



МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технологический университет «СТАНКИН»
(ФГБОУ ВО МГТУ «СТАНКИН»)

Кафедра "Информационные технологии и вычислительные системы"

Новоселова Ольга Вячеславовна

Современные технологии и средства разработки программного обеспечения

Курс лекций
по дисциплине «Современные технологии и средства разработки программного обеспечения»

для студентов МГТУ «СТАНКИН», обучающихся

по направлению: 09.03.01 "Информатика и вычислительная техника" ,

по профилю: "Программное обеспечение средств вычислительной техники и автоматизированных систем"

Москва 2020г.

Оглавление.
7 семестр

Раздел 3. Современные технологии формирования требований к программному продукту.

Лекции 7-9. 3

Тема 8. Спецификация и классификация требований к программному продукту. Управление требованиями к программному продукту.

Тема 9. Переход от моделей бизнес-процессов к спецификации требований к программному продукту. Описание требований к программному продукту с помощью сценариев его использования.

Контрольные вопросы 14

Список литературы 14

7 семестр.

Лекции 7-9.

Тема 8. Спецификация и классификация требований к программному продукту. Управление требованиями к программному продукту.

Тема 9. Переход от моделей бизнес-процессов к спецификации требований к программному продукту. Описание требований к программному продукту с помощью сценариев его использования.

План лекции.

1. Последовательность работы с требованиями.
2. Классификация требований
3. Спецификация требований
4. Модель автоматизируемой предметной задачи (и ее бизнес-процессов)
5. Переход от моделей к спецификации требований программного обеспечения
6. UML: диаграмма вариантов использования (прецедентов)

Основная часть.

Последовательность работы с требованиями.

У пользователя есть технические или бизнес-задачи, для решения которых им нужны программисты. Задача последних состоит в том, чтобы понять проблемы пользователей в их собственной проблемной плоскости и на их языке и построить системы, удовлетворяющие их требованиям. Для понимания проблемы пользователей существует ряд профессиональных приемов.

Программисты должны понять потребности пользователей и других заинтересованных лиц, на чью жизнь повлияет создание программы.

Следующим шагом осуществляется переход в область решения, но до программирования необходимо сформулировать знания о предметной области. На данном этапе составляется список функций, которые должна реализовать система.

Для того чтобы провести анализ, полезно определить, что же собственно представляет собой проблема. По определению Гауса и Вайнберга, проблема – это разница между желаемым и воспринимаемым.

Иногда самым простым решением является изменение бизнес-процесса, а не создание новой системы. Поэтому начинать следует с определения цели. Цель анализа состоит в том, чтобы добиться лучшего понимания решаемой проблемы до начала разработки. Для этого следует пройти 5 этапов:

1. Достигнуть соглашения об определении проблемы.
2. Выделить основные причины - вопросы, стоящие за проблемой.
3. Выявить заинтересованных лиц и пользователей.
4. Определить границу системы решения.
5. Выявить ограничения, которые необходимо наложить на решение.

Классификация требований.

В IEEE Standard Glossary of Software Engineering Terminology (1990) понятие требование определяется следующим образом:

«Требование – это:

- условия или возможности, необходимые пользователю для решения проблем или для достижения целей;
- условия или возможности, которыми должна обладать система или системные компоненты, чтобы выполнить контракт или удовлетворять стандартам, спецификациям или другим формальным документам;
- документирование представления условий или возможностей для пунктов 1 и 2.»

Более кратко можно сказать, что требование – это условие или возможность, которой должна соответствовать система. Требования являются исходными данными, на основании которых проектируются и создаются автоматизированные системы.

При этом первичные данные поступают, как правило, из разных источников, характеризуются неполнотой, противоречивостью, изменчивостью и нечеткостью описания. Но они необходимы для того, чтобы разработчики поняли, что необходимо создать, а также чтобы они могли согласовать с заказчиком время выполнения проекта и финансовые затраты.

В соответствии с этим основная часть требований собирается на начальном этапе проекта, но на практике процесс сбора требований растягивается на весь процесс жизненного цикла ПО или автоматизированной системы.

Список требований к разрабатываемой системе должен включать:

- совокупность условий, при которых предполагается эксплуатировать будущую систему (аппаратные и программные ресурсы, предоставляемые системе; внешние условия ее функционирования; состав людей и работ, имеющих к ней отношение);
- описание выполняемых системой функций;
- ограничения в процессе разработки (директивные сроки завершения отдельных этапов, имеющиеся ресурсы, организационные процедуры и мероприятия, обеспечивающие защиту информации).

Целью анализа является преобразование общих, неясных знаний о требованиях к будущей системе в точные (по возможности) определения. На этом этапе определяются:

- архитектура системы, ее функции, внешние условия, распределение функций между аппаратурой и ПО;
- интерфейсы и распределение функций между человеком и системой;
- требования к программным и информационным компонентам ПО, необходимые аппаратные ресурсы, требования к БД, физические характеристики компонент ПО, их интерфейсы.

Классификация требований:

1. Требования к продукту.

В своей основе требования – это то, что формулирует заказчик. Его цель – получить хороший программный продукт – функциональный и удобный в использовании. Это основной класс требований. Включают функциональные и нефункциональные требования к программному продукту.

2. Требования к проекту.

Включают регламентацию процесса создания программного продукта и его аудит со стороны заказчика для снижения рисков проекта (невыполнения в срок, финансовые издержки и т.п.).

Уровни требований

Уровни требований связаны с уровнем абстракции системы и с уровнем управления на предприятии или в организации. Выделяют три уровня требований:

- бизнес-требования (верхний уровень). Обычно формируются топ-менеджерами либо акционерами предприятия. (что должна делать система в целом – ее задача).
- уровень требований пользователя. Часто плохо структурированы и противоречивы. Поэтому необходим третий уровень. (какие средства она должна предоставлять).

- функциональный. Функции, которые должна предоставлять ПО или автоматизированной системы на предприятии являются следующие – одна точка сбора, данные собираются там, где они появляются. Это приводит к необходимости при внедрении ПО или автоматизированной системы к оснащению всех точек ввода информации автоматизированными рабочими местами.

3. Системные требования.

В практике под системными требованиями в узком смысле понимаются требования, выдвигаемые прикладной программной системой к среде своего функционирования. Пример таких требований – вид операционной системы, объем памяти, тактовая частота процессора.

Требования к продукту.

Функциональные требования – регламентируют функционирование или поведение системы. Отвечают на вопрос «что должна делать система» в тех или иных ситуациях. Функциональные требования определяют основной «фронт работ» разработчика, устанавливают цели, задачи, сервисы, предоставляемые системой заказчику. Функциональные требования описывают сервисы, предоставляемые программной системой, ее поведение в определенных ситуациях, реакцию на те или иные входные данные и действия, которые система позволит выполнить пользователям. Иногда сюда добавляют сведения о том, чего система делать не должна.

Функциональные требования записываются, как правило, при посредстве предписывающих правил: «система должна выполнять _____». Другим способом являются варианты использования – способ представления требований.

Функциональные требования документируются в спецификациях требований к программному обеспечению, где описывается как можно более полно ожидаемое поведение системы. Спецификация должна базироваться на четких понятиях и утверждениях, однозначно понимаемых разработчиками и заказчиками программного продукта.

Функциональная спецификация состоит из трех частей:

1. Описание внешней информационной среды, с которой будет взаимодействовать разрабатываемое программное обеспечение. Должны быть определены все используемые каналы ввода и вывода и все информационные объекты, к которым будет применяться разрабатываемое программное обеспечение, а также существенные связи между этими информационными объектами.
2. Определение функций программного обеспечения, определенных на множестве состояний этой информационной среды. Вводятся обозначения всех определяемых функций, специфицируются их входные данные и результаты выполнения, с указанием типов данных и заданий всех ограничений, которым должны удовлетворять эти данные и результаты. Определяется содержание каждой из этих функций.
3. Описание исключительных ситуаций, если такие могут возникнуть при выполнении программ и реакций на эти ситуации, которые должны обеспечить соответствующие программы. Должны быть перечислены все существенные случаи, когда программное обеспечение не сможет нормально выполнить какую-либо функцию. Для каждого такого случая должна быть определена реакция программы.

Нефункциональные требования регламентируют внутренние и внешние условия или атрибуты функционирования системы. Выделяют следующие группы (по К. Вигерсу):

- внешние интерфейсы;
- атрибуты качества;
- ограничения.

Среди внешних интерфейсов в большинстве программных продуктов наиболее важным является интерфейс пользователя. Кроме того, выделяют интерфейсы с внешними устройствами (аппаратные интерфейсы), программные интерфейсы и интерфейсы передачи информации (коммуникационные интерфейсы).

Ограничения – формулировки условий, модифицирующих требования или наборы требований, сужая выбор возможных решений по их реализации, выбор платформы реализации и/или развертывания (протоколы, серверы приложений, баз данных и т.п.), которые, в свою очередь, могут относиться к внешним интерфейсам.

Основные атрибуты качества, которые должны быть определены:

- применимость;
- надежность;
- производительность;
- эксплуатационная пригодность.

В настоящее время используется модель FURSP: функциональность, применимость, надежность, производительность, эксплуатационная пригодность.

Для модели FURSP+ добавляются:

- ограничения проекта,
- требования выполнения,
- требования к интерфейсу,
- физические требования.

Кроме того, могут выделять:

- требования, указывающие на необходимость согласованности с некоторыми юридическими и нормативными актами;
- требования к лицензированию;
- требования к документированию.

Спецификация требований.

Спецификация требований программного обеспечения ([англ. Software Requirements Specification, SRS](#)) — законченное описание поведения программы, которую требуется разработать.

Включает ряд пользовательских сценариев ([англ. use cases](#)), которые описывают все варианты взаимодействия между пользователями и программным обеспечением.

Пользовательские сценарии являются средством представления функциональных требований. В дополнение к пользовательским сценариям, спецификация также содержит нефункциональные требования, которые налагают ограничения на дизайн или реализацию (такие как требования производительности, стандарты качества, или проектные ограничения).

В стандарте IEEE 830 содержится рекомендации к структуре и методам описания программных требований — «Recommended Practice for Software Requirements Specifications».

Рекомендуемая стандартом IEEE 830 структура SRS

- Введение
- Цели
- Соглашения о терминах
- Предполагаемая аудитория и последовательность восприятия
- Масштаб проекта
- Ссылки на источники
- Общее описание
- Видение продукта
- Функциональность продукта
- Классы и характеристики пользователей
- Среда функционирования продукта (операционная среда)

- Рамки, ограничения, правила и стандарты
- Документация для пользователей
- Допущения и зависимости
- Функциональность системы
- Функциональный блок X (таких блоков может быть несколько)
- Описание и приоритет
- Причинно-следственные связи, алгоритмы (движение процессов, workflows)
- Функциональные требования
- Требования к внешним интерфейсам
- Интерфейсы пользователя (UX)
- Программные интерфейсы
- Интерфейсы оборудования
- Интерфейсы связи и коммуникации
- Нефункциональные требования
- Требования к производительности
- Требования к сохранности (данных)
- Критерии качества программного обеспечения
- Требования к безопасности системы
- Прочие требования
- Приложение А: Глоссарий
- Приложение Б: Модели процессов и предметной области и другие диаграммы
- Приложение В: Список ключевых задач

На стадии определения требований рекомендуется использовать следующие основные артефакты:

- 1.Общая формулировка задачи
- 2.Потребители системы
- 3.Цели
- 4.Функции системы
- 5.Атрибуты системы

Соединение составных частей

Для рассматриваемой в качестве примера системы определение требований выполняется лишь схематично – в реальной системе требования должны быть более полными. Этот этап обычно включает подготовку многочисленных жестких и электронных копий документов, обработку результатов интервьюирования, заседания группы разработчиков требований и другие виды деятельности.

1. Общая формулировка задачи

Цель проекта –

2.Потребители системы

3.Цели

Основная цель – повышение уровня автоматизации для обеспечения более быстрого, эффективного и дешевого выполнения

4.Функции системы

Функции системы (system functions) – это ее основное назначение. Они должны быть определены и систематизированы в логически связанные группы.

Если X действительно является функцией системы, то имеет смысл следующее предположение:

Система должна выполнять X.

Категории функций

Функции системы, должны быть систематизированы с учетом их приоритета.

Существуют следующие категории функций.

Категория функций	Значение
Очевидные	Их выполнение очевидно для пользователя
Скрытые	Должны выполняться незаметно для пользователя. Это касается многих базовых технических функций, таких как сохранение информации на постоянном носителе. Скрытые функции зачастую необоснованно упускаются в процессе определения требований к системе.
Дополнительные	Необязательные функции, добавление которых не приведет к существенному удорожанию проекта и не повлияет на выполнение остальных функций.

Основные функции

п/п	Функция	Категория
		Очевидная
		Скрытая

5. Атрибуты системы – это нефункциональные качества системы, например, простота в использовании, стиль интерфейса, отказоустойчивость, стоимость, время отклика, платформы. Очень часто атрибуты путают с функциями.

(Примечание: Надо обратить внимание, что атрибут «простота в использовании» не может быть поставлен в тестовое предложение: «Система должна выполнять простоту в использовании».)

Атрибуты системы не являются частью функциональной спецификации, а определяются в отдельном документе (спецификации атрибутов).

Атрибуты системы могут относиться ко всем функциям одновременно, как, например, аппаратная платформа, либо к одной или нескольким функциям.

Атрибуты системы характеризуются допустимым набором значений и могут принимать дискретные, нечеткие или смысловые значения, например:

Время отклика = (психологически приемлемое),

Стиль интерфейса = (графический, цветной, основанный на формах).

Некоторые атрибуты могут иметь ограничения или граничные условия, задаваемые, как правило, в виде числового диапазона значений атрибута, например:

Время отклика = (не более 5 с).

Атрибут	Значение или ограничение

Атрибуты системы в функциональной спецификации

Очень удобно описывать атрибуты системы, явно связанные с функциями в функциональной спецификации. Кроме того, значения и ограничения атрибутов обычно записываются в повелительном, а не сослагательном наклонении. (Ограничения всегда формулируются в повелительном наклонении, поскольку они являются жесткими требованиями.)

№ п/п	функция	категория	атрибут	значение и ограничения	категория

Другие артефакты стадии формулировки требований

Выше были изложены лишь краткие сведения о требованиях к системе. Описание системных функций и атрибутов – это лишь минимальный набор документов, создаваемых на этапе определения требований к системе. Однако для более глубокой проработки вопроса и снижения риска разработки необходимо учитывать и другие важные артефакты.

- Группы разработчиков требований – перечень участников, вовлеченных в процесс формулировки требований, определения функций и атрибутов системы, выполнения обзоров и проведения интервьюирования,
- Группы участников разработки – перечень групп, принимающих участие в разработке или развертывании системы,
- Предположения – сделанные допущения,
- Риски – факторы, которые могут повлиять на успешную реализацию проекта,
- Зависимости – компании, системы или продукты, от которых зависит выполнение проекта,
- Словарь терминов – определение основных терминов,
- Прецеденты – описания процессов, происходящих в предметной области,
- Приблизительная концептуальная модель – модель основных понятий и их взаимосвязей.

Модель автоматизируемой предметной задачи (и ее бизнес-процессов). Переход от моделей к спецификации требований.

В модели предметной задачи формируются информационная составляющая, функциональная составляющая.

Каждая из перечисленных составляющих описывается с помощью графической нотации, при этом процесс решения задачи отражает функции, которые должна выполнять программная система, информационная часть модели определяет перечень параметров и структуру информации в системе.

Для графических моделей требований исторически использовались диаграммы или методологии графического моделирования: ER (IDEF1FX), IDEF0, IDEF3, DFD, UML, ARIS (eEPC, VAD, eERM) и другие.

UML: диаграмма вариантов использования (прецедентов).

Базовые понятия UML

Процесс ICONIX состоит из двух частей:

- динамическая модель (включающая диаграммы прецедентов, пригодности и последовательности действий) - описывает поведение системы;
- статическая модель (включающая модель предметной области и диаграмму классов) - описывает структуру системы.

Диаграммы прецедентов: базовые понятия

Диаграмма прецедентов - диаграмма, описывающая функциональное назначение системы или то, что система будет делать в процессе своего функционирования; стрелки указывают на то, какие исполнители заняты в каких прецедентах. Эта диаграмма является исходным концептуальным представлением или концептуальной моделью системы в процессе ее проектирования и разработки.

Разработка диаграммы прецедентов преследует следующие цели:

1. определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы;
2. сформулировать общие требования к функциональному поведению проектируемой системы;
3. разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей;
4. подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями.

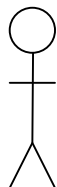
Суть диаграммы состоит в следующем: проектируемая система представляется в форме так называемых прецедентов, с которыми взаимодействуют некоторые внешние сущности или актеры, или исполнители. При этом актером или действующим лицом называется любой объект, субъект или система, взаимодействующая с моделируемой системой извне. Это может быть человек, техническое устройство, программа или любая другая система, которая может служить источником воздействия на моделируемую систему так, как определит сам разработчик. В свою очередь прецедент служит для описания сервисов, которые система предоставляет актеру. Другими словами, каждый прецедент определяет некоторый набор действий, совершаемый системой при диалоге с актером. При этом ничего не говорится о том, каким образом будет реализовано взаимодействие актеров с системой и собственно выполнение прецедентов.

Исполнители и прецеденты

Исполнитель (актер) - роль, которую пользователь играет по отношению к системе, или сущность (например, другая система или база данных) за пределами системы.

Актер представляет собой любую внешнюю по отношению к моделируемой системе сущность, которая взаимодействует с системой и использует ее функциональные возможности для достижения определенных целей или решения частных задач. При этом актеры служат для обозначения согласованного множества ролей, которые могут играть пользователи в процессе взаимодействия с проектируемой системой. Каждый актер может рассматриваться как некоторая отдельная роль относительно конкретного прецедента.

Имена актеров должны начинаться с заглавной буквы и следовать рекомендациям использования имен для типов и классов модели.



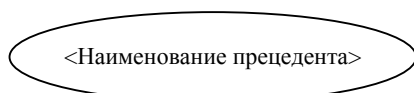
Имя исполнителя (актера)

Прецедент - набор действий, совершаемых исполнителем в системе, для достижения определенной цели.

Прецедент представляет собой спецификацию общих особенностей поведения или функционирования моделируемой системы без рассмотрения внутренней структуры этой

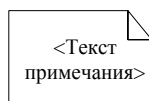
системы. Хотя каждый прецедент определяет некоторую последовательность действий, которые должны быть выполнены проектируемой системой при взаимодействии ее с соответствующим актером, сами эти действия не изображаются на рассматриваемой диаграмме. Содержание прецедента может быть представлено в форме дополнительного пояснительного текста (- сценария), который раскрывает смысл или семантику выполняемых действий при выполнении данного прецедента. Текст должен содержать фразу в глагольной форме.

Каждый прецедент соответствует отдельному сервису, который предоставляет моделируемая система по запросу актера, то есть определяет один из способов применения системы. Диаграмма прецедентов должна содержать конечное множество прецедентов, которые в целом определяют все возможные стороны ожидаемого поведения системы.



Примечания

Примечание – предназначено для включения в модель произвольной текстовой информации, имеющей непосредственное отношение к контексту разрабатываемого проекта (комментарии разработчика, ограничения, помеченные значения, уточняющая информация для прецедентов).



Потоки событий или текстовые сценарии

Для того, чтобы учесть на диаграммах прецедентов особенности функционального поведения достаточно сложной системы рекомендуется дополнять этот тип диаграмм текстовыми сценариями, которые уточняют или детализируют последовательность действий, совершаемых системой при выполнении ее прецедентов.

Потоки событий - варианты развития событий в рамках данного прецедента.

Основной поток событий (главная последовательность) - главный, самый благоприятный вариант развития событий от начала до конца, чему исполнитель и система будут следовать в нормальных условиях.

Исключительный поток событий (альтернативные последовательности) - ход развития прецедента, соответствующий ошибочному состоянию, или маршрут, который исполнитель или система выбирают с наименьшей вероятностью.

Шаблон для написания сценария отдельного варианта использования

Главный раздел	Раздел «Типичный ход событий»	Раздел «исключения»	Раздел «Примечания»
Имя прецедента	Типичный ход событий, приводящий к успешному выполнению прецедента	Исключение 1	Примечания
Актеры		Исключение 2	
Цель			
Краткое описание		Исключение 3	
Тип			
Ссылки на другие прецеденты			

Раздел «Типичный ход событий»

Типичный ход событий	
Действия актеров	Отклик системы

Раздел «Исключения»

Действия актеров	Отклик системы
------------------	----------------

Исключение1	

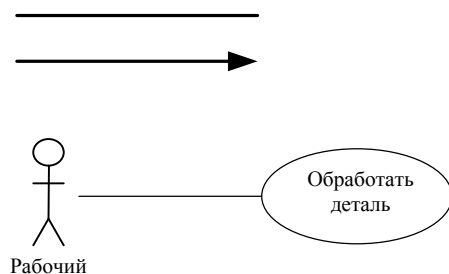
Организация прецедентов

Стандартные виды отношений между актерами и прецедентами, которые описывают их взаимодействие.

Конструкции UML для выделения общего поведения и вариантов развития событий, упрощающие прецеденты:

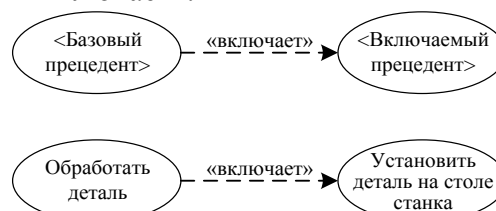
1. отношение ассоциации – служит для обозначения специфической роли актера при взаимодействии с отдельным прецедентом. Отношение ассоциации может иметь собственное имя, а концевые точки – кратность.

В контексте данной диаграммы отношение ассоциации между актером и прецедентом может указывать на тот факт, что актер является инициатором соответствующего прецедента. В этом случае актера называют главным актером. В других случаях подобная ассоциация может указывать на актера, которому предоставляется справочная информация о результатах функционирования моделируемой системы. Таких актеров называют второстепенными.



2. отношение включения – устанавливается только между двумя прецедентами и указывает, что один прецедент явно включает в себя ход действий из другого прецедента. При этом выполнение включаемой последовательности действий происходит всегда при иницировании базового прецедента.

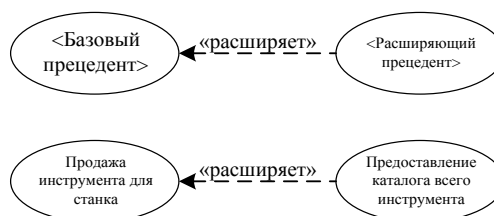
Стрелка направляется от базового прецедента к включаемому, при этом линия со стрелкой помечается стереотипом «включает».



3 отношение расширения – определяет взаимосвязь базового прецедента с некоторым другим прецедентом, функциональное поведение которого задействуется базовым не всегда, а только при выполнении некоторых дополнительных условий. В пределах этого отношения один прецедент неявно включает в себя ход действий из другого прецедента; такая конструкция обычно используется для описания поведения при выполнении некоторого условия.

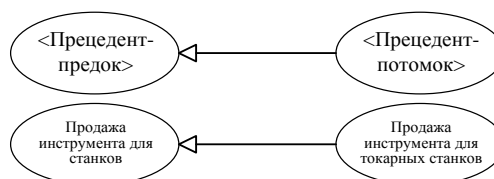
Стрелка направляется от базового прецедента к включаемому, при этом линия со стрелкой помечается стереотипом «расширяет».

Данное отношение всегда предполагает проверку некоторого условия и ссылку на точку расширения в базовом прецеденте. Точка расширения определяет место, в которое должно быть помещено расширение при выполнении соответствующего логического условия (точки расширений могут определяться с помощью текста примечаний, пред- и постусловий).



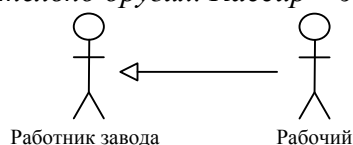
4. отношение обобщения - родительский прецедент определяет поведение, наследуемое потомками; потомки могут предопределить или расширить наследованное поведение. Некоторый прецедент (потомок) является специальным случаем другого прецедента (предок).

Потомок наследует все свойства поведения своего предка, а также может обладать дополнительными особенностями поведения.



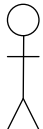
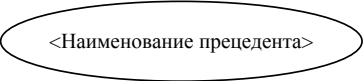
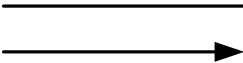
2, 3, 4 отношения – только между прецедентами.

Между двумя актерами также может существовать отношение обобщения. Данное отношение является направленным и указывает на факт специализации одних актеров относительно других. Кассир – это служащий банка.



Элементы, связи, ограничения, манипуляции UML

Элементы и связи между элементами UML для диаграммы вариантов использования.

Обозначение элемента	Название элемента	Что отражает
 Имя исполнителя (актера)	исполнитель	пользователь
	прецедент (вариант использования)	процесс
	связь между исполнителями и прецедентами	участие исполнителей в прецедентах; связь указывает на то, какие исполнители заняты в каких прецедентах

Метод решения ПЗ: Моделирование прецедентов

Работа системы зависит от того, как к ней обращаются и чего хотят добиться. Часто эти вопросы связаны с графическим интерфейсом пользователя, и для определения прецедентов нужно использовать его прототип. Кроме того, работа над моделью прецедентов происходит параллельно с моделированием предметной области на самых ранних этапах

проекта. Вся динамическая часть объектной модели непосредственно вытекает из модели прецедентов. Поскольку динамическая модель определяет статическую, модель прецедентов оказывает решающее воздействие на последнюю.

Чтобы решить задачу построения прецедентов для новой системы, необходимо с самого начала идентифицировать как можно больше прецедентов, а затем составить и постоянно уточнять их словесное описание. При выявлении прецедентов необходимо всегда помнить главный принцип: проект системы базируется на восприятии ее пользователями. Цель состоит в том, чтобы учесть все, что может сделать пользователь. Каждое предложение должно иметь структуру «существительное - глагол – существительное». По мере возникновения новых объектов или уточнения поведения ранее найденных модель предметной области обновляется. Очень важно иметь в виду все альтернативные последовательности действий для каждого прецедента.

Цели моделирования прецедентов:

- созданные прецеденты описывают всю требуемую функциональность системы;
- для каждого прецедента четко и кратко описана главная последовательность

действий, а

также все альтернативные последовательности;

- выделены сценарии, общие для нескольких прецедентов.

Ограничения и манипуляции

Диаграммы прецедентов

Прецедент должен описывать вариант использования системы без ориентации на определенное проектное решение или реализацию.

Исполнитель, инициирующий определенный прецедент, обычно изображается на диаграмме слева. Прецеденты помещаются в центре. Все остальные исполнители, задействованные в данных прецедентах, изображаются в правой части диаграммы.

Контрольные вопросы

1. Как классифицируются требования к программному продукту?
2. Как должно осуществляться управление требованиями к программному продукту?
3. Как влияет на требования к программному продукту наличие у Заказчика (Потребителя) интегрированной информационной среды, в которую должен быть интегрирован программный продукт?
4. Что такое вариант использования программного продукта, для чего он нужен и как связан с задачами, для решения которых создаётся (выбирается) программный продукт?
5. Как и на каком основании осуществляется ранжирование вариантов использования программного продукта?
6. Как реализуются отношения расширения и включения в вариантах использования программного продукта?
7. Как осуществляется анализ вариантов использования программного продукта при выработке функциональных требований к нему?
8. Какие виды моделей предусмотрены в языке моделирования UML, и с чем это связано?
9. Как осуществляется моделирование вариантов использования программного продукта с помощью языка моделирования UML?
10. Каковы сильные и слабые стороны языка моделирования UML при моделировании вариантов использования программного продукта?

Список литературы

1.Зубкова, Т.М. Технология разработки программного обеспечения: учебное пособие / Т.М. Зубкова; Министерство образования и науки Российской Федерации, Федеральное

государственное бюджетное образовательное учреждение высшего образования «Оренбургский государственный университет», Кафедра программного обеспечения вычислительной техники и автоматизированных систем. - Оренбург: ОГУ, 2017. - 469 с.: ил. - Библиогр.: с. 454-459. - ISBN 978-5-7410-1785-2; То же [Электронный ресурс]. - URL: <http://biblioclub.ru/index.php?page=book&id=485553>.

3. Антамошкин, О.А. Программная инженерия. Теория и практика : учебник / О.А. Антамошкин ; Министерство образования и науки Российской Федерации, Сибирский Федеральный университет. - Красноярск : Сибирский федеральный университет, 2012. - 247 с. : ил., табл., схем. - Библиогр.: с. 240. - ISBN 978-5-7638-2511-4 ; То же [Электронный ресурс]. - URL: <http://biblioclub.ru/index.php?page=book&id=363975>.

4. Программная инженерия: учебное пособие / сост. Т.В. Киселева ; Министерство образования и науки РФ, Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет». - Ставрополь : СКФУ, 2017. - Ч. 1. - 137 с. : ил. - Библиогр. в кн. ; То же [Электронный ресурс]. - URL: <http://biblioclub.ru/index.php?page=book&id=467203>.

Дополнительно

1. Леоненков А.В. «Самоучитель UML 2» - СПб.: БХВ-Петербург, 2007. – 576 с.: ил.
2. Техника разработки программ: учебник для студентов вузов: в 2 кн. / Е.В. Крылов, В.А. Острейковский, Н.Г. Типикин. - М.: Высшая школа, 2007 - 2008 - (Для высших учебных заведений. Информатика и вычислительная техника)
3. Современные методы описания функциональных требований к системам / Алистер Коберн; Пер. с англ. - М.: Лори, 2002. - 265 с.
4. UML 2.0. Объектно-ориентированное моделирование и разработка. 2-е изд. / Дж. Рамбо, М.Блаха. - СПб.: Питер, 2007. - 544 с.: ил.