ТЕМА 14

1. Определение и функциональные возможности Rational Unified Process.
2. Статические и динамические аспекты RUP.
3. Создание артефактов в RUP.
4. Создание программных средств в RUP.

14.1 Определение Rational Unified Process

Корпорация *Rational Software*, ведущий производитель программных продуктов для создания сложных программных систем, формализовала технологический процесс разработки ПО и выпустила на рынок структурированную базу знаний под названием *Rational Unified Process* (RUP). В нее вошли методические рекомендации ведущих разработчиков ПО по эффективному созданию приложений и программных систем. *Rational Unified Process – это методология создания ПО, оформленная в виде размещаемой на Web базы знаний, которая снабжена поисковой системой.*

База знаний RUP регулярно обновляется с целью учета передового опыта и улучшается за счет проверенных на практике результатов. *Rational Unified Process* создан в виде страниц формата HTML, имеющих обширную систему гиперссылок, графическую навигацию, подробное оглавление и встроенный поисковый механизм. На рис. 14.1 показано, как выглядит главное окно RUP. *Rational RUP v2002* поддерживает технологию разработки для различных платформ, предоставляет детальные рекомендации как для перехода команды разработчиков к технологии разработки на платформе Microsoft .NET, так и для тех, кто не собирается переходить к платформе .NET.

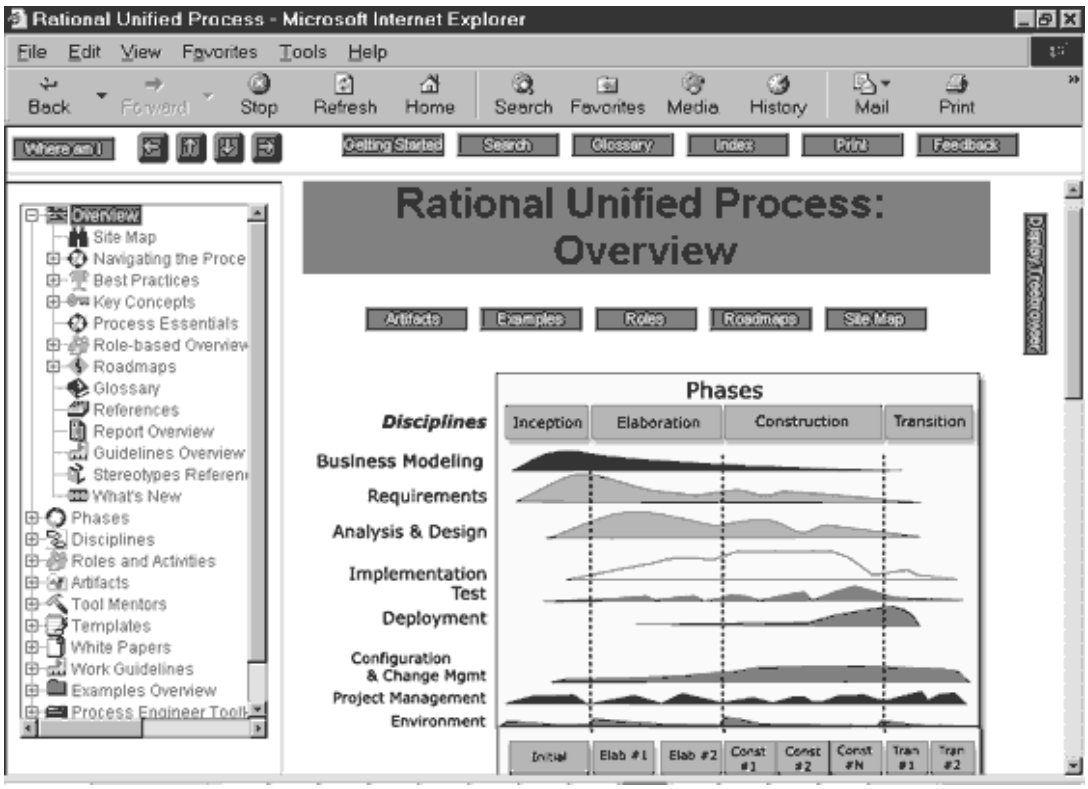


Рисунок 14.1 – Главное окно RUP

Rational Unified Process ведет свою историю от Rational Approach и Objectory Process 3.8, объединение которых произошло после слияния в 1995 году корпорации Rational Software Corporation и Objectory AB. В последних версиях продукта производители разделили технологические процессы для различных видов программных проектов, например, для разработки программ электронной коммерции. Разработчики RUP не забыли и о небольших проектах, включив в v2002 полный пакет рекомендаций для создания проектов командой от трех до десяти человек. Эти рекомендации предложены в виде подробного описания пути разработки малых проектов и снабжены примерами возможных конфигураций таких проектов. Поскольку каждый программный проект по-своему уникален, то хотя в основу RUP и положены рекомендации по всем стадиям и фазам разработки программ, не для каждого проекта они подходит на сто процентов. Но в любом случае, применение унифицированного процесса разработки позволит уменьшить затраты проекта, уложиться в сроки и повысить качество создаваемого программного продукта. RUP обеспечивает строгий подход к распределению задач и ответственности внутри организации-разработчика. Его цель заключается в том, чтобы гарантировать создание качественного ПО точно в срок и в рамках установленного бюджета, отвечающего требованиям конечных пользователей. RUP способствует повышению производительности коллективной разработки, предоставляя команде набор руководств, шаблонов и наставлений по пользованию инструментальными средствами на всех этапах ЖЦ создания и сопровождения ПО. RUP предоставляет каждому члену команды доступ к общей базе знаний

независимо от того, на каком этапе разработки он задействован. Кроме того, RUP предоставляет возможность всем участникам проекта использовать общий язык моделирования UML, являющийся международным стандартом. Поэтому еще RUP можно рассматривать как руководство по эффективному использованию UML.

В результате работы над проектом с помощью RUP вместо создания громадного количества бумажных документов создаются и совершенствуются компьютерные модели. RUP поддерживается инструментальными средствами, автоматизирующими большинство этапов ЖЦ ПО. RUP – это конфигурируемый процесс, поскольку невозможно создать единого руководства на все случаи разработки ПО. RUP пригоден как для маленьких групп разработчиков, так и для больших организаций, занимающихся созданием ПО. В основе RUP лежит простая и понятная архитектура процесса, которая обеспечивает общность для целого семейства процессов. Более того, RUP может конфигурироваться для учета различных ситуаций. В его состав входит Development Kit, который обеспечивает поддержку процесса конфигурирования под нужды конкретных организаций. RUP описывает, как эффективно применять коммерчески обоснованные и практически опробованные подходы к разработке ПО для коллективов разработчиков, где каждый из членов получает преимущества от использования передового опыта в решении следующих вопросов:

- итерационная разработка ПО;
- управление требованиями;
- использование компонентной архитектуры;
- визуальное моделирование;
- тестирование качества ПО;
- контроль за изменениями в ПО.

14.2 Аспекты RUP

RUP организует работу над проектом в терминах статических и динамических аспектов процесса. К статическим аспектам относятся последовательности действий, продуктов деятельности, исполнителей, а к динамическим – циклы, фазы, итерации и временные отметки завершения определенных этапов в создании ПО. В графическом виде процесс можно представить следующим образом. Вдоль горизонтальной оси будем откладывать его динамические аспекты, а вдоль вертикальной оси – статические аспекты процесса. Тогда результат будет выглядеть, как показано на рис. 14.2

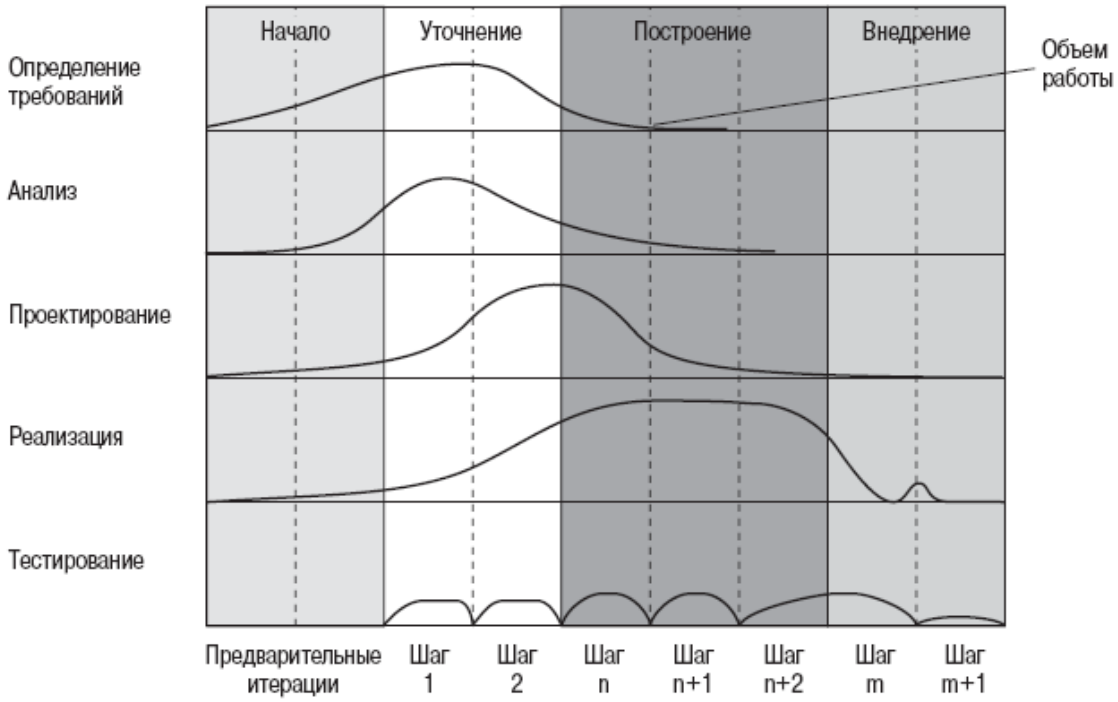


Рисунок 14.2 – Графическое представление аспектов RUP

Каждая из фаз процесса разработки состоит из нескольких итераций, целью которых является последовательное осмысление стоящих проблем, наращивание эффективных решений и снижение риска потенциальных ошибок в проекте. В то же время, каждая из последовательностей действий по созданию ПО выполняется в течение нескольких фаз, проходя пики и спады активности. Каждый цикл итерации проекта начинается с планирования того, что должно быть выполнено. Результатом выполнения должен быть качественный продукт. Заканчивается же цикл оценкой того, что было сделано, и были ли достигнуты цели.

В основу Rational Unified Process положена база знаний, размещаемая на Web-сервере и состоящая из руководств, шаблонов, наставлений по использованию инструментальных средств. База знаний может быть разбита на следующие компоненты:

1. Обширные руководства для всех членов коллектива разработчиков, для каждого временного интервала жизненного цикла ПО. Руководства представлены в двух видах: для осмысления процесса на верхнем уровне и в виде подробных наставлений для повседневной деятельности. Руководства опубликованы в HTML формате.
2. Опубликованы в HTML формате наставления по пользованию инструментальными средствами, автоматизирующими процесс создания ПО.
3. Примеры и шаблоны, которые служат руководствами по тому, как структурировать информацию в Rational Rose, следуя указаниям RUP.
4. Шаблоны для SoDa, помогающие автоматизировать документирование ПО.

5. Шаблоны Microsoft Word, предназначенные для поддержки документирования всех действий по всем интервалам жизненного цикла ПО.

На сегодняшний день тысячи компаний, создающих ПО, воспользовались преимуществами разработки на основе RUP. Унифицированный процесс разработки используется в различных прикладных областях для больших и малых проектов, что доказывает его универсальность и широкую применимость. В телекомунникационной сфере его используют Ericsson, Alcatel, MCI, в авиационно-космической и оборонной промышленностях – Lockheed-Martin, British Aerospace, на промышленных предприятиях – Xerox, Volvo, Intel, в финансовой сфере – Visa, Merrill Lynch, Schwab. Этот процесс также взят на вооружение такими интеграторами, как Ernst & Young, Oracle, Deloitte & Toucche. На рис. 14.3 приведено окно RUP, предназначенное для анализа решаемой проблемы. Каждая из компаний использует RUP по-своему. Одни строго следуют всем рекомендациям Rational Software, другие создают на его основе свои методики, используя для этого базу знаний RUP.

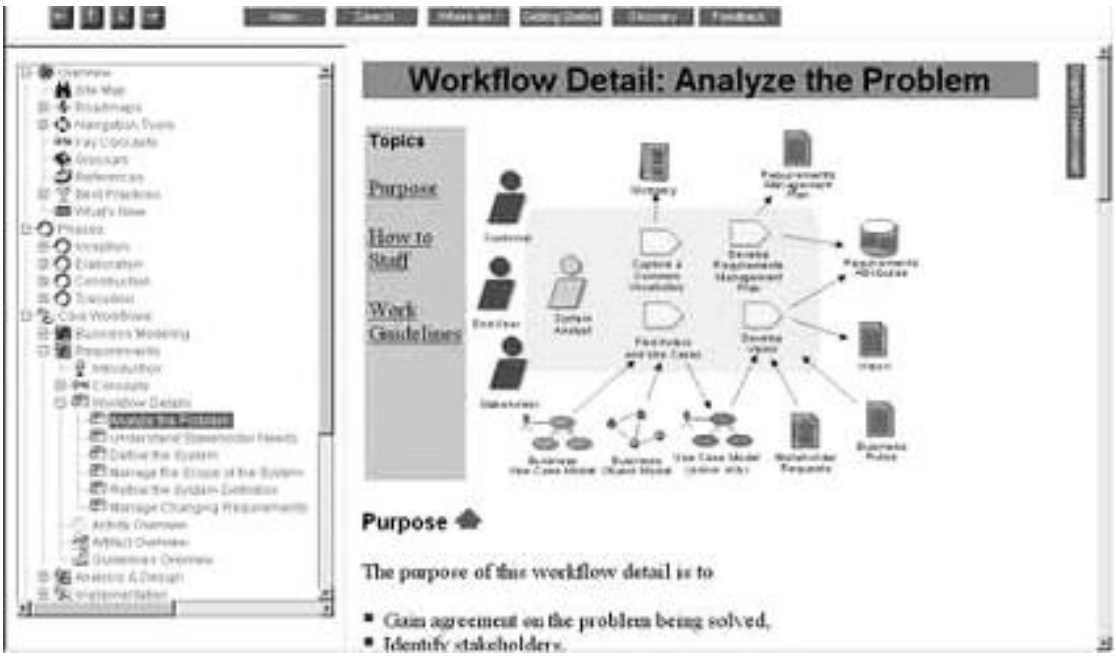


Рис. 14.3 Анализ проблемы в RUP

Основными понятиями RUP являются *артефакт* и *прецедент*. Артефакты – это некоторые продукты проекта, порождаемые или используемые в нем при работе над окончательным продуктом. Прецеденты – это последовательности действий, выполняемых системой для получения результата. Весь процесс разработки программной системы рассматривается в RUP как процесс создания артефактов. Причем то, что попадает в руки конечного пользователя, будь то программный модуль или программная документация, – это один из подклассов всех артефактов проекта. Каждый член проектной группы создает свои артефакты и несет за них

ответственность. Программист разрабатывает программу, руководитель – проектный план, аналитик – модели системы. RUP позволяет определить когда, кому и какой артефакт необходимо создать. Основной упор в RUP делается не на подготовку документов как таковых, а на моделирование разрабатываемой системы. Модели помогают очерчивать как проблему, так и пути ее решения. Они создаются на языке UML, который позволяет разработчикам определять, визуализировать, конструировать и документировать артефакты программных систем.

Все задачи, описанные в RUP, поддерживаются средствами разработки от Rational Software. В RUP включен раздел Tool Mentors, в котором подробно описывается создание UML-моделей, а также работа с продуктами Rational RequisitePro, Rational Clear Quest, Rational Clear Case, Rational Purify, Rational PureCoverage, Rational Quantify, Rational Robot, Rational SoDa. Кроме того, Tool Mentors содержит подробные рекомендации по использованию этих и других средств компании Rational для создания конкретных артефактов разрабатываемой системы.

Весь процесс разработки с точки зрения RUP рассматривается в двух плоскостях. В динамике процесс выражается через циклы, фазы, итерации и вехи, а в статике – через виды деятельности, технологические процессы, артефакты и роли исполнителей. Каждый такой процесс представляется при помощи диаграмм, состоящих из пиктограмм, связанных гиперссылками с другими документами. При активизации гиперссылок происходит детализация процесса, что дает возможность пройти всю последовательность необходимых действий: от общего взгляда на них до создания конкретных артефактов. Система гиперссылок построена таким образом, что можно легко переходить от работ к артефактам, создаваемым в процессе конкретной деятельности, а через них к ролям исполнителей и обратно. К артефактам можно добраться различными путями: через список процессов, примеры итераций, роли исполнителей. Кроме того, можно найти нужный артефакт в дереве ссылок, что позволяет рассматривать один и тот же процесс разработки с различных точек зрения: руководителя, исполнителя, пользователя, программиста.

Основные артефакты, создаваемые в процессе разработки, представлены в RUP в виде готовых шаблонов, облегчающих их создание в конкретном проекте. В базу знаний включены шаблоны более тридцати видов документов, наиболее распространенные типы отчетов представлены в форматах MS Word и Adobe FrameMaker. Шаблоны Rational SoDa позволяют автоматизировать процесс сбора документов из множества источников, а шаблоны RequisitePro облегчают управление требованиями. В помощь специалистам, занимающимся планированием итеративных проектов на основе RUP, в него включены шаблоны Microsoft Project, имеются шаблоны формата HTML, позволяющие расширять базу знаний.

Ответственность за создание артефактов лежит на исполнителях. В последних версиях RUP чаще применяется понятие роль, поскольку один исполнитель может выполнять несколько ролей в проекте и отвечать за

различные артефакты. В RUP определено более тридцати ролей, которые могут выполнять различные члены команды разработчиков. Обязанности каждой роли, последовательность работ и создаваемые артефакты представлены в виде диаграмм. Каждая пиктограмма на диаграмме представляет собой гиперссылку, позволяющую проводить дальнейшую детализацию.

Поскольку каждый программный проект по-своему уникален, нельзя бездумно копировать чужой процесс, создавая артефакты, имеющие незначительную ценность. Поэтому рекомендуется использовать RUP как основу, для дальнейшего уточнения, расширения и специфического настраивания с целью максимального ее приближения к нуждам организации-разработчика. Однако в любом случае применение унифицированного процесса разработки позволит уменьшить затраты на проект, уложиться в заданные сроки и повысить качество создаваемого программного продукта.

14.3 Создание артефактов в RUP

Основываясь на опыте многих успешных программных проектов, унифицированный процесс позволяет создавать сложные программные системы, опираясь на индустриальные методы разработки. Одним из них является процесс создания моделей при помощи унифицированного языка моделирования UML.

Унифицированный процесс можно представить как сумму различных видов деятельности компании-разработчика, необходимых для перевода требований заказчика в программную систему, которая выполняла бы именно то, что от нее ожидают пользователи, и выдавала требуемый результат. Поэтому процесс управляется вариантами использования (Use Case) системы, или прецедентами. Для реализации требований заказчика в установленные сроки унифицированный процесс разделяется на фазы, которые состоят из итераций. В ходе итераций создаются промежуточные артефакты, которые требуются для успешного завершения проекта и вариант программной системы, который реализует некоторый набор функций, увеличивающийся от итерации к итерации. Вся разработка ПО рассматривается в RUP как процесс создания артефактов, а любой результат работы проекта (исходные тексты, объектные модули, документы, модели) – это подклассы всех артефактов проекта. Каждый член проектной группы создает свои артефакты и несет за них ответственность. Программист создает программу, руководитель – проектный план, а аналитик – модели системы.

Одним из важнейших классов артефактов проекта являются модели, позволяющие разработчикам определять, визуализировать, конструировать и документировать артефакты программных систем. Каждая модель предназначена как для очерчивания проблем, так и для предложения решения. Полнота моделей заключается в том, что аналитик или разработчик может из конкретной модели почерпнуть всю необходимую ему информацию, не обращаясь к другим источникам. Большинство моделей

представляются UML диаграммами. Однако не следует забывать, что язык моделирования дает только нотацию – инструмент описания и моделирования системы, а унифицированный процесс определяет методику использования этого инструмента, как и других инструментов поддержки методологии от компании Rational. UML можно использовать и без конкретной методологии, поскольку он не зависит от процесса. Какой бы вариант процесса не был применен, можно использовать диаграммы для документирования принятых в ходе разработки решений и отображения создаваемых моделей. Нужно четко представлять, что даст внедрение данной технологии. Применение UML экономит ресурсы разработки, поскольку позволяет получить представление о системе быстрее, чем при создании макетов и прототипов, затратив на это несравнимо меньше ресурсов. Диаграммы позволяет легче общаться членам проекта между собой, и, что особенно ценно, вовлекают в процесс конечных пользователей системы. Моделирование позволяет уменьшить риски проекта. При этом всегда нужно помнить, что любая программная система создается для решения определенных задач пользователя, а не для апробирования новых технологий программистами и руководителем проекта. Поэтому, даже если программная система разработана при помощи суперсовременных методов и технологий, но пользователь не получит требуемый результат, программный продукт не будет востребован.

14.4 Создание программных средств в RUP

В основе RUP лежит ЖЦ создания ПС. Рассмотрим, как реализуются основные этапы ЖЦ в рамках RUP.

Определение требований. Унифицированный процесс – это процесс, управляемый прецедентами, которые отражают сценарии взаимодействия пользователей. Фактически, это взгляд пользователей на программную систему снаружи. Таким образом, одним из важнейших этапов разработки, согласно RUP, является этап определения требований, который заключается в сборе всех возможных пожеланий к работе системы. Позднее эти данные будут систематизированы и структурированы, но на данном этапе в ходе интервью с пользователями и изучения документов, аналитики должны собрать как можно больше требований к будущей системе. Работа усложняется тем, что пользователи часто сами не представляют, что они должны получить в конечном итоге. Для облегчения этого процесса аналитики используют диаграммы прецедентов (Use Case), а для детализации конкретного прецедента применяются диаграммы активности (Activity Diagram). Простота и понятность диаграммы прецедентов позволяет аналитикам легко общаться с заказчиками в процессе определения требований. Кроме того, диаграмма прецедентов может использоваться для создания сценариев тестирования, поскольку все взаимодействия пользователей и системы уже определены. Определение требований к системе напрямую связано с понимание контекста (части предметной

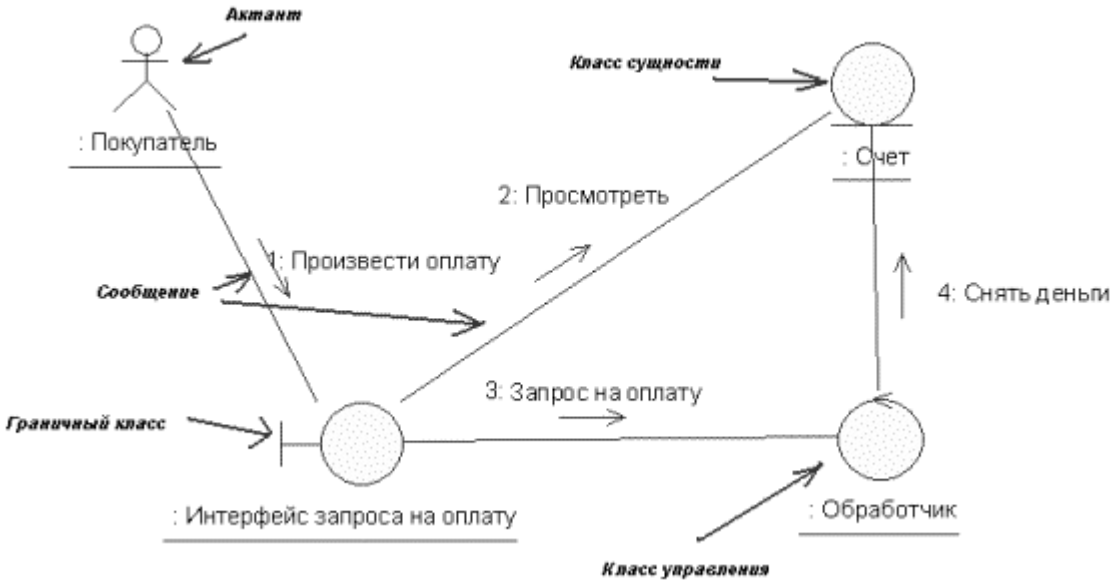
области), в котором будет работать система. Для этого создаются модель предметной области и бизнес-модель. Различия этих моделей в том, что первая описывает важные понятия, с которыми будет работать система и их связи между собой. А вторая описывает бизнес-процессы, которые должна поддерживать система. Поэтому кроме определения бизнес-объектов, вовлеченных в процесс, эта модель определяет работников, их обязанности и действия, которые они должны выполнять.

Для создания модели предметной области используется диаграмма классов (Class Diagram), однако для создания бизнес-модели ее уже недостаточно. В этом случае применяется диаграмма прецедентов с использованием дополнительных значков, отражающих сущность бизнес-процессов – это бизнес-актант, бизнес-прецедент, бизнес-сущность и бизнес-управление. Бизнес-модель во многом похожа на модель этапа анализа.

Анализ требований. После определения требований и контекста, в котором будет работать система, наступает этап анализа полученных данных. В процессе анализа создается аналитическая модель, которая подводит разработчиков к архитектуре будущей системы. Аналитическая модель – это взгляд на систему изнутри, в отличие от модели прецедентов, которая показывает, как система будет выглядеть снаружи. Эта модель позволяет понять, как система должна быть спроектирована, какие в ней должны быть классы и как они должны взаимодействовать между собой. Основное ее назначение – определить направление реализации функциональности, выявленной на этапе сбора требований и сделать набросок архитектуры системы.

Модель анализа является в большей степени концептуальной моделью и только приближает разработчиков к классам реализации. Для отображения модели анализа при помощи UML используется диаграмма классов с образцами поведения (стереотипами) «граничный класс», «сущность», «управление», а для детализации используется диаграмма сотрудничества (Collaboration Diagram) (рис 14.4).

Стереотип «граничный класс» отображает класс, который взаимодействует с внешними актантами, «сущность» отображает классы, которые являются хранилищами данных, а «управление» – классы, управляющие запросами к сущностям. Нумерация сообщений показывает их порядок, однако назначение диаграммы состоит в том, чтобы отразить связи классов друг с другом. Если акцентировать внимание на порядке взаимодействий, то удобнее это реализовать с помощью диаграммы последовательности (Sequence Diagram) (рис. 14.5). Решение о том, какую из двух диаграмм нужно создавать первой, зависит от предпочтений конкретного разработчика. Поскольку эти диаграммы являются отображением одного и того же процесса, то обе они позволяют отразить взаимодействие между объектами.



риунок 14.4 – Диаграмма сотрудничества

Проектирование. Следующим этапом в процессе создания системы будет проектирование, в ходе которого на основании моделей, созданных ранее, строится модель проектирования. Она отражает физическую реализацию системы и описывает создаваемый продукт на уровне классов и компонентов. В отличие от модели анализа, модель проектирования имеет явно выраженную зависимость от условий реализации, применяемых языков программирования и компонентов. Для наиболее точного понимания архитектуры системы эта модель должна быть максимально формализована и поддерживаться в актуальном состоянии на протяжении всего жизненного цикла разработки системы. Для построения модели проектирования используется целый набор UML диаграмм: классов (рис. 14.6), кооперации, взаимодействия, активности.

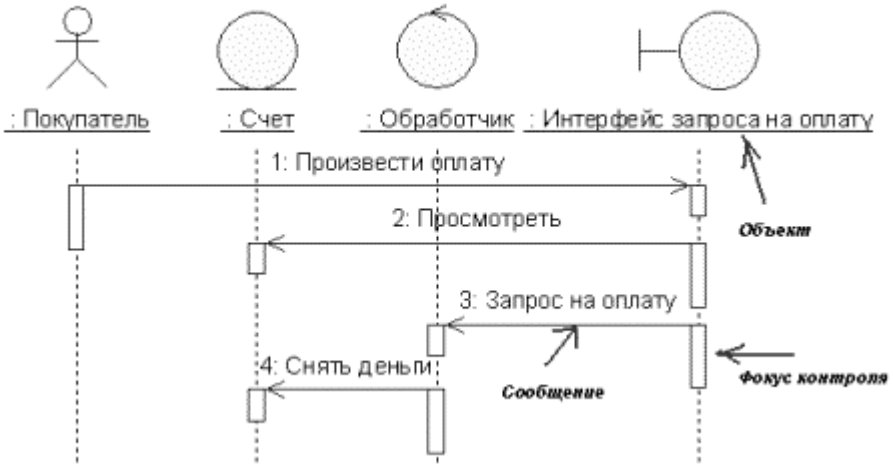


Рисунок 14.5 – Диаграмма последовательности действий

Дополнительно в этом рабочем процессе может создаваться модель развертывания, которая реализуется на основе диаграммы развертывания (Deployment Diagram).

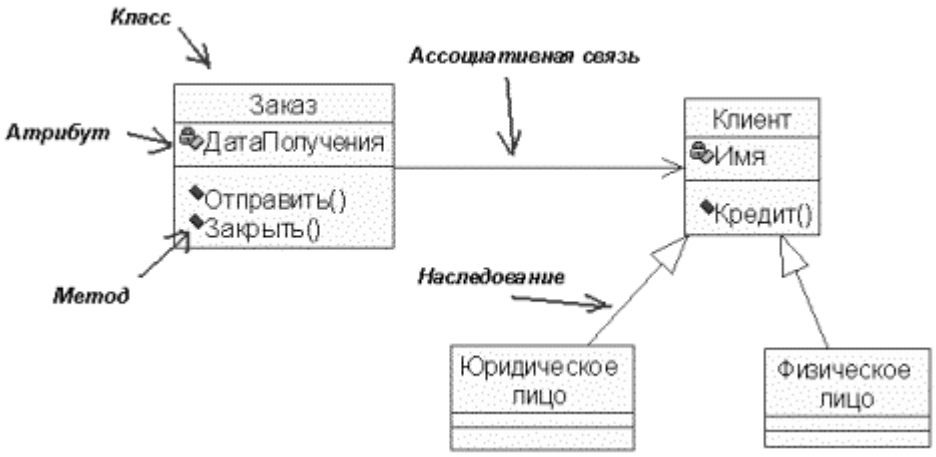


рисунок 14.6 – Диаграмма классов

Реализация. Основная задача процесса реализации – создание системы в виде набора компонентов. Это могут быть исходные тексты программ, сценарии, двоичные файлы, исполняемые модули и т.д. Модель реализации строится в виде диаграммы компонентов (Component Diagram) (рис. 14.7).

Тестирование. В процессе тестирования проверяются результаты реализации. Для данного процесса создается модель тестирования, которая состоит из тестовых примеров, процедур тестирования, тестовых компонентов, однако не имеет отображения в виде UML диаграмм.

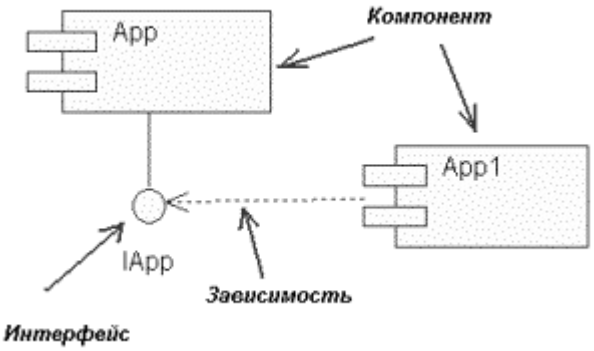


рисунок 14.7 – Диаграмма компонентов

Несмотря на огромное разнообразие создаваемых программных проектов, методы применения моделей, создаваемых при помощи UML, остаются неизменными. Диаграммы UML, создаваемые на различных этапах разработки, неотделимы от остальных артефактов программного проекта и часто являются связующим звеном между отдельными процессами RUP. Решение о применении конкретных диаграмм зависит от принятого в

каждом конкретном случае процесса разработки. Компания Rational не только предлагает его улучшать и дорабатывать, но и предоставляет специальные средства внесения изменений в базу данных RUP. Но в любом случае применение UML вместе с унифицированным процессом позволит получить предсказуемый результат, уложиться в отведенный бюджет, повысить отдачу от участников проекта и качество создаваемого программного продукта.