МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Методические указания для выполнения
практических работ
по дисциплине

Проектирование АСОИУ

для студентов специальности
230102.65
«Автоматизированные системы обработки информации и
управления»

Томск - 2012

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Кафедра автоматизации обработки информации

Утверждаю:
Зав. каф. АОИ
профессор
_____Ю.П. Ехлаков
«__» _____2012 г.

Методические указания для выполнения
практических работ
по дисциплине

Проектирование АСОИУ
для студентов специальности
230102.65
«Автоматизированные системы обработки информации
и управления »

Разработчик:
ст. преподаватель каф. АОИ
_____ Д.А. Соловьев

Содержание

Введение

«Проектирование АСОИУ» относится к циклу специальных дисциплин подготовки студентов специальности 230102.65 «Автоматизированные системы обработки информации и управления».

Целью данного курса является формирование у студентов, обучающихся по специальности 230102, навыков, позволяющих формулировать и решать задачи проектирования автоматизированных систем обработки информации путем создания функциональных и структурных спецификаций, применять полученные специальные знания для решения частных задач разработки АСОИУ конкретного (специального) назначения.

Данные методические указания предназначены для выполнения практических занятий по дисциплине «Проектирование АСОИУ» для студентов специальности 230102.65 «Автоматизированные системы обработки информации и управления».

1. Практическое занятие № 1. Разработка диаграммы прецедентов.

Цель занятия:

Ознакомиться с методологией моделирования прецедентов на основе языка UML.

Содержание работы и методические указания к ее выполнению

UML (Universal Modeling Language) - универсальный язык моделирования, который был разработан компанией Rational Software с целью создания наиболее оптимального и универсального языка для описания как предметной области, так и конкретной задачи в программировании. Визуальное моделирование в UML можно представить как некоторый процесс поуровневого спуска от наиболее общей и абстрактной концептуальной модели системы к логической, а затем и к физической модели соответствующей системы. Любая задача, таким образом, моделируется при помощи некоторого набора иерархических диаграмм, каждая из которых представляет собой некоторую проекцию системы.

Диаграмма (Diagram) - это графическое представление множества элементов. Чаще всего она изображается в виде связного графа с вершинами (сущностями) и ребрами (отношениями).

В UML определено восемь видов диаграмм:

диаграмма прецедентов (Use case diagram) - диаграмма поведения, на которой показано множество прецедентов и актеров, а также отношения между ними;

диаграмма деятельности (Activity diagram) - диаграмма поведения, на которой показан автомат и подчеркнуты переходы потока управления от одной деятельности к другой;

диаграмма классов (Class diagram) - структурная диаграмма, на которой показано множество классов, интерфейсов, кооперaций и отношения между ними;

диаграмма состояний (Statechart diagram) - диаграмма поведения, на которой показан автомат и подчеркнуто поведение объектов с точки зрения порядка получения событий;

диаграмма последовательностей (Sequence diagram) - диаграмма поведения, на которой показано взаимодействие и подчеркнута временная последовательность событий;

диаграмма кооперации (Collaboration diagram) - диаграмма поведения, на которой показано взаимодействие и подчеркнута структурная организация объектов, посылающих и принимающих сообщения;

диаграмма компонентов (Component diagram) - диаграмма поведения, на которой показан автомат и подчеркнуто поведение объектов с точки зрения порядка получения событий

диаграмма развертывания (Deployment diagram) - структурная диаграмма, на которой показаны узлы и отношения между ними.

Диаграмма прецедентов

Диаграммы прецедентов применяются для моделирования вида системы с точки зрения внешнего наблюдателя. На диаграмме прецедентов графически показана совокупность прецедентов и Субъектов, а также отношения между ними.

Рассмотрим основные элементы диаграммы прецедентов.

Субъект (actor) - любая сущность, взаимодействующая с системой извне или множество логически связанных ролей, исполняемых при взаимодействии с прецедентами. Стандартным графическим обозначением субъекта на диаграммах является фигурка "человечка", под которой записывается конкретное имя субъекта, однако субъектом может быть не только человек, но и техническое устройств о, программа или любая другая система, которая может служить источником воздействия на моделируемую систему так, как определит сам разработчик.

Прецеденты (use case) - это описание множества последовательностей действий (включая их варианты), которые выполняются системой для того, чтобы актер получил результат, имеющий для него определенное значение. При этом ничего не говорится о том, каким образом будет реализовано взаимодействие субъектов с системой, это одна из важнейших особенностей разработки прецедентов. Стандартным графическим обозначением прецедента на диаграммах является эллипс, внутри которого содержится краткое название прецедента или имя в форме глагола с пояснительными словами.

Сущность концепции прецедентов подразумевает несколько важных пунктов.

Прецедент представляет собой завершенный фрагмент функциональных возможностей (включая основной поток логики управления, его любые вариации (подпотоки) и исключительные условия (альтернативные потоки)).

Фрагмент внешне наблюдаемых функций (отличных от внутренних функций).

Ортогональный фрагмент функциональных возможностей (прецеденты могут при выполнении совместно использовать объекты, но выполнение каждого прецедента независимо от других прецедентов).

Фрагмент функциональных возможностей, инициируемый субъектом. Будучи инициирован, прецедент может взаимодействовать с другими субъектоми. При этом возможно, что субъект окажется только на принимающем конце прецедента, опосредованно инициированного другим субъектом.

Фрагмент функциональных возможностей, который предоставляет субъекту ощутимый полезный результат (и этот результат достигается в пределах одного прецедента).

Между субъектами и прецедентами - основными компонентами диаграммы прецедентов - могут существовать различные отношения, которые описывают взаимодействие экземпляров одних субъектов и прецедентов с экземплярами других субъектов и прецедентов. В языке UML имеется несколько стандартных видов отношений между субъектами и прецедентами:

Отношение ассоциации (association) - определяет наличие канала связи между экземплярами субъекта и прецедента (или между экземплярами двумх субъектов). Обозначается сплошной линией, возможно наличие стрелки и указание мощности связи.

Отношение расширения (extend) - определяет взаимосвязь экземпляров отдельного прецедента с более общим прецедентом, свойства которого определяются на основе способа совместного объединения данных экземпляров. Обозначается пунктирной линией со стрелкой, направленной от того прецедента, который является расширением для исходного прецедента, и помечается ключевым словом "extend" ("расширяет").

Отношение включения (include) - указывает, что некоторое заданное поведение для одного прецедента включает в качестве составного компонента поведение другого прецедента. Данное отношение является направленным бинарным отношением в том смысле, что пара экземпляров прецедентов всегда упорядочена в отношении включения. Обозначается пунктирной линией со стрелкой, направленной от базового прецедента к включаемому, и помечается ключевым словом "include" ("включает").

Отношение обобщения (generalization) - служит для указания того факта, что некоторый прецедент А может быть обобщен до прецедента В. В этом случае прецедент А будет являться специализацией прецедента В. При этом В называется предком или родителем по отношению к А, а прецедент А - потомком по отношению к прецеденту В. Следует подчеркнуть, что потомок наследует все свойства и поведение своего родителя, а также может быть дополнен новыми свойствами и особенностями поведения. Графически данное

отношение обозначается сплошной линией со стрелкой в форме незакрашенного треугольника, которая указывает на родительский прецедент.

Пример. Магазин видеопродукции.

Магазин продает видеокассеты, DVD-диски, аудио-кассеты, CD-диски и т.д. , а также предлагает широкой публике прокат видеокассет и DVD-дисков.

Товары поставляются несколькими поставщиками. Каждая партия товара предварительно заказывается магазином у некоторого поставщика и доставляется после оплаты счета. Вновь поступивший товар маркируется, заносится в базу данных и затем распределяется в торговый зал или прокат.

Видеоносители выдаются в прокат на срок от 1 до 7 дней. При прокате с клиента взимается залоговая стоимость видеоносителя. При возврате видеоносителя возвращается залоговая стоимость минус сумма за прокат. Если возврат задержан менее чем на 2 дня, взимается штраф в размере суммы за прокат за 1 день* кол-во дней задержки. При задержке возврата более чем на 2 дня - залоговая сумма не возвращается. Клиент может взять одновременно до 4 видеоносителей (прокат-заказ). На каждый видеоноситель оформляется квитанция.

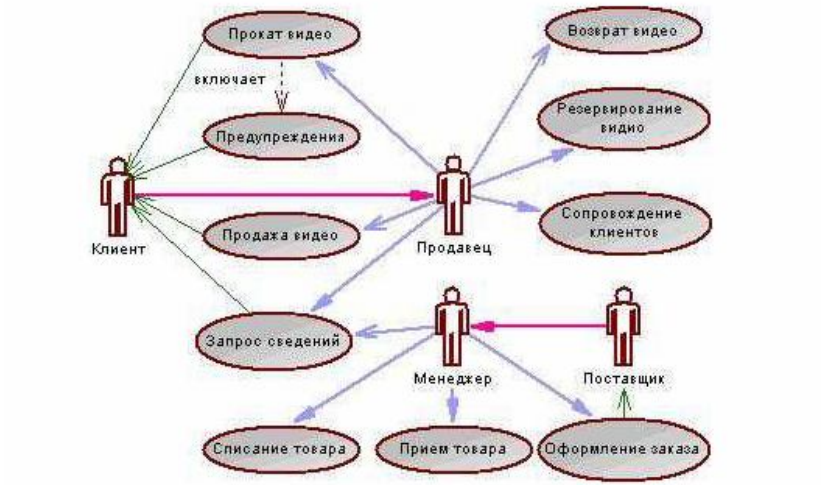
Клиенты могут стать членами видео-клуба и получить пластиковые карточки. С членов клуба не берется залог (за исключением случая описанного ниже), устанавливается скидка на ставку проката и покупку товаров. Члены клуба могут делать предварительные заказы на подбор видеоматериалов для проката или покупки.

Каждый член клуба имеет некоторый стутус. Первоначально - "новичок". При возврате всрок 5 прокат-заказов , статус меняется на "надежный". При задержке хотя бы одного видеоносителя более чем на 2 дня , статус "новичок" или "надежный" меняется на "ненадежный" и клиенту высылается предупреждение. При повторном нарушении правил статус меняется на "нарушитель". Члены клуба со статусом "надежный" могут брать до 8 видеоносителей единовременно, все остальные - 4. С членов клуба со статусом "нарушитель" берется залоговая сумма.

Клиенты при покупке товара или получении видеоносителя в прокат могут расплачиваться наличными или кредитной картой.

Прокатные видеоносители через определенное количество дней проката списываются и утилизируются по акту. Списываются также товары и прокатные видеоносители, у которых обнаружился брак.

На рисунке ниже приведена диаграмма прецедентов для рассматриваемого примера.



В этом примере можно выделить следующие субъекты и соответствующие им прецеденты:

Менеджер - изучает рынок видеопродукции, анализирует продажи (прецедент "Запрос сведений"), работает с поставщиками: составляет заявки на поставки товара (прецедент "Оформление заказа"), оплачивает и принимает товар (прецедент "Прием товара"), списывает товар (прецедент "Списание товара").

Продавец - работает с клиентами: продает товар (прецедент "Продажа видео"), оформляет членство в клубе (прецедент "Сопровождение клиентов"), резервирует (прецедент "Резервирование видио"), выдает в прокат (прецедент "Прокат видео") и принимает назад видеоносители (прецедент "Возврат видео"), отвечает на вопросы клиента (прецедент "Запрос сведений").

Поставщик - оформляет документы для оплаты товара (прецедент "Оформление заказа"), поставляет товар (прецедент "Прием товара"))

Клиент - покупает(прецедент "Продажа видео"), берет на прокат и возвращает видеоносители (прецеденты "Прокат видео" и "Возврат видео"), вступает в клуб (прецедент "Сопровождение клиентов"), задает вопросы (прецедент "Запрос сведений").

Последние два субъекта Поставщик и Клиент не будут иметь непосредственного доступа к разрабатываемой системе (второстепенные субъекты), однако именно они являются основным источником событий, инициализирующих прецеденты, и получателями результата работы прецедентов

От прецедента "Прокат видео" к прецеденту "Предупреждения" установлено отношение включения на том основании, что каждый выданный видеоноситель должен быть проверен на своевременный возврат и, в случае необходимости, выдано предупреждение клиенту.

Дальнейшее развитие модели поведения системы предполагает спецификацию прецедентов. Для этого традиционно используют два способа.

Первый - описание с помощью текстового документа. Такой документ описывает, что должна делать система, когда субъект инициировал прецедент. Типичное описание содержит следующие разделы:

- Краткое описание
- Участвующие субъекты
- Предусловия, необходимые для инициирования прецедента
- Поток событий (основной и, возможно, подпотоки, альтернативный
- Постусловия, определяющие состояние системы, по достижении которого прецедент завершается.

Описательная спецификация прецедента "Прокат видео"

Раздел	*Описание*
Краткое описание	Клиент желает взять на прокат видеокассету или диск, которые снимаются с полки магазина или были предварительно зарезервированы клиентом. При условии, что у клиента нет невозвращенных в срок видеоносителей, сразу после внесения платы фильм выдается напрокат. Если невозвращенные в срок видеоносители есть, клиенту выдается напоминание о просроченном возврате.
Субъекты	Продавец, Клиент.
Предусловия	В наличие имеются видеокассеты или диски, которые можно взять напрокат. У клиентов есть клубные карточки. Устройство сканирования работает правильно. Работники за прилавком знают, как обращаться с системой
Основной поток	Клиент может назвать номер заказа или взять видеоноситель с полки. Видеоноситель и членская карточка сканируются и продавцу не сообщается никаких сведений о задержках, так, что он не задает клиенту соответствующих вопросов. Если клиент имеет статус <надежный>, он может взять до 8 видеоносителей, во всех остальных случаях - до 4-х. Если статус клиента определен как <нарушитель>, его просят внести задаток. Клиент расплачивается наличными или кредитной картой. После получения суммы, информация о наличии фильмов обновляется и видеоносители передаются клиенту вместе с квитанциями на прокат. О прокате каждого видеоносителя делается отдельная запись с указанием идентификационного номера клиента, даты проката, даты возврата, идентификационного номера продавца, полученной суммы. Прецедент генерирует предупреждения о просроченном возврате клиенту, если видеофильм не был возвращен в течение двух дней по истечении даты возврата и изменяет статус клиента на <ненадежный> (первое на-

	рушение) или <нарушитель> (повторное нарушение).
Альтернативный поток	У клиента нет членской карточки. В этом случае прецедент <Сопровождение клиента> может быть активизирован для выдачи новой карточки. Видеофильмы не выдаются, поскольку у клиента есть невозвращенные в срок видеоносители. Попытка взять на прокат слишком много видеоносителей. Видеоноситель или кредитная карта не могут быть отсканированы из-за их повреждения. У клиента не хватило наличных или платеж по кредитной карте отклонен.
Постусловия	Видеофильмы сданы напрокат, и база данных соответствующим образом обновлена

Последовательность проведения занятия:

1. Ознакомиться с методологией моделирования прецедентов на основе языка UML.
3. Построить диаграмму прецедентов для своей предметной области.
4. Описать несколько (2-3) прецедентов.
5. Защитить построенную диаграмму.

Вопросы для самоконтроля:

1. В чем смысл прецедента?
2. Назначение прецедента.
3. Назовите основные компоненты диаграммы прецедентов.
4. Что такое действующее лицо (Actor)?
5. Какую роль могут играть действующие лица по отношению к прецеденту?
6. Назначение обобщения.
7. Аспект в диаграмме прецедентов.

2. Практическое занятие № 2. Моделирование производства с использованием языка унифицированного моделирования.

Цель занятия.

Ознакомиться с технологическим процессом моделирования производства.

Содержание работы и методические указания к ее выполнению.

Цели моделирования производства состоят в следующем.

- Познание структуры и динамики организации (целевой организации), в которой будет использоваться разрабатываемая система.
- Осмысление текущих проблем целевой организации и определение возможностей улучшения.
- Обеспечение общего понимания целевой организации заказчиками, конечными пользователями и разработчиками.
- Определение требований к системе, необходимых для поддержки целевой организации.

Для достижения этих целей нужен технологический процесс моделирования производства, описывающий, как разработать видение новой целевой организации и определить процессы, роли и обязанности этой организации в модели производства. В модель производства входят модель производственных прецедентов и модель объектов производства.

Для более полного удовлетворения требований необходимо вначале попытаться понять область производства, а лишь затем приступать к проекту разработки программного обеспечения (в крайнем случае, нужно выполнять эти действия параллельно).

Сценарии моделирования производства

В зависимости от сути производства и требований к нему возможны шесть сценариев его моделирования.

Сценарий 1. Организационная схема

Может возникнуть желание создать простую схему организации и ее процессов, которая должна помочь понять требования создаваемого приложения. В этом случае моделирование производства — это часть программно-технического проекта, обычно выполняемая в фазе исследования.

Сценарий 2. Моделирование предметной области

При создании приложений, основной целью которых является предоставление информации и управление ею (например, система управления заказами или банковская система), модель этой информации можно создать на производственном уровне, не рассматривая производственные технологические процессы. Называется этот процесс моделированием предметной области. Обычно это часть программно-технического проекта, выполняемая в фазах исследования и уточнения плана.

Сценарий 3. Одно производство для нескольких систем

При создании большой системы или семейства приложений может получиться так, что одна работа, связанная с моделированием производства, будет использоваться в качестве основы для нескольких программно-технических проектов. В этом случае модели производства помогут выявить функциональные требования, а также послужат основой для создания архитектуры семейства приложений. В этом случае моделирование производства часто рассматривается как отдельный проект.

Сценарий 4. Общая модель производства

При разработке приложения, которое будет использоваться несколькими организациями (например, приложение организации продаж или составления счетов), полезно будет усреднить способы ведения организациями дел. Это поможет избежать слишком больших требований к системе. Если же усреднение невозможно, то работы по моделированию производства помогут

уяснить, чем будет отличаться использование приложения в разных организациях, а также помогут понять, как использовать эту информацию и как распределить приоритеты функциональных возможностей приложения.

Сценарий 5. Новое производство

Если организация решила открыть совершенно новую сферу деятельности и создать для ее поддержки информационные системы, то в этом случае также требуется проведение работ по моделированию производства. Цель этих работ - определить не только требования к системам, но и реальность реализации нового проекта. В этом случае работы по моделированию производства также часто рассматриваются как отдельный проект.

Сценарий 6. Реорганизация

Если организация решает полностью пересмотреть свой способ ведения дел (с помощью исходных текстов создать новую структурную схему и алгоритм работы производственного процесса), то моделирование производства — это зачастую один из нескольких самостоятельных процессов. Реорганизация, как правило, выполняется в несколько этапов: представление в основных чертах нового предприятия, "переконструирование" существующего процесса, создание нового процесса и его установка.

Исполнители и артефакты

В Rational Unified Process все сказанное выше выражается через исполнителей, артефакты, виды деятельности и технологические процессы. Рассмотрим рисунок, на котором показаны основные исполнители и артефакты процесса моделирования производства.

Исполнители и артефакты технологического процесса моделирования

В процессе моделирования производства задействованы следующие основные исполнители.

Аналитик процесса производства возглавляет и координирует моделирование производственных прецедентов. Для этого он очерчивает и ограничивает моделируемую организацию.

Разработчик производства уточняет спецификацию части организации путем описания одного или нескольких производственных прецедентов. Он указывает, какие производственные исполнители и категории производства требуются для реализации производственных прецедентов, а также определяет принципы их совместной работы. Разработчик производства устанавливает обязанности, действия, параметры одного или нескольких производственных исполнителей, а также их связь с категориями производства.

Кроме того, в технологическом процессе участвуют следующие исполнители.

Заинтересованные стороны, организовывающие общий надзор и предоставляющие информацию.

Рецензент производства, представляющий рецензию на результирующие артефакты.

В процессе моделирования производства создаются ключевые артефакты.

Документ видения производства: определяет цели моделирования производства.

Модель производственных прецедентов: модель предполагаемых функций производства. Необходимая основа для определения ролей и комплектующих узлов.

Модель объектов производства: объектная модель, описывающая реализацию производственных прецедентов.

Кроме того, создаются следующие артефакты.

Оценка целевой организации: описывает текущее состояние организации, в которой будет использоваться система.

Правила производства: определение политики или условий, которые должны удовлетворяться.

Дополнительная производственная спецификация: документ, представляющий определения производства, не включенные в модель производственных прецедентов или модель объектов производства.

Словарь производства: определяет важные термины, используемые в производстве.

Модель производственных прецедентов включает производственные акторы и производственные прецеденты. Акторы представляют роли, внешние по отношению к производству (например, роли заказчиков), а производственные прецеденты — производимые процессы.

Модель объектов производства включает реализации производственных прецедентов, показывающие, как производственные прецеденты "выполняются" в терминах производственных акторов и категорий производства.

Последовательность проведения занятия:

1. Получить у преподавателя вариант задания (предметную область).

2. Для рассматриваемой предметной области выделить 3-4 процесса, выполнение которых является определяющим для работы системы.
3. Для каждого из процессов определить его границы и основные составляющие его элементы.
4. Построить диаграмму производственных прецедентов для рассматриваемой предметной области.
5. Защитить выполненное задание перед преподавателем.

Вопросы для самоконтроля

1. В чем состоят цели моделирования производства?
2. Какие существуют сценарии производства?
3. Перечислите основных исполнителей, участвующих в процессе моделирования производства.

3. Практическое занятие № 3. Шаблоны проектирования, реинжиниринг информационной системы.

Цель занятия.

Ознакомление с шаблонами проектирования, использование средств реинженеринга Rational Rose.

Содержание работы и методические указания по выполнению.

В разработке программного обеспечения, шаблон проектирования или паттерн (англ. design pattern) — повторимая архитектурная конструкция, представляющая собой решение проблемы проектирования в рамках некоторого часто возникающего контекста.

Обычно шаблон не является законченным образцом, который может быть прямо преобразован в код; это лишь пример решения задачи, который можно использовать в различных ситуациях. Объектно-ориентированные шаблоны показывают отношения и взаимодействия между классами или объектами, без определения того, какие конечные классы или объекты приложения будут использоваться.

«Низкоуровневые» шаблоны, учитывающие специфику конкретного языка программирования, называются идиомами. Это хорошие решения проектирования, характерные для конкретного языка или программной платформы, и потому не универсальные.

На наивысшем уровне существуют архитектурные шаблоны, они охватывают собой архитектуру всей программной системы.

Алгоритмы по своей сути также являются шаблонами, но не проектирования, а вычисления, так как решают вычислительные задачи.

Главная польза каждого отдельного шаблона состоит в том, что он описывает решение целого класса абстрактных проблем. Также тот факт, что каждый шаблон имеет свое имя, облегчает дискуссию об абстрактных структурах данных (ADT) между разработчиками, так как они могут ссылаться на известные шаблоны. Таким образом, за счёт шаблонов производится унификация терминологии, названий модулей и элементов проекта.

Правильно сформулированный шаблон проектирования позволяет, отыскав удачное решение, пользоваться им снова и снова.

Типы шаблонов проектирования.

ОСНОВНЫЕ ШАБЛОНЫ

Шаблон делегирования - объект внешне выражает некоторое поведение, но в реальности передаёт ответственность за выполнение этого поведения связанному объекту.

Шаблон функционального дизайна - гарантирует, что каждый модуль компьютерной программы имеет только одну обязанность и исполняет её с минимумом побочных эффектов на другие части программы.

Неизменяемый объект - объект, который не может быть изменён после своего создания.

Интерфейс - общий метод для структурирования компьютерных программ для того, чтобы их было проще понять.

ПОРОЖДАЮЩИЕ ШАБЛОНЫ

Шаблоны проектирования, которые абстрагируют процесс инстанцирования. Они позволяют сделать систему независимой от способа создания, композиции и представления объектов. Шаблон, порождающий классы, использует наследование, чтобы изменять инстанцируемый класс, а шаблон, порождающий объекты, делегирует инстанцирование другому объекту.

Абстрактная фабрика - класс, который представляет собой интерфейс для создания компонентов системы.

Строитель - класс, который представляет собой интерфейс для создания сложного объекта.

Фабричный метод - определяет интерфейс для создания объекта, но оставляет подклассам решение о том, какой класс инстанциировать.

Отложенная инициализация - объект, инициализируемый во время первого обращения к нему.

Пул одиночек - гарантирует, что класс имеет поименованные экземпляры объекта и обеспечивает глобальную точку доступа к ним.

Объектный пул - класс, который представляет собой интерфейс для работы с набором инициализированных и готовых к использованию объектов.

Прототип - определяет интерфейс создания объекта через клонирование другого объекта вместо создания через конструктор.

Получение ресурса есть инициализация - получение некоторого ресурса совмещается с инициализацией, а освобождение — с уничтожением объекта.

Одиночка - класс, который может иметь только один экземпляр.

СТРУКТУРНЫЕ ШАБЛОНЫ

Определяют различные сложные структуры, которые изменяют интерфейс уже существующих объектов или его реализацию, позволяя облегчить разработку и оптимизировать программу.

Адаптер - объект, обеспечивающий взаимодействие двух других объектов, один из которых использует, а другой предоставляет несовместимый с первым интерфейс.

Мост - структура, позволяющая изменять интерфейс обращения и интерфейс реализации класса независимо.

Компоновщик - объект, который объединяет в себе объекты, подобные ему самому.

Декоратор (Обёртка) - класс, расширяющий функциональность другого класса без использования наследования.

Фасад - объект, который абстрагирует работу с несколькими классами, объединяя их в единое целое.

Единая точка входа - обеспечивает унифицированный интерфейс для интерфейсов в подсистеме. Определяет высокоуровневый интерфейс, упрощающий использование подсистемы.

Приспособленец - это объект, представляющий себя как уникальный экземпляр в разных местах программы, но по факту не являющийся таковым.

Заместитель - объект, который является посредником между двумя другими объектами, и который реализовывает/ограничивает доступ к объекту, к которому обращаются через него.

ПОВЕДЕНЧЕСКИЕ ШАБЛОНЫ

Определяют взаимодействие между объектами, увеличивая таким образом его гибкость.

Цепочка ответственности - предназначен для организации в системе уровней ответственности.

Команда - представляет действие. Объект команды заключает в себе само действие и его параметры.

Интерпретатор - решает часто встречающуюся, но подверженную изменениям, задачу.

Итератор - представляет собой объект, позволяющий получить последовательный доступ к элементам объекта-агрегата без использования описаний каждого из объектов, входящий в состав агрегации.

Посредник - обеспечивает взаимодействие множества объектов, формируя при этом слабую связанность и избавляя объекты от необходимости явно ссылаться друг на друга.

Хранитель - позволяет не нарушая инкапсуляцию зафиксировать и сохранить внутреннее состояния объекта так, чтобы позднее восстановить его в этом состоянии (1). Предотвращает нулевые указатели, предоставляя объект «по умолчанию» (2).

Наблюдатель - определяет зависимость типа «один ко многим» между объектами таким образом, что при изменении состояния одного объекта все зависящие от него оповещаются об этом событии.

Слуга - используется для обеспечения общей функциональности группе классов (1). Служит для связывания бизнес-логики (2).

Состояние - используется в тех случаях, когда во время выполнения программы объект должен менять свое поведение в зависимости от своего состояния.

Стратегия - предназначен для определения семейства алгоритмов, инкапсуляции каждого из них и обеспечения их взаимозаменяемости.

Шаблонный метод - определяет основу алгоритма и позволяет наследникам переопределять некоторые шаги алгоритма, не изменяя его структуру в целом.

Посетитель - описывает операцию, которая выполняется над объектами других классов. При изменении класса нет необходимости изменять обслуживаемые классы.

Single-serving visitor - оптимизирует реализацию шаблона посетитель, который инициализируется, единожды используется, и затем удаляется.

Hierarchical visitor - предоставляет способ обхода всех вершин иерархической структуры данных (напр. древовидной).

Объектно-ориентированные CASE-средства (Rational Rose)

Rational Rose - CASE-средство фирмы Rational Software Corporation (США) - предназначено для автоматизации этапов анализа и проектирования программного обеспечения, а также для генерации кодов на различных языках и выпуска проектной документации. Rational Rose использует синтез-методологию объектно-ориентированного анализа и проектирования, основанную на подходах трех ведущих специалистов в данной области: Буча, Рамбо и Джекобсона. Разработанная ими универсальная нотация для моделирования объектов (UML - Unified Modeling Language) претендует на роль стандарта в области объектно-ориентированного анализа и проектирования. Конкретный вариант Rational Rose определяется языком, на котором генерируются коды программ (C++, Smalltalk, PowerBuilder, Ada, SQLWindows и ObjectPro). Основной вариант - Rational Rose/C++ - позволяет разрабатывать проектную документацию в виде диаграмм и спецификаций, а также генерировать программные коды на C++. Кроме того, Rational Rose содержит средства реинжиниринга программ, обеспечивающие повторное использование программных компонент в новых проектах.

Структура и функции

В основе работы Rational Rose лежит построение различного рода диаграмм и спецификаций, определяющих логическую и физическую структуры модели, ее статические и динамические аспекты. В их число входят диаграммы классов, состояний, сценариев, модулей, процессов.

В составе Rational Rose можно выделить 6 основных структурных компонент: репозиторий, графический интерфейс пользователя, средства просмотра проекта (browser), средства контроля проекта, средства сбора статистики и генератор документов. К ним добавляются генератор кодов (индивидуальный для каждого языка) и анализатор для C++, обеспечивающий реинжиниринг - восстановление модели проекта по исходным текстам программ.

Репозиторий представляет собой объектно-ориентированную базу данных. Средства просмотра обеспечивают "навигацию" по проекту, в том числе, перемещение по иерархиям классов и подсистем, переключение от одного вида диаграмм к другому и т. д. Средства контроля и сбора статистики дают возможность находить и устранять ошибки по мере развития проекта, а не после завершения его описания. Генератор отчетов формирует тексты выходных документов на основе содержащейся в репозитории информации.

Средства автоматической генерации кодов программ на языке C++, используя информацию, содержащуюся в логической и физической моделях проекта, формируют файлы заголовков и файлы описаний классов и объектов. Создаваемый таким образом скелет программы может быть уточнен путем прямого программирования на языке C++. Анализатор кодов C++ реализован в

виде отдельного программного модуля. Его назначение состоит в том, чтобы создавать модули проектов в форме Rational Rose на основе информации, содержащейся в определяемых пользователем исходных текстах на C++. В процессе работы анализатор осуществляет контроль правильности исходных текстов и диагностику ошибок. Модель, полученная в результате его работы, может целиком или фрагментарно использоваться в различных проектах. Анализатор обладает широкими возможностями настройки по входу и выходу. Например, можно определить типы исходных файлов, базовый компилятор, задать, какая информация должна быть включена в формируемую модель и какие элементы выходной модели следует выводить на экран. Таким образом, Rational Rose/C++ обеспечивает возможность повторного использования программных компонент.

В результате разработки проекта с помощью CASE-средства Rational Rose формируются следующие документы:

диаграммы классов;
диаграммы состояний;
диаграммы сценариев;
диаграммы модулей;
диаграммы процессов;
спецификации классов, объектов, атрибутов и операций
заготовки текстов программ;
модель разрабатываемой программной системы.

Последний из перечисленных документов является текстовым файлом, содержащим всю необходимую информацию о проекте (в том числе необходимую для получения всех диаграмм и спецификаций).

Тексты программ являются заготовками для последующей работы программистов. Они формируются в рабочем каталоге в виде файлов типов .h (заголовки, содержащие описания классов) и .cpp (заготовки программ для методов). Система включает в программные файлы собственные комментарии, которые начинаются с последовательности символов //##. Состав информации, включаемой в программные файлы, определяется либо по умолчанию, либо по усмотрению пользователя. В дальнейшем эти исходные тексты развиваются программистами в полноценные программы.

Последовательность проведения занятия:

1. Получить вариант задания у преподавателя.
2. Выполнить проектирование заданной предметной области с помощью шаблонов проектирования
3. Выполнить генерацию скелета программы
4. Доработать программу
5. Выполнить реинжиниринг полученной программы средствами Rational Rose.

Вопросы для самоконтроля

1. Перечислите основные шаблоны.

2. Перечислите поведенческие шаблоны.
3. Приведите пример использования шаблона Адаптер.
4. Определите понятие реинжиниринга программ.

4. Практическое занятие № 4. Мозговой штурм, совещание, посвященное требованиям.

Цель занятия.

Формирование у студентов умения работать в команде. Ознакомление с основами управления требованиями.

Содержание работы и методические указания по выполнению.

Управление требованиями к программному обеспечению — процесс, включающий идентификацию, выявление, документирование, анализ, отслеживание, приоритезацию требований, достижение соглашения по требованиям и затем управление изменениями и уведомление соответствующих заинтересованных лиц. Управление требованиями — непрерывный процесс на протяжении всего проекта разработки программного обеспечения.

Цель управления требованиями состоит в том, чтобы гарантировать что организация документирует, проверяет и удовлетворяет потребности и ожидания её клиентов и внутренних или внешних заинтересованных лиц. Управление требованиями начинается с выявления и анализа целей и ограничений клиента. Управление требованиями далее включает поддержку требований, интеграцию требований и организацию работы с требованиями и сопутствующей информацией, поставляющейся вместе с требованиями.

Управление требованиями включает общение между проектной командой и заинтересованными лицами с целью корректировки требований на протяжении всего проекта. Постоянное общение всех участников проекта важно для того, чтобы ни один класс требований не доминировал над другими.

Задачи управления требованиями

На каждом этапе процесса разработки существуют ключевые методы и задачи связанные с управлением требованиями. Для примера, рассмотрим к стандартный процесс разработки с пятью фазами: исследованием, анализом осуществимости, дизайном, разработкой и тестированием и выпуском.

Исследование

Во время фазы исследования собираются первые три класса требований от пользователей, бизнеса и команды разработчиков. В каждой области задают одинаковые вопросы: каковы цели, каковы ограничения, какие используются процессы и инструменты и так далее. Только когда эти требования хорошо поняты, можно приступать к разработке функциональных требований.

Результатом стадии исследования является документ — спецификация требований, одобренный всеми членами проекта. В то время как многие организаций всё ещё используют обычные документы для управления требованиями, другие управляют своими базовыми требованиями, используя программные инструменты. Эти инструменты управляют требованиями используя базу данных, и обычно имеют функции автоматизации отслеживаемости (например, позволяя создавать связи между родительскими и дочерними требованиями, или между тестами и требованиями), управления версиями, и управле-

ния изменениями. Обычно такие инструментальные средства содержат функцию экспорта, которая позволяет создавать обычный документ, экспортируя данные требований.

Анализ осуществимости

На стадии анализа осуществимости определяется стоимость требований.

Для пользовательских требований текущая стоимость работы сравнивается с будущей стоимостью установленной системы. Задаются вопросы такие как: «Сколько нам сейчас стоят ошибки ввода данных?» Или, «Какова стоимость потери данных из-за ошибки оператора связанной с используемым интерфейсом?». Фактически, потребность в новом инструменте часто распознаётся, когда подобные вопросы попадают во внимание людей, занимающихся в организации финансами.

Деловая стоимость включает ответы на такие вопросы как: «У какого отдела есть бюджет для этого?» «Каков уровень возврата средств от нового продукта на рынке?» «Каков уровень сокращения внутренних расходов на обучение и поддержку, если мы сделаем новую, более простую в использовании систему?»

Техническая стоимость связана со стоимостью разработки программного обеспечения и аппаратной стоимостью. «Есть ли у нас нужные люди, чтобы создать инструмент?» «Нуждаемся ли мы в новом оборудовании для поддержки новой системы?»

Результатом стадии анализа осуществимости является бюджет и график проекта.

Дизайн

Предположение, что стоимость точно определена и преимущества, которые будут получены, являются достаточно большими, проект может перейти к стадии проектирования.

На стадии дизайна основная деятельность управления требованиями состоит в том, чтобы проверять соответствуют ли результаты дизайна документу требований, чтобы удостовериться, что работа остается в границах проекта.

Документ требований становится ключевым инструментом, который помогает команде принимать решения об изменениях дизайна.

Разработка и тестирование

На стадии разработки и тестирования, основная деятельность управления требованиями — это гарантировать, что работа и цена остаются в пределах графика и бюджета, и что создаваемый инструмент действительно отвечает требованиям. Основным инструментом, используемым на этой стадии, является создание прототипа и итерационное тестирование. Для программного приложения пользовательский интерфейс может быть создан на бумаге и проверен с потенциальными пользователями, в то время как создаётся основа программы. Результаты этих тестов записываются в руководстве по дизайну пользовательского интерфейса и передаются коллективу дизайнеров.

Выпуск

Управление требованиями не заканчивается выпуском продукта. С этого момента полученные данные о приемлемости приложения собираются и используются во время фазы исследования для следующего поколения системы или выпуска. Таким образом, процесс начинается снова.

Метод мозгового штурма (мозговой штурм, мозговая атака) — оперативный метод решения проблемы на основе стимулирования творческой активности, при котором участникам обсуждения предлагают высказывать как можно большее количество вариантов решения, в том числе самых фантастичных. Затем из общего числа высказанных идей отбирают наиболее удачные, которые могут быть использованы на практике. Является методом экспертного оценивания.

Этапы и правила мозгового штурма

Правильно организованный мозговой штурм включает три обязательных этапа. Этапы отличаются организацией и правилами их проведения:

Постановка проблемы. Предварительный этап. В начале этого этапа проблема должна быть четко сформулирована. Происходит отбор участников штурма, определение ведущего и распределение прочих ролей участников в зависимости от поставленной проблемы и выбранного способа проведения штурма.

Генерация идей. Основной этап, от которого во многом зависит успех (см. ниже) всего мозгового штурма. Поэтому очень важно соблюдать правила для этого этапа:

Главное — количество идей. Не делайте никаких ограничений.

Полный запрет на критику и любую (в том числе положительную) оценку высказываемых идей, так как оценка отвлекает от основной задачи и сбивает творческий настрой.

Необычные и даже абсурдные идеи приветствуются.

Комбинируйте и улучшайте любые идеи.

Группировка, отбор и оценка идей. Этот этап часто забывают, но именно он позволяет выделить наиболее ценные идеи и дать окончательный результат мозгового штурма. На этом этапе, в отличие от второго, оценка не ограничивается, а наоборот, приветствуется. Методы анализа и оценки идей могут быть очень разными. Успешность этого этапа напрямую зависит от того, насколько "одинаково" участники понимают критерии отбора и оценки идей.

Мозговые атаки

Для проведения мозговой атаки обычно создают две группы:

участники, предлагающие новые варианты решения задачи;

члены комиссии, обрабатывающие предложенные решения.

Различают индивидуальные и коллективные мозговые атаки.

В мозговом штурме участвует коллектив из нескольких специалистов и ведущий. Перед самим сеансом мозгового штурма ведущий производит четкую постановку задачи, подлежащей решению. В ходе мозгового штурма участники высказывают свои идеи, направленные на решение поставленной задачи, причём как логичные, так и абсурдные. Если в мозговом штурме принимают участие люди различных чинов или рангов, то рекомендуется заслушивать идеи в порядке возрастания ранжира, что позволяет исключить психологический фактор «соглашения с начальством».

В процессе мозгового штурма, как правило, вначале решения не отличаются высокой оригинальностью, но по прошествии некоторого времени типовые, шаблонные решения исчерпываются, и у участников начинают возникать

необычные идеи. Ведущий записывает или как-то иначе регистрирует все идеи, возникшие в ходе мозгового штурма.

Затем, когда все идеи высказаны, производится их анализ, развитие и отбор. В итоге находится максимально эффективное и часто нетривиальное решение задачи.

Успех

Успех мозгового штурма сильно зависит от психологической атмосферы и активности обсуждения, поэтому роль ведущего в мозговом штурме очень важна. Именно он может «вывести из тупика» и вдохнуть свежие силы в процесс.

Изобретателем метода мозгового штурма считается Алекс Осборн, сотрудник рекламного агентства BBD&O.

Порядок проведения занятия

1. Подготовка к проведению совещания;
2. Проведение совещания;
3. Мозговой штурм и отбор идей.

Результаты проведенного совещания используются на следующем практическом занятии.

5. Практическое занятие № 5. Работа с требованиями.

Цель занятия. В рамках занятия студенты приобретают навыки:

- создания проекта и настройки его свойств;
- определение типов и атрибутов требований;
- создание требований с использованием матрицы атрибутов.

Общие сведения о репозитарии. Репозитарий *«IBM Rational RequisitePro»* образуют:

база данных, реализованная в форме реляционных таблиц;

набор документов в формате *«Microsoft Word»*.

В базе данных хранится следующая информация:

типы документов;

типы требований, их описания и атрибуты;

протоколы дискуссий;

информация о трассировке требований;

сведения о политиках безопасности групп и отдельных пользователей.

Среда *«IBM Rational RequisitePro»* предоставляет в распоряжение разработчика шаблоны репозитариев, позволяющие оптимизировать решение конкретной задачи. *«IBM Rational RequisitePro»* версии 7.1.1 содержит:

«Use-Case Template» **(***«Шаблон прецедентов»*), использующий методологию сбора требований, основанную на анализе прецедентов;

«Traditional Template» **(***«Классический шаблон»***)**, основанный на классическом сборе требований к программному обеспечению без применения методологии прецедентов;

«Composite Template» **(***«Составной шаблон»***),** представляющий собой комбинацию упомянутых выше шаблонов.

«Business Modeling Template» **(***«Шаблон для моделирования организационной системы»***)** - шаблон, наиболее подходящий для моделирования организационных систем, поскольку содержит соответствующие типы документов и требований.

«RUP Template» **(***«Шаблон Унифицированного процесса от компании Rational»***)** - шаблон репозитария, предполагающий его создание в строгом соответствии с методологией RUP.

Кроме того, среда предоставляет возможность:

создать новый пользовательский шаблон **(***«Make New Template»);*

создать репозитарий с *«чистого листа» («Blank»);*

создать репозитарий на основании данных основного каталога **(***«Create from Baseline»***)**.

Перечисленные шаблоны поставляются со средой *«IBM Rational RequisitePro» версии 7.1.1* по умолчанию. Для среды *«Rational RequisitePro» версии 7.0.**0*** два последних шаблона должны быть инсталлированы дополнительно. Загрузить *«Business Modeling Template», «RUP Template»* можно с официального сайта компании IBM.

Версия среды 7.1.1 кроме перечисленных шаблонов содержит *"SAP Template"***.**

Следует отметить, что помимо добавления новых шаблонов эволюция *«IBM Rational RequisitePro»* направлена на обеспечение совместимости с более поздними версиями *«Microsoft Word»*.

*«IBM Rational RequisitePro» версии 7.0.0.0***,** идущий в составе пакета *«IBM Rational Suite 2003»* позволяет использовать *«Microsoft Word 2000»***.**

Апгрейд *«IBM Rational RequisitePro»* до *версии 7.0.1.0* позволяет использовать *Microsoft Word 2000/2003***.**

И, наконец, сборка *«IBM Rational RequisitePro» версии 7.1.1,* являющаяся наиболее актуальной, на момент написания пособия, позволяет использовать *«Microsoft Word 2000/2003/2007»* согласно информации, опубликованной на сайте компании IBM.

Обратите внимание: обратная совместимость формата хранения репозитария не поддерживается. Это означает, что репозитарии, созданные с использованием *«IBM Rational RequisitePro» версии 7.1.1* не будут корректно открыты более ранними версиями приложения.

Проблема локализации. Безусловно, шаблоны, содержащиеся в рамках пакета, предоставляют возможность сократить время, затрачиваемое на настройку структуры репозитария. Вместе с тем по понятным причинам их использование русскоязычными пользователями затруднено.

Существует три пути устранения указанной проблемы.

Первый достаточно трудоемкий путь предполагает использование англоязычного шаблона и последовательный перевод элементов, полученных на его основе, с использованием возможностей среды *«IBM Rational RequisitePro»***.** Учитывая, что приложение позволяет достаточно гибко настраивать репозитарий, можно утверждать, что полученный таким образом результат удовлетворит большинство пользователей. Проблемой, не решаемой исключительно в рамках среды, остается использование англоязычных шаблонов для артефактов, разрабатываемых с использованием *«Microsoft Word»*.

Локализацию этих составляющих репозитария необходимо выполнять в *«Microsoft Word»*.

Второй путь предполагает использование полностью русскоязычных шаблонов, которые необходимо предварительно создать.

Третий путь, рассмотренный в данной лабораторной работе, предполагает создание репозитария с «чистого листа», генерацию и использование русскоязычных элементов по мере их возникновения в репозитарии.

Не следует использовать символы национальных алфавитов при задании имени репозитария и пути его размещения. Сложности возникают при обращении к проекту посредством *«IBM Rational RequisiteWeb»* – Web-ориентированного приложения, в основном повторяющего функциональность *«IBM Rational RequisitePro»* и предназначенного для организации групповой распределенной работы над требованиями. Имя репозитария рекомендуется давать с использованием символов латинского алфавита, равно как и имя каталога, в котором физически будут располагаться файлы репозитария.

Для рассматриваемого в пособии примера данное ограничение несущественно, поскольку компонента *«IBM Rational RequisiteWeb»* использоваться не будет.

Текущая реализация *«IBM Rational RequisitePro»* позволяет организовать репозитарий требований в форматах СУБД *«Microsoft Access», «Microsoft SQL Server»* и *«Oracle»*. Представители компании-разработчика утверждают, что для команд численностью до 255 человек достаточно использовать базу данных, созданную на сервере в одной из папок общего доступа в формате СУБД *«Microsoft Access»*. Для более крупных проектов рекомендуется использовать *«Microsoft SQL Server»* или *«Oracle»*.

Создание проекта на основе чистого шаблона. Предметной областью, рассматриваемой в рамках упражнений, является процедура записи клиента на тест-драйв в салоне официального дилера “Chevrolet”.

Упражнение. Создание репозитария.

Процесс создания нового проекта начинается с запуска *«IBM Rational RequisitePro»*.

Если пакет *«IBM Rational RequisitePro»* был установлен автономно и группы созданы по умолчанию, необходимо вызвать пункты меню *Пуск / Программы / IBM Rational / Rational RequisitePro*. Если *«IBM Rational RequisitePro»* был установлен в составе IBM Rational Suite, необходимо вызвать пункты меню *Пуск / Программы / IBM Rational Suite / Rational RequisitePro*.

При настройках по умолчанию приложение предложит пройти процедуру быстрого ознакомления с возможностями среды, как показано на рис. 1.

Рисунок 1. Окно ознакомления с возможностями среды

Установите переключатель *«Show this on startup» («Показывать это окно при запуске»)* в положение *"отключен"***,** чтобы окно не появлялось при последующих запусках приложения.

Пропустите процедуру ознакомления с возможностями среды, закрыв окно, показанное на рис. 2.1. кнопкой *«Close» («Закрыть»).*

При настройках приложения по умолчанию в ответ на экран будет выведено окно создания репозитария, показанное на рис. 2.

Рисунок 2. Окно создания репозитария Rational RequisitePro.

По умолчанию в окне, показанном на рис.2.2, активна вкладка *«New» («Новый»)***.** Если это не так, выберите ее.

Кликните мышью по пиктограмме *«Blank» («Чистый шаблон»)***.** По умолчанию переключатель *«Detail» («Подробнее»)* находится в активированном состоянии и в текстовой области, расположенной внизу окна, отображается описание шаблона.

Нажмите кнопку *«Ok»***.** Приложение отобразит диалоговое окно, приведенное на рис.3.

Заполните поля *«Name» («Имя»)* и *«Description» («Описание»)* как показано на рисунке 3.

В случае необходимости измените каталог, в котором будут храниться файлы репозитария.

В качестве СУБД используйте *«MS Access»***,** отображаемую в выпадающем списке *«Database» («База данных»)* по умолчанию.

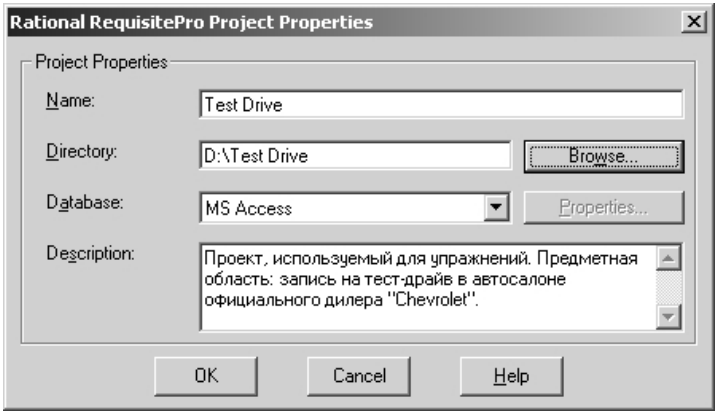


Рисунок 3. Создание репозитария. Окно свойств.

Нажмите кнопку *«Ok»***.**

Приложение отобразит диалоговое окно, содержащее предложение создать каталог файлов репозитария, как показано на рис. 4.

Рисунок 4. Создание каталога репозитария. Диалоговое окно.

Нажмите кнопку *«Да»***.**

Приложение отобразит модальное окно, информирующее о прогрессе создания репозитария. После завершения процесса содержимое окна имеет вид, приведенный на рис. 5.

Рисунок 5. Успешное создание репозитария. Модальное окно.

Нажмите кнопку *«Close»***.**

В результате выполнения перечисленных шагов произойдет запуск приложения. Рассмотрим основные элементы интерфейса *«IBM Rational RequisitePro»***.**

Основные элементы интерфейса. Интерфейс *«IBM Rational RequisitePro 7.1.1»***,** приведен на рис. 6. Рассмотрим основные элементы.

Панель инструментов предназначена для быстрого доступа к наиболее часто используемым командам приложения. При наведении курсора мыши на кнопку панели инструментов появляется подсказка, описывающая закрепленные за ней функции.

Проводник предназначен для навигации по репозитарию. Все основные артефакты (документы, требования, виды и пакеты) представлены в форме иерархического дерева. Проводник служит для выбора, просмотра и изменения артефактов репозитария. Поддерживается технология «drug and drop» для перемещения артефактов между пакетами. Окно проводника связано с другими окнами приложения, отражая изменения, вносимые в открытый документ, вид или требование.

Описание артефакта появляется при выборе артефакта в окне, расположенном под окном Проводника.

Пакеты содержат связанную с требованиями информацию. Пакет может содержать в себе: документы (упорядоченные в алфавитном порядке), виды, сортируемые по типу, а внутри типа - по алфавиту, и требования (упорядоченные по типу и признакам). Разработчику предоставляется возможность вкладывать один пакет в другой.

Представления или Виды предназначены для отображения требований. Все требования, их атрибуты и взаимосвязи с другими требованиями представляются и управляются при помощи представлений. Приложение предоставляет возможность строить запросы, фильтрующие и упорядочивающие требования и их атрибуты.

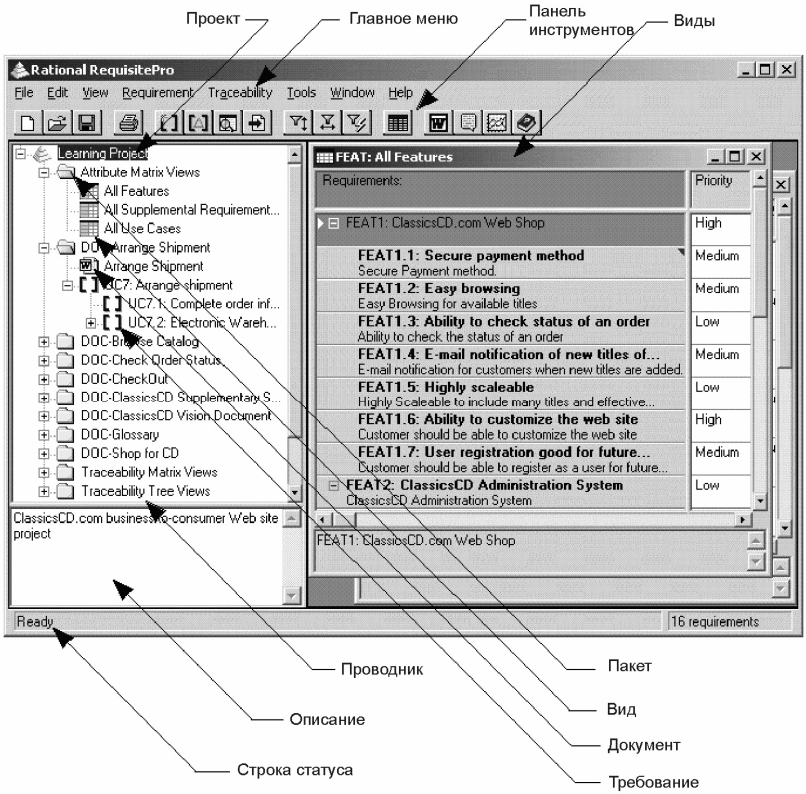


Рисунок 6. Интерфейс IBM Rational RequisitePro.

*Виды бизнес-правил.*В рамках цикла лабораторных работ необходимо определить структуру репозитария для хранения различных типов требований. Целесообразно начать с *бизнес-правил***,** поскольку именно этот тип требований выявляется на начальных этапах анализа требований.

«Бизнес-правило» - это положение, определяющее или ограничивающее какие-либо стороны бизнеса. Его назначение - защитить структуру бизнеса, контролировать или влиять на его операции.

Следует отметить, что существует множество различных схем классификации бизнес-правил на виды. Одна из них, предложенная Карлом Вигерсом, приведена на рис 7.

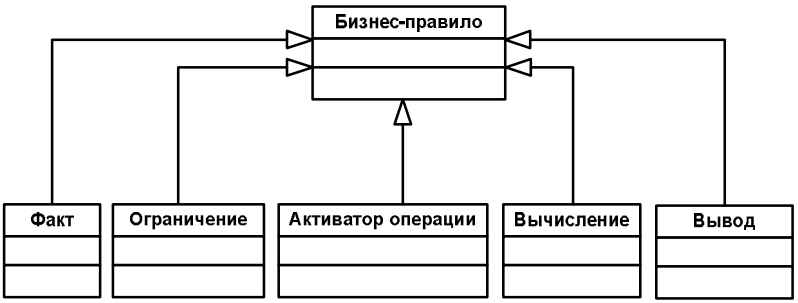


Рисунок 7. Виды бизнес-правил.

Определения для каждого из приведенных на рис. 7. видов бизнес-правил и примеры их формулирования приведены ниже.

Факты (facts) - это верные утверждения о бизнесе. Они описывают связи и отношения между важными бизнес-терминами. Факты также называют инвариантами - неизменными истинами о сущности данных и их атрибутах. Обычно факты напрямую не преобразуются в функциональные требования к системе. Сведения о сущности данных, важных для системы, применяют в моделях данных, создаваемых аналитиком или архитектором базы данных.

Пример: «Доставка заказа оплачивается клиентом».

Ограничения (Constraints) определяют, какие операции могут выполняться в рамках системы. Как правило, при формулировании ограничений используются слова и фразы вида: *должен / не должен, может / не может, только.*

Пример: «При отгрузке заказа менеджер должен запросить у бухгалтера товарно-транспортную накладную и счет-фактуру».

Активаторы операций (Action enabler) - правило, при определенных условиях приводящее к выполнению каких-либо действий. Выражение вида «Если <некоторое условие верно или наступило определенное событие>, то <что-то произойдет>», - это ключ, который описывает активатор операции.

Пример: «Если заказанный товар отсутствует на складе, то заказ передается в производственный отдел».

Вывод (Inference) - это правило, устанавливающее новые реалии на основе достоверности определенных условий. Вывод создает новый факт на основе других фактов или вычислений. Выводы зачастую записывают в формате «если — то», применяемом при записи активаторов. Однако раздел «то» вывода заключает в себе факт или предположение, а не действие.

Пример: «Если оплата по счету не поступила в течение 15 дней, заказ считается отменённым».

Вычисления (Computations) – вид бизнес-правил определяющий вычисления, выполняемые с использованием математических формул и алгоритмов. В отличие от активирующих операции бизнес-правил, для реализации которых иногда приходится создавать специфические функциональные требования, правила вычислений в той форме, в которой они выражены, можно рассматривать в качестве требований к программному обеспечению.

Пример:

«Если сумма заказа составляет от 50 тыс.руб. до 100 тыс.руб., скидка составляет 5%».

«Если сумма заказа составляет от 100 тыс.руб. до 200 тыс.руб., скидка составляет 10%».

«Если сумма заказа составляет свыше 200 тыс.руб., скидка составляет 15%».

Еще одной разновидностью бизнес-правил можно считать термины, которые согласно рекомендациям *«Rational Unified Process»*, следует хранить в отдельном документе под названием *«Глоссарий»***.**

Термины – важные для бизнеса слова, фразы и аббревиатуры.

Определение типа требований. Для хранения требований различных типов *«IBM Rational RequisitePro»* обладает достаточно гибким и развитым функционалом, который будет изучен по мере выполнения упражнений.

Пользователю предоставляется возможность использовать как типы требований, предопределенные в шаблоне, на основе которого создается репозитарий, так и определять собственные.

Каждый тип требований в *«IBM Rational RequisitePro»* имеет следующий атрибутный состав.

- *«Имя» («Name»)*, которое позволяет идентифицировать его в репозитарии. Имя типа является обязательным атрибутом, длина которого ограничена 64 символами.
- *«Описание» («Description»)***,** раскрывающее его семантику. Размер описания ограничен 256 символами.
- *«Начальный номер» («Initial requirement»)* используемый для нумерации требований данного типа. *«IBM Rational RequisitePro»* последовательно нумерует требования в рамках одного типа, по умолчанию начиная нумерацию с единицы. Ситуация, когда нумерацию потребуется начать с другой величины может возникнуть, например, в случае, когда репозитарий требований использует данные другого внешнего репозитария посредством внешней трассировки.
- *«Разрешить внешнюю трассировку…» («Allow External Traceability»)* – признак, разрешающий или запрещающий требованиям данного типа быть связанными отношением трассировки с требованиями, физически хранимыми в других репозитариях *«IBM Rational RequisitePro»***.**
- *«Требование должно содержать…» («Requirement Must Contain»)* – необязательное текстовое поле длиной до 32 символов. Используется для указания слова или фразы, которая должна содержаться в тексте требований данного типа. Приложение не запрещает создавать требования, не содержащие выражения, указанного в данном поле. Контроль со стороны *«IBM Rational RequisitePro»* сводится к выдаче предупреждающих сообщений всякий при создании требований, не содержащих указанной фразы. Например, хорошо сформулированные функциональные требования к системе рекомендуется создавать с использованием словосочетания «система должна…», а активатор операции со слова «Если…» (см. раздел 2.6). Если вписать текст в данное поле, приложение будет ожидать, что словосочетание будет встречаться в тексте каждого требования данного

типа и выдавать предупреждения при его отсутствии. При поиске подстроки в тексте требования приложение игнорирует регистр, использованный разработчиком при воде текста в данное поле.

- *«Префикс требования» («Requirement Tag Prefix»)* – обязательное поле длиной до 20 символов. Для простоты идентификации требований различных типов, *«IBM Rational RequisitePro»* предлагает разработчику использовать префиксы, представляющих собой аббревиатуру типа требования. Например, для *требований к программному обеспечению (software requirement)* логично использовать аббревиатуру *SR***,** для *требований прецедентов (use case requirements) – UC*, для *высокоуровневых функциональных требований к системе*, также называемых возможностями *(features) – FEAT*.
- *«Цвет требования» («Requirement Color»)* **-** выпадающий список, содержащий перечень наименований цветов, которыми требования данного типа будут выделяться в документе *«Microsoft Word»***.** Целесообразно использовать различные цвета для различных типов требований при условии, что они хранятся в одном документе. Это позволяет визуально идентифицировать их на экране монитора при просмотре документов, не сосредотачиваясь на префиксе (который также предназначен для идентификации требований). Если требования разных типов хранятся в разных документах, целесообразно использовать цвет по умолчанию, которым является синий (blue).
- *«Стиль требования» («Requirement Style»)* **-** выпадающий список, содержащий предопределенные стили для отображения требований данного типа в документе *«Microsoft Word»***.** По умолчанию имя требования, хранимого в документе, выделяется подчеркиванием (Double Underline), что в совокупности с синим цветом делает имя требования похожим на гиперссылку. Следует отметить, что фактически она таковой и является: нажатие левой кнопкой мыши на имени требования в документе *Microsoft Word* вызывает отображение атрибутов требования, хранимых в базе данных *«IBM Rational RequisitePro»* (будет рассмотрено позже)**.**

Создание структуры репозитария начинается с определения типов требований. В рамках следующего упражнения будет создан новый тип требований: бизнес-правила.

*Упражнение. Определение типа требований***.**

1. Для доступа к окну *«Project Properties» («Свойства проекта»)* выберите пункты меню *File / Project Administrator / Properties (Файл / Администратор проекта / Свойства)***.**
2. Приложение отобразит окно, показанное на рис. 8.
3. Выберите вкладку *«Requirement Types» («Типы требований»)***.**

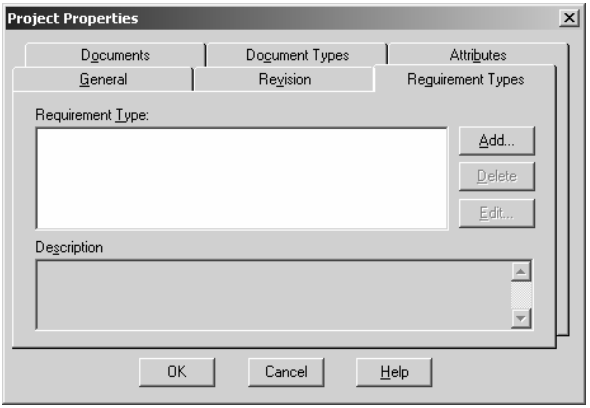


Рисунок 8. Окно свойств репозитария.

4. Для того чтобы добавить новый тип требований, нажмите на кнопку *«Add…» («Добавить…»)*.
5. Приложение отобразит диалоговое окно *«Requirement Type» («Тип требования»)*.
6. Заполните поля, как показано на рис. 9: в поле *«Name»* введите имя типа: «Бизнес-правило», в поле *«Description»* введите описание типа: «Положение, определяющее или ограничивающее какие-либо стороны бизнеса. Его назначение - защитить структуру бизнеса, контролировать или влиять на его операции». В качестве префикса, вводимого в поле *«Requirement Tag Prefix»* используйте аббревиатуру «BR» (от английского «Business Rule»). Остальные поля оставьте без изменений.

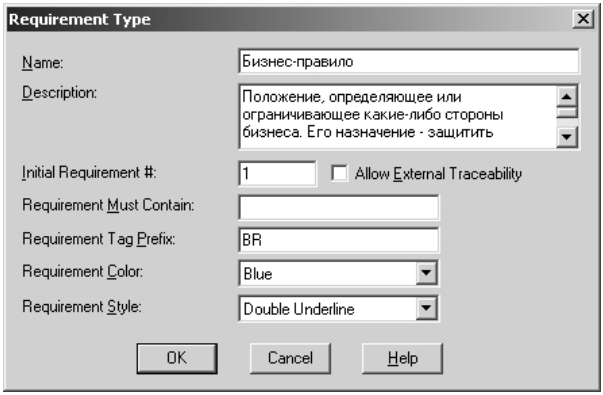


Рисунок 9. Окно создания нового типа требований.

7. Нажмите кнопку *«Ok»*.
8. Среда отобразит окно свойств репозитария, показанное на рис. 10.

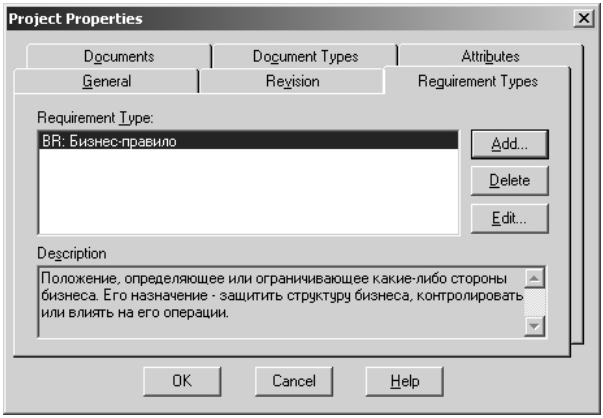


Рисунок 10. Окно свойств репозитария после добавления нового типа требований.

Определение атрибутов требований. С каждым типом требований приложение связывает набор атрибутов, который можно разделить на две основные категории:

- системные атрибуты;
- атрибуты, определяемые разработчиком репозитария.

При определении атрибутов требований в окне *«Project Properties» («Свойства проекта»)***,** которое показано рис. 11, приложение не отображает системные атрибуты, поскольку они не могут быть изменены разработчиком репозитария.

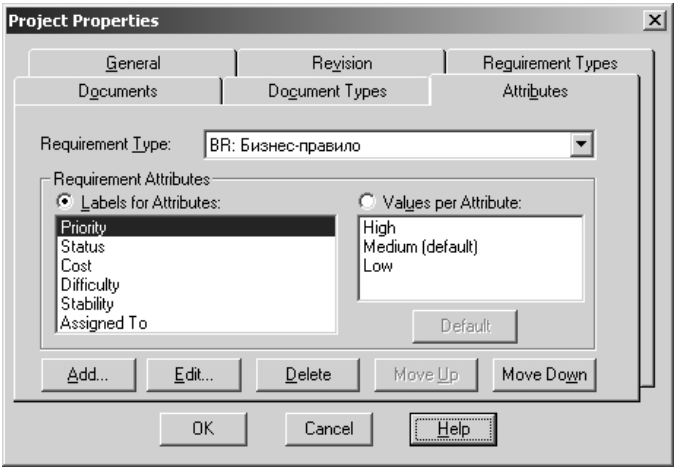


Рисунок 11. Окно свойств репозитария, вкладка «Attributes».

Системными атрибутам требования являются:

- *«Имя требования» («Name»)*;
- *«Текст требования» («Text»)*;
- *«Уникальный идентификатор требования» («Unique ID»)*;
- *«Местоположение» («Location»)*;
- *«Автор» («Author»)*.

В свою очередь атрибуты, определяемые разработчиком репозитария, делятся на две категории:

- атрибуты спискового типа;
- атрибуты со свободным вводом значения.

Полная классификация атрибутов требований приведена на рис. 12.

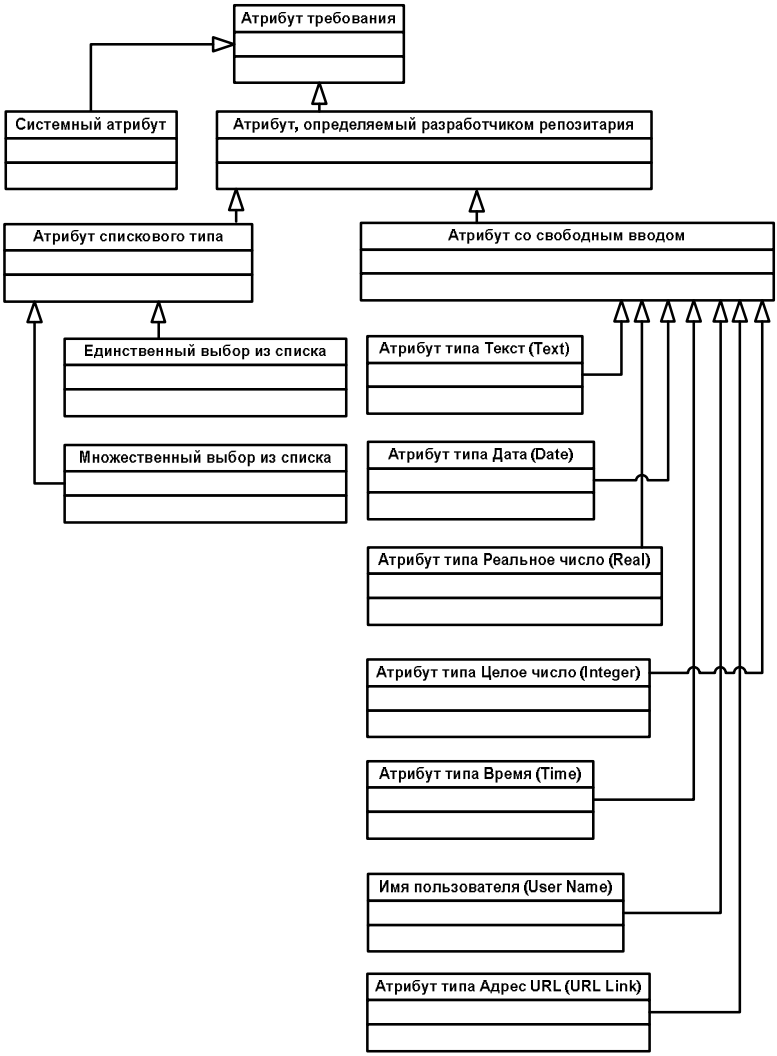


Рисунок 12. Классификация атрибутов требований «IBM Rational RequisitePro»

При создании нового типа требования *«IBM Rational RequisitePro»* предлагает использовать определенный по умолчанию набор атрибутов, определяемых разработчиком репозитария.

Рассмотрим содержимое окна *«Project Properties» («Свойства проекта»)*, показанное на рис. 11. Оно отображается на экране при выборе вкладки *«Attributes» («Атрибуты»)*.

Логику работы окна проще понять, если разделить его на три области, которые изменяются в зависимости от типа атрибута, определяемого разработчиком репозитария.

- *«Labels for Attributes» («Метки атрибутов»)* – область меток атрибутов, представленная в виде списка в левой части окна. Двойное нажатие левой кнопкой мыши приводит к открытию окна редактирования метки атрибута, показанного на рис. 13.
- *Values per Attribute» («Значения атрибута»)* – область, содержащая список возможных значений для атрибута спискового типа расположена в правой части окна. Область появляется при выборе определенного значения в списке *«Метки атрибутов»* Примером атрибута спискового типа является атрибут *«Priority» («Приоритет»)***.** Двойное нажатие левой кнопки мыши приводит к открытию окна редактирования значений атрибута, показанного на рис. 14.
- *«Default» («Значение по умолчанию»)* – третья область окна, представленная единственной кнопкой *«Default».* Она позволяет быстро назначить или отменить значение по умолчанию для атрибута спискового типа.

Редактирование атрибутов спискового типа производится с использованием окна, показанного на рис 13. Вызов окна осуществляется двойным нажатием левой кнопки мыши на элементе списка *«Labels for Attribute» («Метки атрибутов»)* либо путем нажатия кнопки *«Edit…» («Редактировать»)* (при условии предварительного выбора значения в списке *«Labels for Attribute» («Метки атрибутов»))*

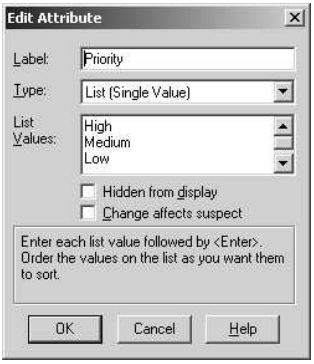


Рисунок 13. Окно редактирования атрибута спискового типа.

Использование приведенного на рис. 13 окна редактирования атрибута является особенно эффективным, если разработчику репозитария необходимо изменить метку атрибута, тип или содержимое списка возможных значений.

Если необходимо изменить только набор возможных значений для атрибута спискового типа, целесообразным может оказаться вызов окна *«Attribute Value» («Значение атрибута»)***,** показанного на рис 14.

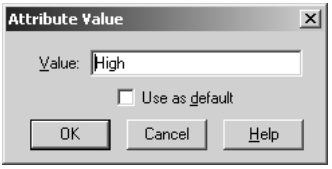


Рисунок 14. Окно редактирования значений атрибута.

Вызов окна осуществляется двойным нажатием мыши на элементе списка *«Values per Attribute» («Значения атрибута»).* По понятным причинам указанный метод изменения списка возможных значений доступен только для атрибутов спискового типа: для атрибутов со свободным вводом список *«Values per Attribute» («Значения атрибута»)* является недоступным.

*Определение атрибутов бизнес-правил.*Стандартный набор атрибутов, предлагаемых *«IBM Rational RequisitePro»* для требований, определяемых разработчиком репозитария, приведен на рис. 11. Этот набор атрибутов применяется для описания функциональных требований к системе, но не характерен для бизнес-правил. Если при создании репозитария использовать шаблон *«Business Modeling Template»* **(***«Шаблон для моделирования организационной системы»***)**, репозитарий будет содержать такой тип требований, как *«Business Rule» («Бизнес-правило»)*. С точки зрения разработчиков шаблона, бизнес-правилам не присущи атрибуты, поэтому при выборе вкладки *«Attributes» («Атрибуты»)* окна *«Project Properties» («Свойства проекта»)* можно видеть пустой список *«Labels for Attributes» («Метки атрибутов»),* как показано на рис. 15.

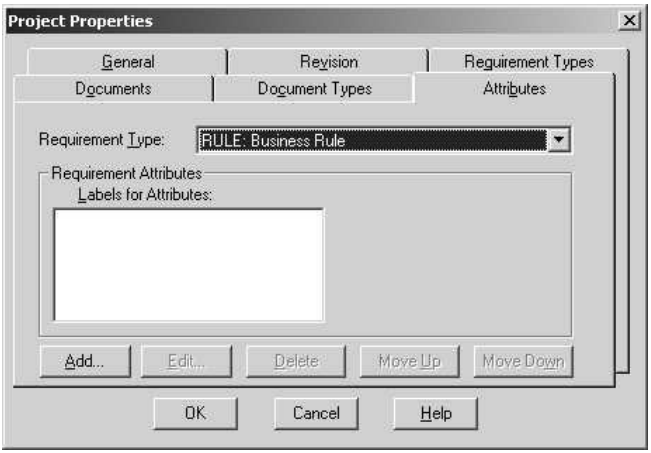


Рисунок 15. Атрибуты бизнес-правил при использовании шаблона «Business Modeling Template»

В результате разработчику репозитария доступны для определения значения атрибутов *«Name» («Имя требования»)* и *«Text» («Текст требования»)*, которые являются системными. Присваивание значений этим атрибутам осу-

ществляется при помощи *«Attribute Matrix» («Матрицы атрибутов»)*, как показано на рис. 16.

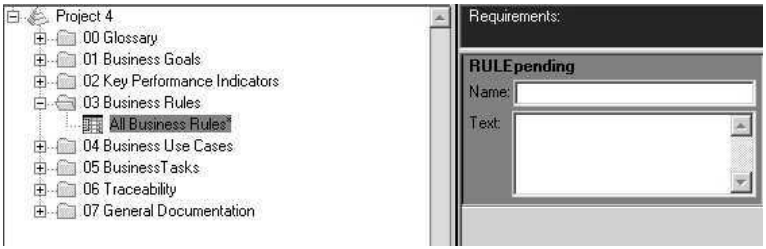


Рисунок 16. Присваивание значений атрибутам «Имя» и «Текст требования» бизнес-правил, содержащихся в «Business Modeling Template» при помощи «Матрицы атрибутов»

Помимо упомянутых выше атрибутов бизнес-правил *«Имя»* и *«Текст требования»* в рамках следующего упражнения будет определен еще один атрибут с меткой *«Вид бизнес-правила»*.

Этот атрибут будет использован для разбиения бизнес-правил на виды согласно классификации, приведенной выше. Сведения о том, как должно формулироваться бизнес-правило определенного вида, призваны побудить разработчика репозитария формулировать их более точно и строго.

Упражнение. Определение атрибутов для бизнес-правил.

1. Выберите вкладку *«Attributes» («Атрибуты»)* окна *«Project Properties» («Свойства проекта»)*.
2. Приложение отобразит окно, показанное на рис. 11.
3. Выделите элемент списка *«Labels for Attribute» («Метки атрибутов»)* и, нажмите кнопку *«Delete» («Удалить»)*.
4. В ответ среда отобразит модальное диалоговое окно, содержащее запрос на получение эксклюзивного доступа к репозитарию, показанное на рис. 17.

Рисунок 17. Диалоговое окно, содержащее запрос на получение эксклюзивного доступа к репозитарию.

5. Ответьте утвердительно, нажав кнопку *«Да»*.
6. Среда предоставит эксклюзивный доступ к репозитарию, отобразив модальное окно, показанное на рис. 18.

Рисунок 18. Модальное окно, уведомляющее о получении эксклюзивного доступа к репозитарию.

7. Среда удалит первый элемент списка *«Labels for Attribute»*.
8. Очищайте список *«Labels for Attribute»*, последовательно выделяя элементы до тех пор, пока окно не примет вид, показанный на рисунке 19.

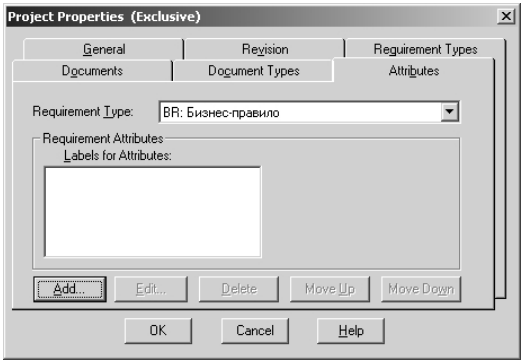


Рисунок 19. Пустой список атрибутов.

9. Нажмите кнопку *«Add…» («Добавить»)*.
10. Среда выведет окно *«Add Attribute» («Добавить атрибут»)*, показанное на рис. 20.
11. Присвойте имени атрибута бизнес-правила значение *«Вид бизнес-правила»*, введя соответствующий текст в поле *«Label» («Метка»)*.
12. В выпадающем списке *«Type» («Тип»)* выберите значение *«List (Single Value)» («Список с единственным вариантом выбора»)*.
13. В области *«List Values» («Список значений»)* введите имена пяти видов: *«Факт», «Ограничение», «Активатор операции», «Вывод», «Вычисление»*. Используйте клавишу *«Ввод»*, чтобы разделить виды правил между собой.
14. Убедитесь, что содержимое окна соответствует содержимому, показанному на рис. 20.

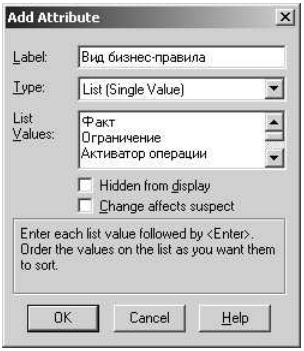


Рисунок 20. Добавление атрибута «Вид бизнес-правила» и определение списка его возможных значений.

15. Нажмите кнопку *«Ok»***.**
16. Среда закроет окно добавления атрибута и отобразит окно *«Project Properties»* с активной вкладкой *«Attributes»***.**
17. Выделите в списке *«Values per Attribute» («Значения для атрибутов»)* значение *«Факт»* и нажмите кнопку *«Default»*.
18. Нажмите кнопку *«Ok»***,** чтобы закрыть окно *«Project Properties» («Свойства проекта»)* и сохранить сделанные изменения.

Создание требований посредством матрицы атрибутов. Представления в *«IBM Rational RequisitePro»* служат для отображения результатов запросов к базе данных, содержащей требования. Представления содержат требования одного типа и их атрибуты либо требования различных типов и возникающие между ними отношения трассировки.

«IBM Rational RequisitePro» создает представления, основываясь на типах требований, определенных разработчиком репозитария. Приложение поддерживает три вида представлений:

- *«Attribute Matrix» («Матрица атрибутов»);*
- *«Traceability Matrix» («Матрица трассировки»);*
- *«Traceability tree» («Дерево трассировки»).*

Рассмотрим интерфейс, предоставляемый *«IBM Rational RequisitePro»* для создания представления, показанный на рис. 21.

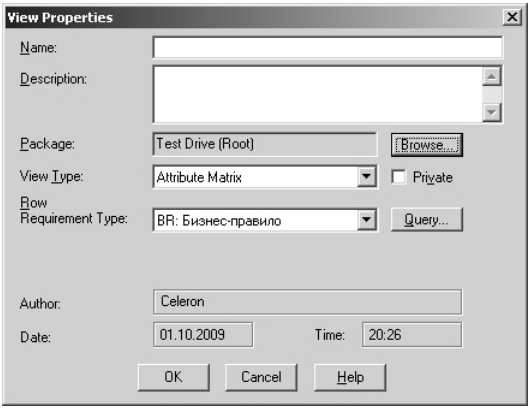


Рисунок 21. Окно свойств представления.

Помимо уникального имени и описания, вводимых в поля *«Name» («Имя»)* и *«Description» («Описание»)* соответственно**,** представление характеризуется следующими атрибутами:

- *«View Type» («Тип представления»)* который может принимать одно из трех перечисленных выше значений;
- *«Package» («Пакет»)* - имя пакета, в котором расположено представление;
- *«Private» («Личное»)* – признак того, что представление будет доступно только тому пользователю приложения, который его создал;
- *«Row Requirement Type» («Тип требования, располагаемый в строке»)* – атрибут со списком возможных значений, который образуют типы требований, определенные разработчиком репозитария;
- *«Query» («Запрос»)* – атрибут, содержащий запрос. Он позволяет отображать в представлении требования, удовлетворяющие определенным условиям.
- *«Column Requirement Type» («Тип требования, располагаемый в столбце»)* – атрибут доступный только для матрицы трассировки.
- *«Author» («Автор»)* **- атрибут, содержащий имя пользователя** *«IBM Rational RequisitePro»*, который последним модифицировал представление.

Упражнение. Создание требований посредством матрицы атрибутов.

1. Выберите в проводнике корневой узел репозитария, имеющий имя *«Test Drive»***.**
2. Нажмите правую кнопку мыши.
3. Приложение отобразит контекстное меню, показанное на рис. 22.

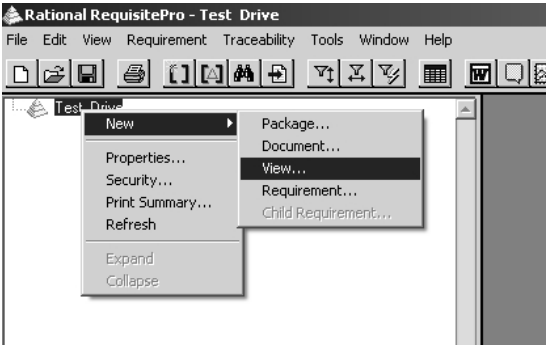


Рисунок 22. Пустой список атрибутов.

4. Выберите пункты *«New» / «View…» («Новое» / «Представление»)*.
5. Присвойте представлению имя *«Список бизнес-правил проекта»*, введя соответствующий текст в поле *«Name» («Имя»)*.
6. В поле *«Description» («Описание»)* введите следующий текст *«Представление, отображающее все бизнес-правила проекта «Запись на тест-драйв»*.
7. Убедитесь, что в выпадающем списке *«Row Requirement Type» («Тип требования, располагаемый в строке»)* отображается значение *«BR: Бизнес-правило»*.
8. Убедитесь, что окно имеет вид, приведенный на рис. 23.

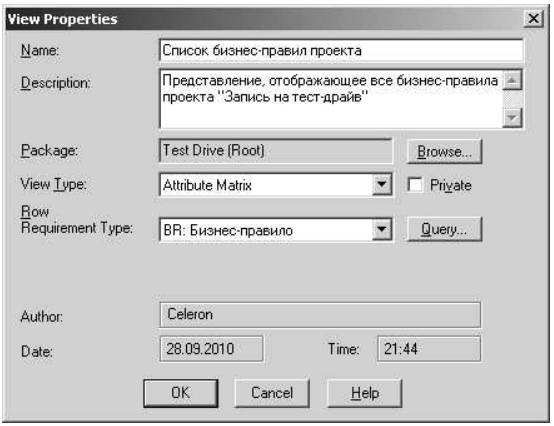


Рисунок 23. Окно создания представления «Список бизнес-правил проекта».

9. Закройте диалоговое окно, нажав на кнопку *«Ok»*.
10. Приложение создаст новое представление *«Список бизнес-правил проекта»* и сделает его активным.
11. Полученное представление содержит наименования атрибутов бизнес-правил, которые представлены заголовками столбцов. По умолча-

нию первыми слева располагаются столбцы атрибутов, определенных разработчиком репозитария. За ними следуют системные атрибуты, см. раздел Определение атрибутов требований. Системные атрибуты не являются важными для работы, их целесообразно скрыть.

12. Нажмите правой кнопкой мыши на заголовке первого столбца с именем системного атрибута, по умолчанию им является *«Unique ID» («Уникальный идентификатор требования»)*.

13. В появившемся контекстном меню выберите пункт *«Hide from View» («Скрыть в представлении»),* как показано на рис. 24 и нажмите левой кнопкой мыши.

Рисунок 24. Скрытие атрибутов требований в матрице атрибутов.

14. Приложение перестанет отображать столбец.
15. Повторите шаги 12-13 для всех системных атрибутов.
16. Нажмите пиктограмму дискеты, расположенную на панели инструментов для сохранения в репозитарии информации о том, как должна выглядеть матрица атрибутов. Обратите внимание: в результате данной операции приложение не сохраняет все сведения о проекте. Сохраняются только данные текущего представления.

17. Для добавления нового требования нажмите левой кнопкой мыши на первой строке столбца *«Requirements» («Требования»)*.

18. Приложение отобразит форму ввода нового требования, показанную на рисунке 25.

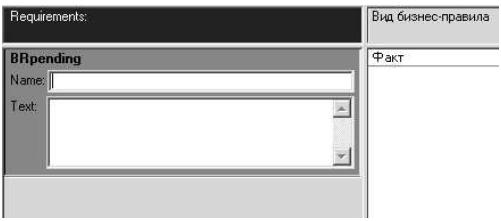


Рисунок 25. Форма ввода нового требования.

19. В поле *«Name» («Имя»)* введите *«Тест-драйв в часы работы салона»*

20. В поле *«Text» («Текст»)* введите *«Тест-драйв может осуществляться только в часы работы салона»*.

21. Нажмите левой кнопкой мыши в любом месте за пределами формы.

22. Приложение присвоит только что созданному бизнес-правилу уникальный идентификатор *«BR1:»*, состоящий из префикса и порядкового номера.
23. Присвойте атрибуту *«Вид бизнес-правила»* значение *«Ограничение»*, нажав левой кнопкой мыши в соответствующей ячейке и выбрав это значение из выпадающего списка.
24. Повторите шаги 16-22 для всех бизнес-правил репозитария, корректно определяя их вид.
25. Закройте приложение, сохранив репозитарий с результатами работы.

Последовательность проведения занятия:
1. Получить у преподавателя вариант задания (предметную область).
2. Создать проект на основе чистого шаблона.
3. Определить типы требований.
4. Определить атрибуты требований.
5. Определить бизнес-правила для проекта.
6. Определить атрибуты для бизнес-правил.
7. Создать требования посредством матрицы атрибутов.
8. Защитить работу перед преподавателем.

Вопросы для самоконтроля
1. Дайте определение проекта.
2. Дайте определение требования.
3. Перечислите существующие типы требований.
4. Перечислите основные атрибуты требований.

6. Список литературы

1. Карл И. Вигерс. Разработка требований к программному обеспечению./Пер. с англ. – М.:Издательско-торговый дом «Русская редакция», 2004. – 576 с.

2. Буч Г. Введение в UML от создателей языка. – ДМК-Пресс, 2012 – 496 с.

3. Леоненков А. В. Объектно-ориентированный анализ и проектирование с использованием UML и IBM RATIONAL ROSE : Учебное пособие для вузов / А. В. Леоненков. - М. : Интернет-Университет Информационных Технологий, 2006 ; М. : БИНОМ. Лаборатория знаний, 2006. - 318[2] с.: ил.

4. Леоненков А. В. Самоучитель UML / А. В. Леоненков. - 2-е изд. - СПб. : БХВ-Петербург, 2006. - 427[5] с. : ил.

5. Грекул В. И. Проектирование информационных систем. Курс лекций : Учебное пособие для вузов / В. И. Грекул, Г. Н. Денищенко, Н. Л. Коровкина. - М. : Интернет-Университет Информационных Технологий, 2005. - 298[5] с. : ил.

6. Ларман К. Применение UML и шаблонов проектирования. Введение в объектно-ориентированный анализ, проектирование и унифицированный

процесс UP : Пер. с англ. / К. Ларман ; ред. пер., пер. А. Ю. Шелестов. - 2-е изд. - М. : Вильямс, 2002. - 619[5] с. : ил.

7. Бабич А. В. UML: Первое знакомство. Пособие для подготовки к сдаче теста UMO-100 (OMG Certified UML Professional Fundamental) : учебное пособие / А. В. Бабич. - М. : БИНОМ. Лаборатория знаний, 2008. - 175, [1] с. : ил.

8. Максимчук Р. UML для простых смертных : пер. с англ. / Р. А. Максимчук, Э. Дж. Нейбург ; пер. М. Ц. Горелик. - М. : ЛОРИ, 2008. - XXXII, 268 с. : ил.

9. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на C++ : Пер. с англ. / Гради Буч; Ред. пер. И. Романовский, Ред. пер. Ф. Андреев. - 2-е изд. - М. : БИНОМ, 2000 ; СПб. : Невский Диалект, 2000. - 360 с. : ил

10. Орлов С. А. Технологии разработки программного обеспечения. Разработка сложных программных систем : Учебное пособие для вузов / Сергей Александрович Орлов. - СПб. : Питер, 2002. - 464 с. : ил.