать. Это первый этап в построении понимания проблемы, для решения которой требуется программное обеспечение. По сути, это человеческая деятельность, и именно здесь определяются заинтересованные стороны и устанавливаются отношения между командой разработчиков и заказчиком. Его по-разному называют «сбор требований», «обнаружение требований» и «получение требований».

Одним из фундаментальных принципов хорошего процесса выявления требований является эффективная коммуникация между различными заинтересованными сторонами. Это общение продолжается на протяжении всего процесса жизненного цикла разработки программного обеспечения (SDLC) с различными заинтересованными сторонами в разные моменты времени. Перед началом разработки специалисты по требованиям могут сформировать канал для этой связи. Они должны быть посредниками между сферой деятельности пользователей программного обеспечения (и других заинтересованных сторон) и техническим миром инженера-программиста. Набор внутренне согласованных моделей на разных уровнях абстракции облегчает общение между пользователями программного обеспечения/заинтересованными сторонами и разработчиками программного обеспечения.

Важным элементом выявления требований является информирование о масштабе проекта. Это включает в себя предоставление описания программного обеспечения и его назначения, а также определение приоритетов результатов, чтобы гарантировать, что наиболее важные бизнес-потребности клиента будут удовлетворены в первую очередь. Это сводит к минимуму риск того, что специалисты по требованиям будут тратить время на выявление маловажных требований или требований, которые перестанут быть актуальными после поставки программного обеспечения. С другой стороны, описание должно быть масштабируемым и расширяемым, чтобы принимать дополнительные требования, не выраженные в первых формальных списках, и совместимые с предыдущими, предусмотренными в рекурсивных методах.

3.1 Источники требований

Требования имеют множество источников в типичном программном обеспечении, и важно, чтобы все потенциальные источники были идентифицированы и оценены. Этот раздел предназначен для повышения осведомленности о различных источниках требований к программному обеспечению и основах для управления ими. Основные моменты, о которых идет речь, следующие:

- Цели. Термин «цель» (иногда называемый «деловой интерес» или «критический фактор успеха») относится к общим высокоуровневым задачам программного обеспечения. Цели обеспечивают мотивацию для программного обеспечения, но часто формулируются расплывчато. Инженеры-программисты должны уделять особое внимание оценке ценности (относительно приоритета) и стоимости целей. Технико-экономическое обоснование является относительно недорогим способом сделать это.
- Базовые знания. Инженер-программист должен приобрести или иметь доступные знания о предметной области. Знание предметной области обеспечивает фон, на котором должны быть

установлены все выявленные знания о требованиях, чтобы понять их. Хорошей практикой является подражание онтологическому подходу в предметной области. Должны быть идентифицированы отношения между релевантными понятиями в области приложения.

- Заинтересованные стороны (см. раздел 2.2, Участники процесса). Многие программы оказались неудовлетворительными, потому что они подчеркивали требования одной группы заинтересованных сторон за счет других. Следовательно, поставленное программное обеспечение сложно использовать или оно подрывает культурные или политические структуры организации-заказчика. Инженеру-программисту необходимо выявлять, представлять и управлять «точками зрения» многих различных заинтересованных сторон.
- Бизнес правила. Это утверждения, которые определяют или ограничивают некоторые аспекты структуры или поведения самого бизнеса. «Студент не может зарегистрироваться на курсы следующего семестра, если остается неоплаченная плата за обучение» — это пример бизнес-правила, которое может быть источником требований для университетского программного обеспечения для регистрации курсов.
- Оперативная среда. Требования будут получены из среды, в которой будет выполняться программное обеспечение. Это могут быть, например, временные ограничения в программном обеспечении реального времени или ограничения производительности в бизнес-среде. Их необходимо активно искать, потому что они могут сильно повлиять на осуществимость и стоимость программного обеспечения, а также ограничить выбор дизайна.
- Организационная среда. Программное обеспечение часто требуется для поддержки бизнес-процесса, выбор которого может быть обусловлен структурой, культурой и внутренней политикой организации. Инженер-программист должен учитывать это, поскольку, как правило, новое программное обеспечение не должно приводить к незапланированным изменениям в бизнес-процессе.

3.2 Методы выявления

4 Анализ требований

[1 , c4c1, c4c5, c10c4, c12c5] [2 , c7, c11, c12, c17]

Эта тема касается процесса анализа требований к

- обнаруживать и разрешать конфликты между требованиями;
- узнать границы программного обеспечения и то, как оно должно взаимодействовать с организационной и операционной средой;
- разработать системные требования для получения требований к программному обеспечению.

Традиционный взгляд на анализ требований сводился к концептуальному моделированию с использованием одного из нескольких методов анализа, таких как метод структурного анализа. Хотя концептуальное моделирование важно, мы включаем классификацию требований, чтобы помочь определить компромиссы между требованиями (классификация требований) и процесс установления этих компромиссов (согласование требований).

Необходимо позаботиться о том, чтобы описать требования достаточно точно, чтобы можно было проверить требования, проверить их реализацию и оценить их стоимость.

4.1 Классификация требований

Требования можно классифицировать по ряду параметров. Примеры включают следующее:

- Является ли требование функциональным или нефункциональным (см. раздел 1.3, Функциональные и нефункциональные требования).
- Является ли требование производным от одного или нескольких требований высокого уровня или возникающего свойства (см. раздел 1.4, Возникающие свойства) или налагается непосредственно на программное обеспечение заинтересованной стороной или каким-либо другим источником.
- Относится ли требование к продукту или к процессу (см. раздел 1.2, Требования к продукту и процессу). Требования к процессу могут ограничивать выбор подрядчика, процесс разработки программного обеспечения, который необходимо принять, или стандарты, которых необходимо придерживаться.

- Приоритет требования. Чем выше приоритет, тем важнее требование для достижения общих целей программного обеспечения. Часто классифицируемый по шкале с фиксированной точкой, такой как обязательный, крайне желательный, желательный или необязательный, приоритет часто должен быть сбалансирован со стоимостью разработки и реализации.
- Объем требования. Объем относится к степени, в которой требование влияет на программное обеспечение и программные компоненты. Некоторые требования, особенно некоторые нефункциональные, имеют глобальную область действия, поскольку их удовлетворение не может быть отнесено к отдельному компоненту. Следовательно, требование с глобальной областью действия может сильно повлиять на архитектуру программного обеспечения и дизайн многих компонентов, в то время как требование с узкой областью действия может предложить несколько вариантов дизайна и мало повлиять на удовлетворение других требований.

4.2 Концептуальное моделирование

Разработка моделей реальной проблемы является ключом к анализу требований к программному обеспечению. Их цель — помочь понять ситуацию, в которой возникает проблема, а также показать решение. Следовательно, концептуальные модели включают в себя модели сущностей из проблемной области, сконфигурированные для отражения их отношений и зависимостей в реальном мире. Эта тема тесно связана с моделями и методами разработки программного обеспечения KA.

Можно разработать несколько видов моделей. К ним относятся диаграммы вариантов использования, модели потоков данных, модели состояний, модели на основе целей, взаимодействия с пользователем, объектные модели, модели данных и многие другие. Многие из этих нотаций моделирования являются частью унифицированного языка моделирования (UML). Диаграммы вариантов использования, например, обычно используются для описания сценариев, в которых граница отделяет действующих лиц (пользователей или системы во внешней среде) от внутреннего поведения, где каждый вариант использования отображает функциональность системы.

Факторы, влияющие на выбор нотации моделирования, включают следующее:

- Характер проблемы. Некоторые типы программного обеспечения требуют особенно тщательного анализа определенных аспектов. Например, модели состояния и параметрические модели, являющиеся частью SysML [4], скорее всего, будут более важны для программного обеспечения реального времени, чем для информационных систем, в то время как для моделей объектов и действий обычно бывает наоборот.
- Квалификация инженера-программиста. Часто более продуктивно использовать нотацию или метод моделирования, с которыми у инженера-программиста есть опыт.
- Технологические требования заказчика (см. раздел 1.2 «Требования к продукции и процессу»). Клиенты могут навязывать свои любимые обозначения или методы или запрещать любые, с которыми они не знакомы. Этот фактор может конфликтовать с предыдущим фактором.

Обратите внимание, что почти во всех случаях полезно начинать с построения модели программного контекста. Контекст программного обеспечения обеспечивает связь между предполагаемым программным обеспечением и его внешней средой.

Это крайне важно для понимания контекста программного обеспечения в среде его эксплуатации и для идентификации его интерфейсов с окружающей средой.

Эта подтема не направлена на «обучение» определенному стилю моделирования или нотации, а скорее дает руководство по цели и намерению моделирования.

4.3 Архитектурный дизайн и распределение требований

В какой-то момент архитектура решения должна быть получена. Архитектурный дизайн — это точка, в которой процесс создания требований пересекается с проектированием программного обеспечения или систем и показывает, насколько невозможно четко разделить две задачи. Эта тема тесно связана со структурой и архитектурой программного обеспечения в KA проектирования программного обеспечения. Во многих случаях инженер-программист выступает в роли архитектора программного обеспечения, потому что процесс анализа и разработки требований требует определения компонентов архитектуры/дизайна, которые будут отвечать за удовлетворение требований. Это распределение требований — назначение компонентов архитектуры, ответственных за удовлетворение требований.

Распределение важно для обеспечения детального анализа требований. Следовательно, например, после того, как компоненту был назначен набор требований, отдельные требования могут быть дополнительно проанализированы, чтобы обнаружить дополнительные требования о том, как компонент должен взаимодействовать с другими компонентами, чтобы удовлетворить назначенные требования. В больших проектах распределение стимулирует новый раунд анализа для каждой подсистемы. Например, требования к конкретным характеристикам торможения автомобиля (тормозной путь, безопасность в плохих условиях движения, плавность срабатывания, необходимое усилие на педали и т. д.) могут быть отнесены к тормозному оборудованию (механическим и гидравлическим узлам) и к тормозному оборудованию. антиблокировочная тормозная система (АБС). Только когда требование к антиблокировочной тормозной системе определено и требования к ней отнесены,

Архитектурный дизайн тесно связан с концептуальным моделированием (см. раздел 4.2, Концептуальное моделирование).

4.4 Обсуждение требований

Другой термин, обычно используемый для этой подтемы, — «разрешение конфликтов». Это касается решения проблем с требованиями, когда возникают конфликты между двумя заинтересованными сторонами, требующими взаимно несовместимых функций, между требованиями и ресурсами или, например, между функциональными и нефункциональными требованиями. В большинстве случаев инженеру-программисту неразумно принимать одностороннее решение, поэтому возникает необходимость проконсультироваться с заинтересованными сторонами, чтобы достичь консенсуса по соответствующему компромиссу. Часто по договорным причинам важно, чтобы такие решения можно было проследить до клиента. Мы классифицировали это как тему анализа требований к программному обеспечению, потому что проблемы возникают в результате анализа. Тем не менее, можно также привести веские доводы в пользу того, чтобы считать это темой проверки требований (см. тему 6, Проверка требований).

Приоритизация требований необходима не только как средство фильтрации важных требований, но и для разрешения конфликтов и планирования поэтапных поставок, что означает принятие сложных решений, требующих детального знания предметной области и хороших навыков оценки. Однако часто бывает сложно получить реальную информацию, которая может послужить основой для таких решений. Кроме того, требования часто зависят друг от друга, а приоритеты относительны. На практике разработчики программного обеспечения часто выполняют приоритизацию требований, не зная обо всех требованиях. Приоритизация требований может следовать стоимостному подходу, который включает анализ со стороны заинтересованных сторон, определяющих в масштабе выгоды или совокупную ценность, которую им приносит реализация требования. по сравнению со штрафами за несоблюдение конкретного требования. Это также включает в себя анализ со стороны разработчиков программного обеспечения, оценивающих в масштабе стоимость реализации каждого требования по сравнению с другими требованиями. Другой подход к приоритизации требований, называемый процессом аналитической иерархии, включает сравнение всех уникальных пар требований, чтобы определить, какое из них имеет более высокий приоритет и в какой степени.

4.5 Формальный анализ

Формальный анализ касается не только темы 4, но и разделов 5.3 и 6.3. Эта тема также связана с формальными методами в области знаний о моделях и методах разработки программного обеспечения.

Формальный анализ оказал влияние на некоторые области приложений, особенно на системы с высокой степенью интеграции. Формальное выражение требований требует языка с формально определенной семантикой. Использование формального анализа для выражения требований имеет два преимущества. Во-первых, он позволяет точно и недвусмысленно задавать требования, выраженные в языке, тем самым (в принципе) избегая возможности неправильного толкования. Во-вторых, требования могут быть аргументированы, что позволяет доказать желаемые свойства указанного программного обеспечения. Формальные рассуждения требуют, чтобы инструментальная поддержка была применима для чего-либо, кроме тривиальных систем, а инструменты обычно делятся на два типа: средства доказательства теорем или средства проверки моделей. Ни в том, ни в другом случае доказательство не может быть полностью автоматизировано.

Наиболее формальный анализ сосредоточен на относительно поздних стадиях анализа требований. Обычно непродуктивно применять формализацию до тех пор, пока бизнес-цели и требования пользователей не будут четко сфокусированы с помощью таких средств, как описано в другом месте в разделе 4. Однако, как только требования стабилизируются и будут разработаны для определения конкретных свойств программного обеспечения, может быть полезно формализовать хотя бы критические требования. Это позволяет проводить статическую проверку того, что программное обеспечение, указанное в требованиях, действительно обладает свойствами (например, отсутствием взаимоблокировок), которые ожидают от него заказчик, пользователи и инженер-программист.

5 Спецификация требований

[1 , c4c2, c4c3, c12c2-5] [2 , c10]

Для большинства инженерных профессий термин «спецификация» относится к назначению числовых значений или ограничений для целей проектирования продукта. В программной инженерии «спецификация требований к программному обеспечению» обычно относится к созданию документа, который можно систематически просматривать, оценивать и утверждать. Для сложных систем, особенно тех, которые содержат существенные непрограммные компоненты, создается до трех различных типов документов: определение системы, системные требования и требования к программному обеспечению. Для простых программных продуктов требуется только треть из них. Здесь описаны все три документа с пониманием того, что они могут быть объединены соответствующим образом. Описание системной инженерии можно найти в главе «Связанные дисциплины программной инженерии» данного руководства.

5.1 Документ определения системы

Этот документ (иногда называемый документом требований пользователя или документом концепции операций) содержит системные требования. Он определяет системные требования высокого уровня с точки зрения предметной области. Его читательская аудитория включает представителей пользователей/ заказчиков системы (маркетинг может играть эти роли для ориентированного на рынок программного обеспечения), поэтому его содержание должно быть сформулировано с точки зрения предметной области. В документе перечислены системные требования, а также справочная информация об общих целях системы, ее целевой среде, а также изложение ограничений, допущений и нефункциональных требований. Он может включать концептуальные модели, предназначенные для иллюстрации системного контекста, сценариев использования и основных объектов предметной области, а также рабочих процессов.

5.2 Спецификация системных требований

Разработчики систем со значительным количеством программных и непрограммных компонентов — например, современного авиалайнера — часто отделяют описание требований к системе от описания требований к программному обеспечению. В этом представлении указываются системные требования, требования к программному обеспечению выводятся из системных требований, а затем указываются требования к программным компонентам. Строго говоря, спецификация системных требований представляет собой деятельность по системному проектированию и выходит за рамки настоящего Руководства.

5.3 Спецификация требований к программному обеспечению

Спецификация требований к программному обеспечению устанавливает основу для соглашения между заказчиками и подрядчиками или поставщиками (в рыночных проектах эти роли могут играть отделы маркетинга и разработки) о том, что должен делать программный продукт, а также чего от него не ожидается. делать.

Спецификация требований к программному обеспечению позволяет провести тщательную оценку требований до начала проектирования и сократить количество последующих изменений. Он также должен обеспечить реалистичную основу для оценки стоимости продукта, рисков и графиков.

Организации также могут использовать документ спецификации требований к программному обеспечению в качестве основы для разработки эффективных планов проверки и проверки.

Спецификация требований к программному обеспечению обеспечивает обоснованную основу для передачи программного продукта новым пользователям или программным платформам. Наконец, это может стать основой для усовершенствования программного обеспечения.

Требования к программному обеспечению часто пишутся на естественном языке, но в спецификации требований к программному обеспечению это может быть дополнено формальными или полуформальными описаниями. Выбор соответствующих обозначений позволяет описать конкретные требования и аспекты архитектуры программного обеспечения более точно и лаконично, чем на естественном языке. Общее правило состоит в том, что следует использовать обозначения, позволяющие максимально точно описать требования. Это особенно важно для критически важного для безопасности, нормативного и некоторых других типов надежного программного обеспечения. Однако выбор обозначения часто ограничивается обучением, навыками и предпочтениями авторов и читателей документа.

Был разработан ряд показателей качества, которые можно использовать для соотнесения качества спецификации требований к программному обеспечению с другими переменными проекта, такими как стоимость, приемка, производительность, график и воспроизводимость. Показатели качества для отдельных заявлений спецификации требований к программному обеспечению включают императивы, директивы, слабые фразы, опции и продолжения. Показатели для всего документа спецификации требований к программному обеспечению включают размер, удобочитаемость, спецификацию, глубину и структуру текста.

6 Проверка требований

[1 , c4c6] [2 , c13, c15]

Документы с требованиями могут подвергаться процедурам валидации и проверки. Требования могут быть проверены, чтобы гарантировать, что инженер-программист понял требования; также важно убедиться, что документ с требованиями соответствует стандартам компании и является понятным, непротиворечивым и полным. В тех случаях, когда задокументированные стандарты или терминология компании несовместимы с общепринятыми стандартами, необходимо согласовать их соответствие и приложить к документу.

Формальные обозначения предлагают важное преимущество, позволяя доказать последние два свойства (по крайней мере, в ограниченном смысле). Разные заинтересованные стороны, в том числе представители заказчика и разработчика, должны просмотреть документ(ы). Документы с требованиями подчиняются тем же методам управления конфигурацией, что и другие результаты процессов жизненного цикла программного обеспечения. Когда это целесообразно, отдельные требования также подлежат управлению конфигурацией, как правило, с использованием инструмента управления требованиями (см. тему 8, Инструменты для требований к программному обеспечению).

Нормально явно запланировать одну или несколько точек в процессе требований, где требования проверяются. Цель состоит в том, чтобы выявить любые проблемы до того, как ресурсы будут выделены для удовлетворения требований. Проверка требований связана с процессом изучения документа с требованиями, чтобы убедиться, что он определяет правильное программное обеспечение (то есть программное обеспечение, которое ожидают пользователи).

6.1 Проверка требований

Возможно, наиболее распространенным средством проверки является проверка или обзор документа (ов) требований. Группе рецензентов поручается поиск ошибок, ошибочных предположений, отсутствия ясности и отклонений от стандартной практики. Важен состав группы, проводящей обзор (например, для проекта, ориентированного на клиента, должен быть включен по крайней мере один представитель заказчика), и это может помочь предоставить рекомендации о том, что искать в форме контрольных списков. .

Обзоры могут быть составлены после завершения документа определения системы, документа спецификации системы, документа спецификации требований к программному обеспечению, базовой спецификации для новой версии или на любом другом этапе процесса.

6.2 Прототип

Прототипирование обычно является средством проверки интерпретации требований к программному обеспечению инженером-программистом, а также для выявления новых требований. Как и в случае с выявлением, существует ряд методов прототипирования и ряд моментов в процессе, где проверка прототипа может быть уместной. Преимущество прототипов в том, что они могут облегчить интерпретацию предположений инженера-программиста и, при необходимости, дать полезную информацию о том, почему они неверны. Например, динамическое поведение пользовательского интерфейса лучше понять с помощью анимированного прототипа, чем с помощью текстового описания или графических моделей. Волатильность требования, которое определено после создания прототипа, чрезвычайно мала, поскольку существует соглашение между заинтересованной стороной и инженером-программистом — следовательно, для критичных с точки зрения безопасности и важных функций прототипирование действительно помогло бы. Однако есть и недостатки. К ним относится опасность отвлечения внимания пользователей от основной базовой функциональности из-за косметических проблем или проблем с качеством прототипа. По этой причине некоторые выступают за прототипы, которые избегают программного обеспечения, например макеты на основе флипчартов. Разработка прототипов может быть дорогостоящей. Однако, если они избегают растраты ресурсов, вызванной попыткой удовлетворить ошибочные требования, их стоимость может быть легче оправдана. Ранние прототипы могут содержать аспекты окончательного решения. Прототипы могут быть эволюционными, а не одноразовыми. К ним относится опасность отвлечения внимания пользователей от основной базовой функциональности из-за косметических проблем или проблем с качеством прототипа. По этой причине некоторые выступают за прототипы, которые избегают программного обеспечения, например макеты на основе флипчартов. Разработка прототипов может быть дорогостоящей. Однако, если они избегают растраты ресурсов, вызванной попыткой удовлетворить ошибочные требования, их стоимость может быть легче оправдана. Ранние прототипы могут содержать аспекты окончательного решения. Прототипы могут быть эволюционными, а не одноразовыми. К ним относится опасность отвлечения внимания пользователей от основной базовой функциональности из-за косметических проблем или проблем с качеством прототипа. По этой причине некоторые выступают за прототипы, которые избегают программного обеспечения, например макеты на основе флипчартов. Разработка прототипов может быть дорогостоящей. Однако, если они избегают растраты ресурсов, вызванной попыткой удовлетворить ошибочные требования, их стоимость может быть легче оправдана. Ранние прототипы могут содержать аспекты окончательного решения. Прототипы могут быть эволюционными, а не одноразовыми. их стоимость может быть легче оправдана. Ранние прототипы могут содержать аспекты окончательного решения. Прототипы могут быть эволюционными, а не одноразовыми. их стоимость может быть легче оправдана. Ранние прототипы могут содержать аспекты окончательного решения. Прототипы могут быть эволюционными, а не одноразовыми.

6.3 Проверка модели

Как правило, необходимо проверить качество моделей, разработанных в ходе анализа. Например, в объектных моделях полезно выполнить статический анализ, чтобы убедиться, что существуют каналы связи между объектами, которые в домене заинтересованных сторон обмениваются данными. Если используются нотации формального анализа, можно использовать формальные рассуждения для доказательства свойств спецификации. Эта тема тесно связана с моделями и методами разработки программного обеспечения КА.

6.4 Приемочные испытания

Существенным свойством требования к программному обеспечению является возможность подтверждения того, что готовый продукт ему удовлетворяет. Требования, которые не могут быть подтверждены, на самом деле являются просто «пожеланиями». Поэтому важной задачей является планирование проверки каждого требования. В большинстве случаев разработка приемочных тестов направлена на то, как конечные пользователи обычно ведут бизнес, используя систему.

Идентификация и разработка приемочных тестов могут быть затруднены для нефункциональных требований (см. раздел 1.3, Функциональные и нефункциональные требования). Чтобы быть подтвержденными, они должны быть сначала проанализированы и разложены до такой степени, чтобы их можно было выразить количественно.

Дополнительную информацию можно найти в разделе «Приемка/Квалификация/Тестирование на соответствие» в КА по тестированию программного обеспечения.

7 практических соображений

[1 , c4c1, c4c4, c4c6, c4c7] [2 , c3, c12, c14, c16, c18-21]

Может показаться, что первый уровень декомпозиции темы, представленный в этом КА, описывает линейную последовательность действий. Это упрощенный взгляд на процесс.

Процесс требований охватывает весь жизненный цикл программного обеспечения. Управление изменениями и поддержание требований в состоянии, точно отражающем создаваемое или уже созданное программное обеспечение, являются ключом к успеху процесса разработки программного обеспечения.

Не в каждой организации есть культура документирования требований и управления ими. Динамичные начинающие компании, движимые сильным «видением продукта» и ограниченными ресурсами, часто рассматривают документацию по требованиям как ненужные накладные расходы. Однако чаще всего по мере расширения этих компаний, роста их клиентской базы и развития их продукта они обнаруживают, что им необходимо восстановить требования, которые мотивировали функции продукта, чтобы оценить влияние предлагаемых изменений. Следовательно, документирование требований и управление изменениями являются ключом к успеху любого процесса требований.

7.1 Итеративный характер процесса требований

В индустрии программного обеспечения существует общее давление в пользу еще более коротких циклов разработки, и это особенно заметно в высококонкурентных, ориентированных на рынок секторах. Более того, большинство проектов в той или иной мере ограничены средой, и многие из них представляют собой обновления или версии существующего программного обеспечения, архитектура которых является заданной. Поэтому на практике почти всегда нецелесообразно реализовывать процесс требований как линейный, детерминированный процесс, в котором требования к программному обеспечению выявляются у заинтересованных сторон, определяются, распределяются и передаются группе разработчиков программного обеспечения. Это, безусловно, миф, что требования к крупным программным проектам когда-либо полностью поняты или идеально определены.

Вместо этого требования обычно повторяются до уровня качества и детализации, достаточного для принятия решений о проектировании и закупках. В некоторых проектах это может привести к тому, что требования будут определены до того, как будут полностью поняты все их свойства. Это может привести к дорогостоящей доработке, если проблемы возникнут на поздних этапах процесса разработки программного обеспечения. Однако инженеры-программисты обязательно ограничены планами управления проектами и поэтому должны предпринимать шаги для обеспечения максимально возможного «качества» требований с учетом имеющихся ресурсов. Они должны, например, сделать явными любые предположения, лежащие в основе требований, а также любые известные проблемы.

Для программных продуктов, которые разрабатываются итеративно, проектная группа может заложить в основу только те требования, которые необходимы для текущей итерации. Специалист по требованиям может продолжить разработку требований для будущих итераций, в то время как разработчики займутся проектированием и созданием текущей итерации. Такой подход обеспечивает клиентам быструю коммерческую ценность при минимальных затратах на доработку.

Почти во всех случаях понимание требований продолжает развиваться по ходу проектирования и разработки. Это часто приводит к пересмотру требований в конце жизненного цикла. Возможно, самым важным моментом в понимании требований к программному обеспечению является то, что значительная часть требований изменится. Иногда это происходит из-за ошибок в анализе, но часто является неизбежным следствием изменения «окружающей среды» — например, операционной или бизнес-среды клиента, регулирующих процессов, навязанных властями, или рынка, на котором должно продаваться программное обеспечение. . Какой бы ни была причина, важно осознавать неизбежность изменений и предпринимать шаги для смягчения их последствий. Необходимо управлять изменениями, гарантируя, что предлагаемые изменения проходят определенный процесс проверки и утверждения, а также путем тщательного отслеживания требований, анализа влияния и управления конфигурацией программного обеспечения (см. КА по управлению конфигурацией программного обеспечения). Таким образом, процесс требований — это не просто предварительная задача разработки программного обеспечения, он охватывает весь жизненный цикл программного обеспечения. В типичном проекте действия по разработке требований к программному обеспечению со временем эволюционируют от выявления до управления изменениями. Сочетание нисходящего анализа и методов проектирования с

восходящей реализацией и методами рефакторинга, которые встречаются посередине, может дать лучшее из обоих миров. Однако на практике этого добиться трудно, так как это во многом зависит от зрелости и опыта инженеров-программистов. и управление конфигурацией программного обеспечения (см. Управление конфигурацией программного обеспечения KA). Таким образом, процесс требований — это не просто предварительная задача разработки программного обеспечения, он охватывает весь жизненный цикл программного обеспечения. В типичном проекте действия по разработке требований к программному обеспечению со временем эволюционируют от выявления до управления изменениями. Сочетание нисходящего анализа и методов проектирования с восходящей реализацией и методами рефакторинга, которые встречаются посередине, может дать лучшее из обоих миров. Однако на практике этого добиться трудно, так как это во многом зависит от зрелости и опыта инженеров-программистов. и управление конфигурацией программного обеспечения (см. Управление конфигурацией программного обеспечения KA). Таким образом, процесс требований — это не просто предварительная задача разработки программного обеспечения, он охватывает весь жизненный цикл программного обеспечения. В типичном проекте действия по разработке требований к программному обеспечению со временем эволюционируют от выявления до управления изменениями. Сочетание нисходящего анализа и методов проектирования с восходящей реализацией и методами рефакторинга, которые встречаются посередине, может дать лучшее из обоих миров. Однако на практике этого добиться трудно, так как это во многом зависит от зрелости и опыта инженеров-программистов. действия по требованиям к программному обеспечению со временем эволюционируют от выявления до управления изменениями. Сочетание нисходящего анализа и методов проектирования с восходящей реализацией и методами рефакторинга, которые встречаются посередине, может дать лучшее из обоих миров. Однако на практике этого добиться трудно, так как это во многом зависит от зрелости и опыта инженеров-программистов. действия по требованиям к программному обеспечению со временем эволюционируют от выявления до управления изменениями. Сочетание нисходящего анализа и методов проектирования с восходящей реализацией и методами рефакторинга, которые встречаются посередине, может дать лучшее из обоих миров. Однако на практике этого добиться трудно, так как это во многом зависит от зрелости и опыта инженеров-программистов.

7.2 Управление изменениями

Управление изменениями занимает центральное место в управлении требованиями. В этом разделе описывается роль управления изменениями, необходимые процедуры и анализ, который следует применять к предлагаемым изменениям. Он тесно связан с KA управления конфигурацией программного обеспечения.

7.3 Атрибуты требований

Требования должны состоять не только из спецификации того, что требуется, но и из вспомогательной информации, которая помогает управлять требованиями и интерпретировать их. Атрибуты требований должны быть определены, записаны и обновлены по мере развития разрабатываемого или поддерживаемого программного обеспечения. Это должно включать различные параметры классификации требования (см. раздел 4.1, Классификация требований) и метод проверки или соответствующий раздел плана приемочных испытаний. Он также может включать дополнительную информацию, такую как краткое обоснование каждого требования, источник каждого требования и историю изменений. Однако наиболее важным атрибутом требований является идентификатор, который позволяет однозначно и однозначно идентифицировать требования.

7.4 Отслеживание требований

Отслеживание требований связано с восстановлением источника требований и прогнозированием последствий требований. Отслеживание имеет основополагающее значение для выполнения анализа воздействия при изменении требований. Требование должно прослеживаться в обратном направлении до требований и заинтересованных сторон, которые его мотивировали (например, от требования к программному обеспечению до требований к системе, которые оно помогает удовлетворить). И наоборот, требование должно прослеживаться до требований и элементов дизайна, которые его удовлетворяют (например, от системного требования до требований к программному обеспечению, которые были разработаны на его основе, и далее до модулей кода, которые его реализуют, или тестовых примеров). связанный с этим кодом и даже с данным разделом руководства пользователя, в котором описывается реальная функциональность) и в тестовый пример, который его проверяет.

Отслеживание требований для типичного проекта будет формировать сложный ориентированный ациклический граф (DAG) (см. Графы в Computing Foundations KA) требований. Поддержание актуального графика или матрицы прослеживаемости — это деятельность, которую необходимо учитывать в течение всего жизненного цикла продукта. Если информация о прослеживаемости не обновляется по мере того, как продолжают происходить изменения в требованиях, информация о прослеживаемости становится ненадежной для анализа влияния.

7.5 Требования к измерениям

На практике обычно бывает полезно иметь некоторое представление об «объеме» требований к конкретному программному продукту. Это число полезно при оценке «размера» изменения требований, при оценке стоимости задачи разработки или обслуживания или просто для использования в качестве знаменателя в других измерениях. Измерение функционального размера (FSM) — это метод оценки размера совокупности функциональных требований.

Дополнительную информацию об измерении размера и стандартах можно найти в KA процесса разработки программного обеспечения.

8 инструментов требований к программному обеспечению

Инструменты для работы с требованиями к программному обеспечению можно разделить на две категории: инструменты для моделирования и инструменты для управления требованиями.

Инструменты управления требованиями обычно поддерживают ряд действий, включая документирование, отслеживание и управление изменениями, и оказали значительное влияние на практику. Действительно, отслеживание и управление изменениями реально осуществимы только в том случае, если они поддерживаются инструментом. Поскольку управление требованиями является основой хорошей практики требований, многие организации вложили средства в инструменты управления требованиями, хотя многие другие управляют своими требованиями более ситуативными и, как правило, менее удовлетворительными способами (например, с помощью электронных таблиц).

ДАЛЬНЕЙШИЕ ЧТЕНИЯ

И. Александер и Л. Беус-Дукич, *Обнаружение требований* [5].

Легко усваиваемая и практически ориентированная книга о требованиях к программному обеспечению, пожалуй, лучший из современных учебников о том, как различные элементы требований к программному обеспечению сочетаются друг с другом. Он полон практических советов, например, о том, как определить различные заинтересованные стороны системы и как оценить альтернативные решения. Его охват является образцовым и служит полезным справочником по ключевым методам, таким как моделирование вариантов использования и приоритизация требований.

К. Поттс, К. Такахаши и А. Антон, «Анализ требований на основе запросов» [6].

Этот документ представляет собой легко усваиваемый отчет о работе, оказавшей большое влияние на развитие обработки требований. Он описывает, как и почему разработка требований не может быть линейным процессом, посредством которого аналитик просто расшифровывает и переформулирует требования, полученные от заказчика. Роль сценариев описана таким образом, чтобы помочь определить их использование при обнаружении и описании требований.

А. ван Ламсвеерде, Разработка *требований* : от целей системы к моделям UML и спецификациям программного обеспечения *[7]*.

Служит хорошим введением в разработку требований, но его уникальная ценность заключается в том, что он является справочником по языку моделирования требований KAOS, ориентированному на цели. Объясняет, чем полезно моделирование целей, и показывает, как его можно интегрировать с основными методами моделирования с использованием UML.

О. Готел и А. Финкельштейн, «Анализ проблемы прослеживаемости требований» [8].

Эта статья представляет собой классический справочник по ключевому элементу управления требованиями. На основе эмпирических исследований в нем излагаются причины и препятствия на пути эффективного отслеживания требований. Это необходимо прочитать, чтобы понять, почему отслеживание требований является важным элементом эффективного процесса разработки программного обеспечения.

Мейден Н., Нкубе К., «Получение требований к выбору программного обеспечения COTS» [9].

Этот документ имеет важное значение, поскольку в нем прямо признается, что программные продукты часто интегрируют сторонние компоненты. Он предлагает понимание проблем выбора готового программного обеспечения для удовлетворения требований: обычно бывает несоответствие. Это бросает вызов некоторым предположениям, лежащим в основе большей части традиционной обработки требований, которая, как правило, предполагает использование заказного программного обеспечения.

ИСПОЛЬЗОВАННАЯ ЛИТЕРАТУРА

[1] И. Соммервиль, *Разработка программного обеспечения* , 9-е изд., Addison-Wesley, 2011.

[2] К.Е. Вигерс, *Требования к программному обеспечению* , 2- е изд., Microsoft Press, 2003.

[3] INCOSE, *Справочник по системной инженерии: Руководство по процессам и действиям жизненного цикла системы* , версия 3.2.2, Международный совет по системной инженерии, 2012 г.

[4] С. Фриденталь, А. Морре и Р. Штайнер, *Практическое руководство по SysML: язык моделирования систем* , 2-е изд., Морган Кауфманн, 2012.

[5] И. Александер и Л. Беус-Дуекич, *Обнаружение требований: как определить продукты и услуги* , Wiley, 2009.

[6] К. Поттс, К. Такахаши и А.И. Антон, «Анализ требований на основе запросов», *Программное обеспечение IEEE* , Международный совет по системной инженерии, том. 11, нет. 2, март 1994 г., стр. 21-32.

[7] А. ван Ламсвеерде, Разработка *требований: от целей системы к моделям UML и спецификациям программного обеспечения* , Wiley, 2009.

[8] О. Готель и К. В. Финкельштейн, «Анализ проблемы прослеживаемости требований», *Proc 1st Int'l Conf. Требования инж.* , ИИЭР, 1994.

[9] Н.А. Мейден и К. Нкубе, «Принятие требований к выбору программного обеспечения COTS», *IEEE Software* , vol. 15, нет. 2, март-апр. 1998, стр. 46-56.

- Последнее изменение этой страницы: 20 августа 2015 г., 20:33.