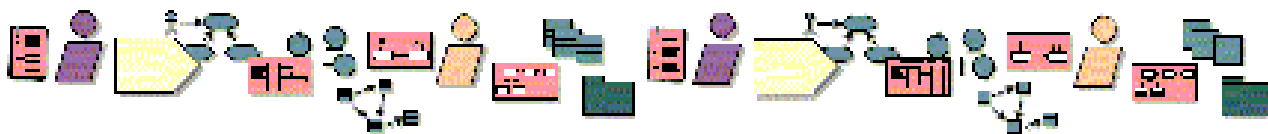


Введение в Rational Unified Process

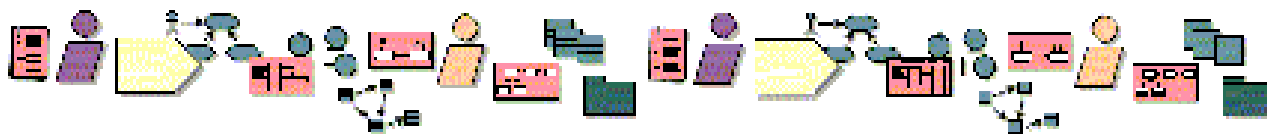




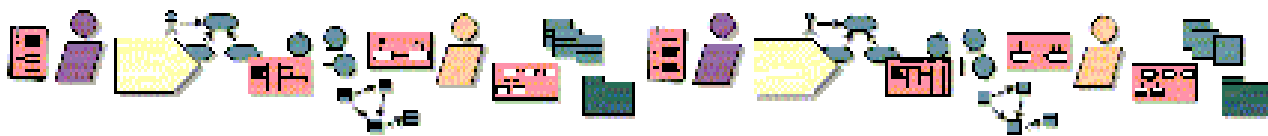


Содержание

Предисловие	5
Что такое Rational Unified Process?	7
Rational Unified Process как технология	7
Rational Unified Process как продукт	9
Архитектура процесса	11
Два измерения	11
Статическое содержание процесса	12
Структура жизненного цикла	16
Начальная стадия	17
Стадия уточнения	17
Стадия конструирования	18
Стадия перехода	18
Итерации	18
Особенности Rational Unified Process	21
Визуальное моделирование	21
Субъекты	22
Прецеденты	22
Объекты	23
Классы	23
Обобщения	23
Модели	23
Инструментальная поддержка	24
Управление прецедентами	24
Что такое прецедент?	25
Использование прецедентов при разработке	26



Итеративная разработка	28
Процесс «водопада»	28
Преимущества итераций	29
Управление требованиями	32
Кто такие совладельцы?	33
Анализ проблемы	33
Определение потребностей совладельцев	34
Определение системы	34
Управление контекстом проекта	34
Уточнение определения системы	35
Управление изменением требований	35
Инструментальная поддержка	36
Акцент на архитектуре	36
Что такое архитектура приложения?	36
Почему так нужен проект архитектуры?	37
Описание архитектуры	37
Круг интересов архитектуры	39
Методическая поддержка	40
Инструментальная поддержка	40
Разработка на базе компонентов	41
Настраиваемый процесс	42



Предисловие

Сегодня трудно найти книгу или статью, посвященную технологии создания больших программных систем, которая не цитировала бы убийственную характеристику нынешней ситуации: “По нашим оценкам только 26% проектов создания ИС заканчиваются успешно”. (Standish Group CHAOS Report 1998)

Большинство авторов приблизительно одинаково оценивают главную причину такого положения вещей. Это - возросшая сложность создаваемых систем. Проявлениями этой сложности являются не только увеличение размеров и функциональности. Не меньше, если не больше, проблем порождается частой сменой потребностей пользователей и ростом требований к качеству систем.

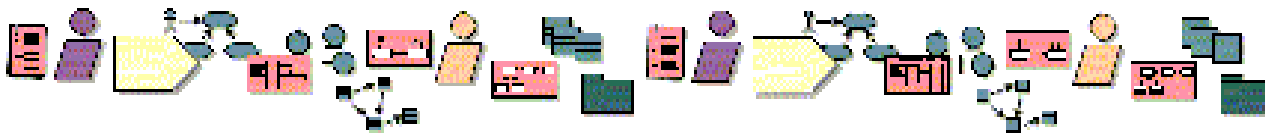
Традиционные способы разрешения проблемы сложности, такие как увеличение коллективов разработчиков, их специализация, распределение работ в чистом виде, ведут к еще большим трудностям согласования результатов и сборки готовых систем.

Серьезным шагом к победе над сложностью явилось появление объектно-ориентированного программирования (ООП). Но только шагом. Появление ООП не устранило проблем недостаточного взаимопонимания разработчиков и пользователей, неэффективного управления разработкой в условиях изменяющихся требований, неконтролируемости изменений в процессе выполнения работ, субъективности в оценке качества продуктов разработки и т.д.

И, тем не менее, 26% проектов заканчиваются успешно! Почему же все остальные снова и снова наступают на те же грабли? Почему они настойчиво продолжают учиться только на своих ошибках? Ведь решение лежит на поверхности: **нужно сформировать и документировать набор проверенных на практике принципов, методов и процессов качественной и производительной работы над проектами по созданию программного обеспечения.**

Эту миссию взвалила на свои плечи корпорация Rational Software и ее идеологи, столпы объектно-ориентированного подхода Grady Booch, Ivar Jacobson и James Rumbaugh. Rational Software выпустила в продажу Rational Unified Process, базу знаний, представляющую собой путеводитель для всех участников проекта по разработке большой программной системы. Далее, Rational Software выпустила комплект Rational Suite, ядром которого является Rational Unified Process, а остальные компоненты обеспечивают его инструментальную поддержку. И, наконец, Rational Software определила перечень продуктов других фирм, с которыми Rational Unified Process также тесно интегрирован.

Таким образом, вполне правомерно говорить о Rational Software технологии разработки программного обеспечения. Целью цикла публикаций, начинаемого этой статьей, является



представление специалистам Rational Software технологии.

Я хочу подчеркнуть, что этот цикл не носит рекламного характера. В статьях цикла будут описываться принципы и приемы действий, способы использования рекомендуемых Rational Software инструментальных средств, приводиться примеры. Вы сможете попробовать применить полученные знания в своей работе. Если у Вас есть инструменты Rational Software – хорошо. Если нет – рисуйте диаграммы и планы на промокашках. Вы быстро убедитесь, что это хорошо, но мало. И тогда Вы сами, без всякой рекламы, обратитесь на наш сайт <http://www.interface.ru>, где представлены все необходимые сведения для приобретения нужных Вам продуктов.

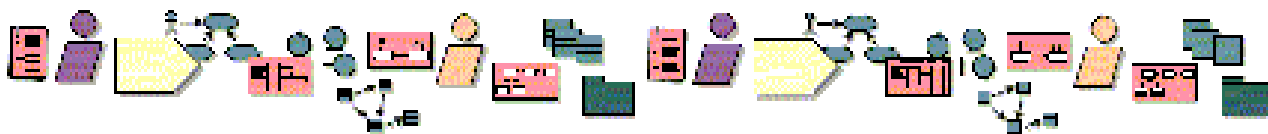
Статьи цикла предполагается публиковать приблизительно раз в две недели. Формат публикации (.pdf) позволяет устанавливать гиперссылки внутри и между статьями. Поэтому, при появлении новых статей, возможно появление новых версий уже опубликованных. Это может быть связано как с появлением ссылок, так и с необходимостью уточнений или устранения ошибок. Я очень рассчитываю на обратную связь с читателями.

По окончании цикла Вы сможете напечатать на своем принтере маленькую книжку под условным названием «Введение в Rational Unified Process». А может быть, к этому времени появится локализованная русская версия Rational Unified Process. При дальнейшем чтении Вы убедитесь, что это было бы очень полезно!

С уважением

Леонид Новиков
(novikov@interface.ru)

А теперь приступим...

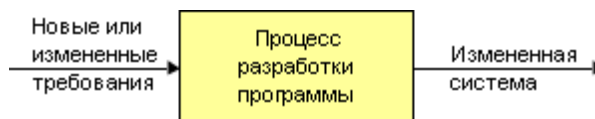


Что такое Rational Unified Process?

Rational Unified Process как технология

Rational Unified Process - это процесс разработки программного обеспечения.

Процесс – частично упорядоченный набор шагов, которые нужно проделать для достижения цели; при разработке программного обеспечения цель состоит в формировании или расширении существующего программного изделия.

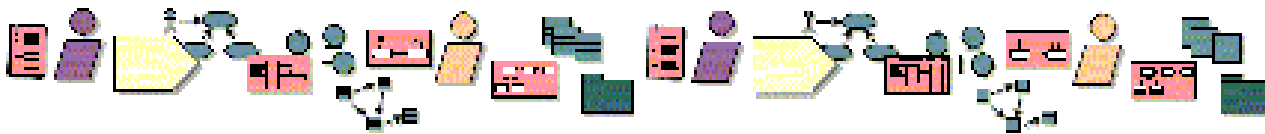


Процесс разработки программного обеспечения - процесс разработки системы из требований, новых (начальный цикл развития) или измененных (цикл эволюции).

Когда программная система разрабатывается на пустом месте, ее разработка – это процесс создания системы из требований. Но как только система приняла какую-то форму (или в наших терминах, когда система прошла начальный цикл развития), то любая доработка – это процесс приспособления системы к новым или изменившимся требованиям. Разработка и все последующие доработки составляют **жизненный цикл системы**.

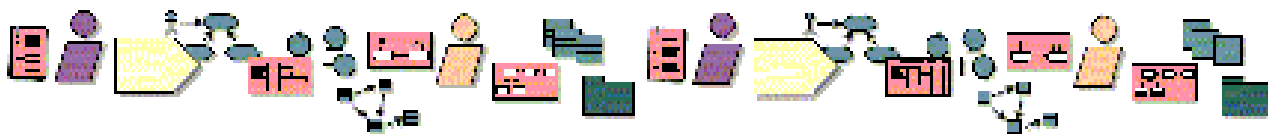
Rational Unified Process обеспечивает строгий подход к назначению задач и ответственности в пределах группы разработки. Его **цель** состоит в том, чтобы гарантировать высокое качество программного продукта, отвечающего потребностям конечных пользователей, в пределах предсказуемого временного графика и бюджета.

- Rational Unified Process – это **итеративный** процесс. Создавать современные сложные программные системы последовательно, т.е. сначала определять все проблемы, затем принимать все проектные решения, формировать программное обеспечение и, наконец, проверять изделие, невозможно. **Итерационный подход** позволяет улучшать понимание проблемы через последовательные усовершенствования и с приращением конкретизировать эффективные решения. Этот подход обеспечивает большую гибкость при учете новых требований или тактических изменений в деловых целях, и позволяет проекту заранее **идентифицировать и разрешать риски**.
- Rational Unified Process – это **управляемый** процесс. Итерационный подход предполагает **управление требованиями и управление изменениями**, чтобы по всем пунктам вовремя



гарантировать общее понимание ожидаемых функциональных возможностей, ожидаемый уровень качества и гарантировать наилучшее управление связанными затратами и графиками выполнения.

- Rational Unified Process заключается в создании и обслуживании **моделей**. Rational Unified Process фокусирует внимание не на создании большого количества бумажных документов, а на развитии и эксплуатации моделей - семантически богатых представлений программной системы при ее разработке.
- Rational Unified Process сосредотачивает внимание на первоначальной разработке и компоновке устойчивой **архитектуры** программы, которая облегчает параллельную разработку, минимизирует переделки, увеличивает возможность многократного использования и надежность эксплуатации. Эта архитектура используется для планирования использования и управления развитием программных **компонентов**.
- Действия при выполнении Rational Unified Process **управляются прецедентами**. Понятия прецедентов и сценария управляют технологическим маршрутом от делового моделирования и требований до испытаний, и обеспечивают связанные и доступные для анализа маршруты разработки и поставки системы.
- Rational Unified Process поддерживает **объектно-ориентированную технологию**. Некоторые из моделей являются объектно-ориентированными моделями, которые базируются на понятиях объектов, классов и зависимостей между ними. Эти модели, подобно многим другим техническим искусственным объектам (артефактам), используют Унифицированный язык моделирования (UML) как общую систему обозначений.
- Rational Unified Process поддерживает **компонентно-ориентированное программирование**. Компоненты - это нетривиальные модули или подсистемы, которые выполняют конкретную функцию и могут быть смонтированы в строго очерченной архитектуре, специальной или некоторой общедоступной инфраструктуре компонентов, типа Internet, CORBA, COM/DCOM, для которых появляется индустрия многократно используемых компонентов.
- Rational Unified Process – это процесс **с перестраиваемой конфигурацией**. *Никакой одиночный процесс не подходит для всех случаев разработки программного обеспечения.* Rational Unified Process удовлетворяет и маленькие группы разработчиков и большие организации. Rational Unified Process основан на простой и корректной архитектуре, которая обеспечивает общность для семейства процессов, и все же может быть изменена ради приспособления к конкретным ситуациям. Он содержит рекомендации по конфигурированию процесса для удовлетворения потребностей данной организации.
- Rational Unified Process поощряет объективно осуществляемое **управление качеством**. Оценка качества всех действий и их участников, формируемая в процессе, использует объективные измерения и критерии.



- Rational Unified Process поддерживается **инструментальными средствами**, которые автоматизируют большинство действий процесса. Инструментальные средства используются для создания и обслуживания различных артефактов процесса разработки программного обеспечения: визуального моделирования, программирования, испытаний и так далее. Они неопределимы в поддержке всей бухгалтерии, связанной с управлением изменениями и управлением конфигурацией, которыми сопровождается каждая итерация.

Впоследствии мы более подробно охарактеризуем каждое из этих качеств Rational Unified Process.

Rational Unified Process как продукт

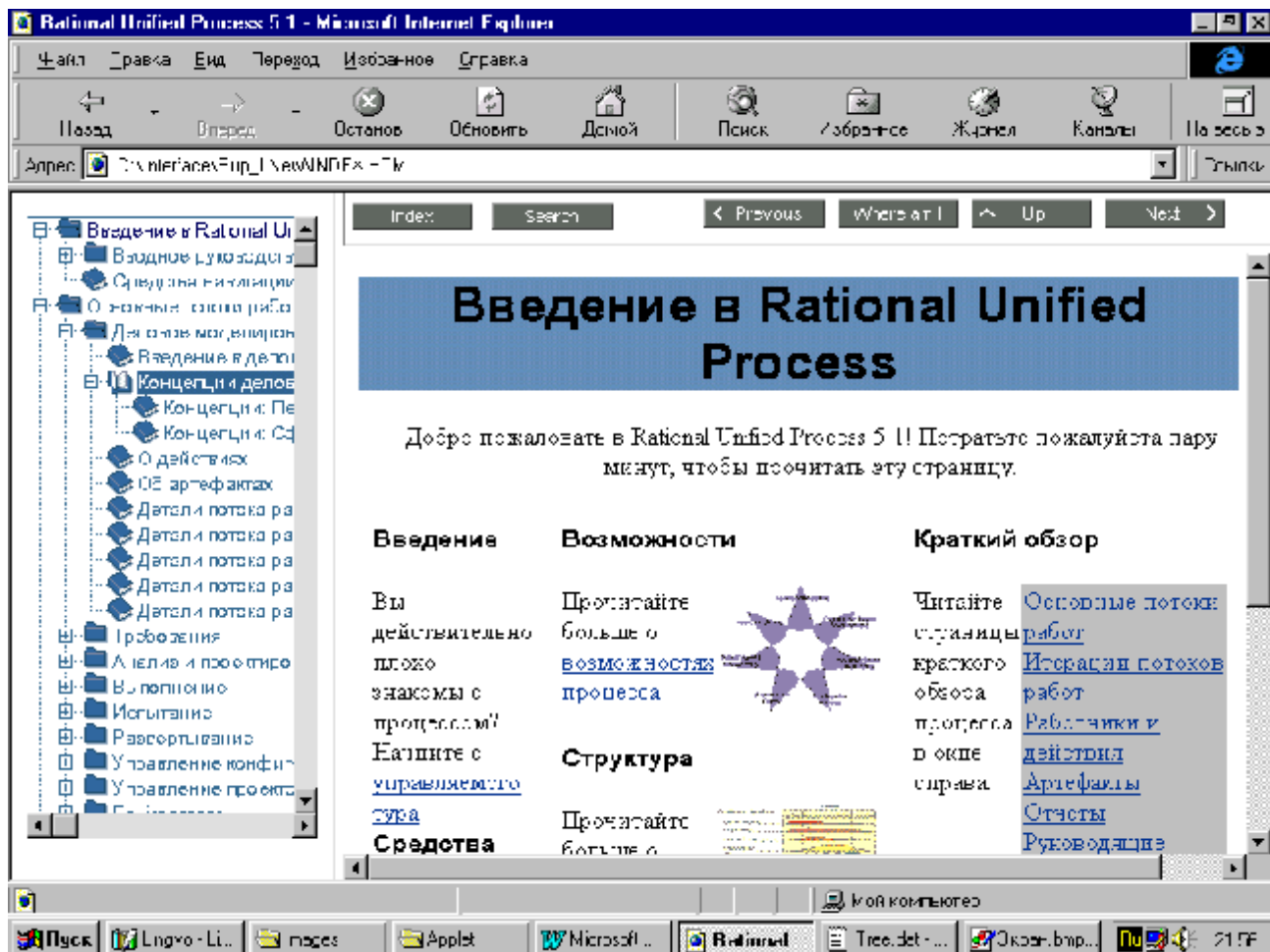
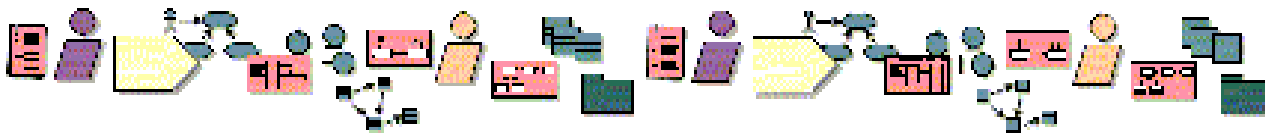
Продукт Rational Unified Process входит в состав Rational Suite всех комплектаций. В поставку входит:

- интерактивная версия базы знаний в формате HTML, к которой приложен
- комплект документов, описывающих процесс.

База знаний содержит описания архитектуры процесса и его составляющих (стадий разработки и потоков работ), технологии работы, правил создания различных артефактов: моделей, отчетов, планов и т.д. и порядка использования средств инструментальной поддержки.

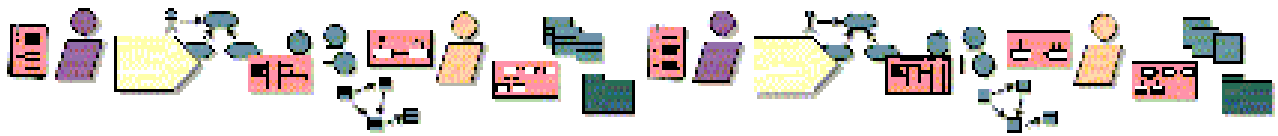
Интерактивная версия содержит настраиваемые шаблоны, которые позволяют использовать информацию, порожденную в процессе работы над проектом, для создания отчетных документов с использованием Microsoft Word, Microsoft Project и Microsoft FrontPage.

Для работы с интерактивной версией Rational Unified Process на Вашем компьютере должен быть установлен Netscape Navigator или Microsoft Internet Explorer. Ниже показан один из экранов Rational Unified Process, на котором Вы можете узнать типичные для подобного рода продуктов средства управления.



Так выглядит одно из окон Rational Unified Process на дисплее компьютера.

В одной из следующих статей мы подробно остановимся на интерактивных возможностях Rational Unified Process и попытаемся смоделировать его работу.

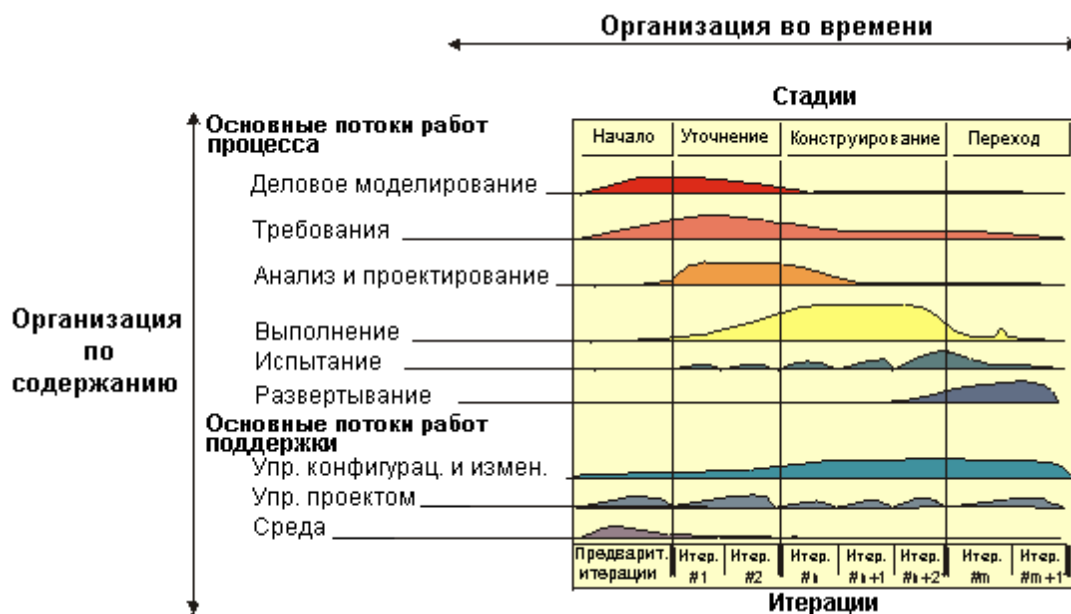


Архитектура процесса

Два измерения

Rational Unified Process представляет процесс разработки программной системы в двух измерениях:

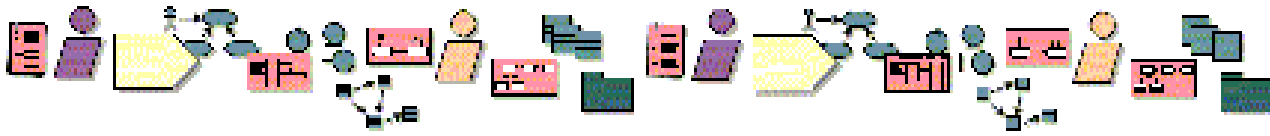
- По содержанию действий участников групп разработки (по основным потокам работ).
- Во времени (по стадиям жизненного цикла разрабатываемой системы).



Процесс организован по содержанию (основные потоки работ) и во времени (стадии).

Примечание: Установите курсор на надписи «Организация во времени». Если Вы нажмете кнопку мыши, Вы перейдете в начало раздела «Структура жизненного цикла». Такой способ ссылок мы будем использовать очень часто. Поэтому, увидев на иллюстрации объект, о котором Вы хотите получить дополнительную информацию прямо сейчас, проверьте, не имеет ли он ссылки.

Первое измерение представляет **статический** аспект процесса: оно описано в терминах основных потоков работ (исполнители, действия, их последовательность и так далее).



Второе измерение представляет **динамический** аспект процесса, поскольку оно выражено в терминах циклов, стадий, итераций и этапов.

Описание процесса выполняется с двух различных точек зрения:

- **Техническая** точка зрения фокусируется на артефактах и представлениях, связанных с разрабатываемым изделием.
- **Организационная** точка зрения фокусируется на времени, бюджете, людях и других экономических соображениях.

Примечание: Обратите внимание, что в этом изложении мы считаем деловое моделирование частью процесса разработки программного обеспечения. Это так, если Вы выполняете ограниченное деловое моделирование специально, чтобы выяснить требования к прикладной программе. Но деловое моделирование может быть выполнено и совершенно отдельно, с первичной целью заново спроектировать организацию, а не только сформировать новые прикладные программы. В этом последнем случае деловое моделирование должно рассматриваться как его собственный процесс.

Статическое содержание процесса

Статическое содержание процесса организовано в основных потоках работ. Rational Unified Process включает девять основных потоков работ; шесть потоков работ процесса разработки:

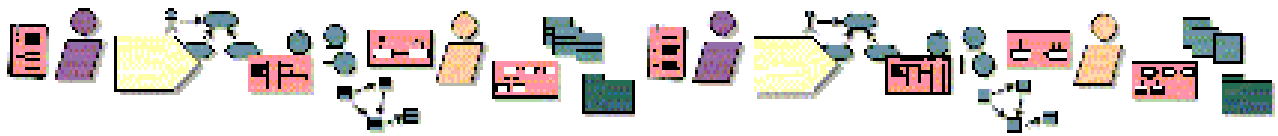
- Деловое моделирование
- Требования
- Анализ и проектирование
- Выполнение
- Испытание
- Развертывание

и три потока работ поддержки:

- Управление конфигурацией и изменением
- Управление проектом
- Среда

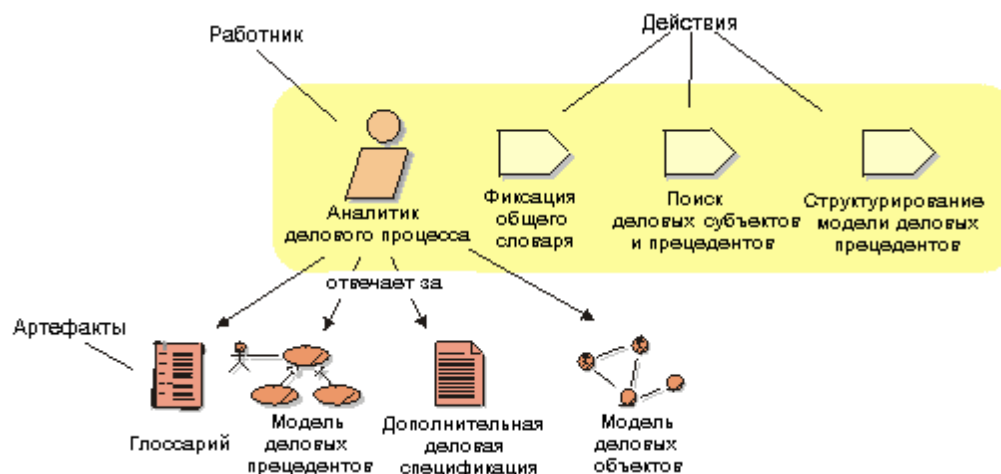
Основные потоки работ описаны в терминах работников, действий и артефактов.

- **Работник** определяет поведение и ответственности индивидуума или нескольких индивидуумов, работающих вместе как группа. Это - важное отличие, потому что обычно о работнике думают как о конкретном человеке или группе. В Rational Unified Process, работник - больше роль, которая определяет, как индивидуумы должны выполнять работу.



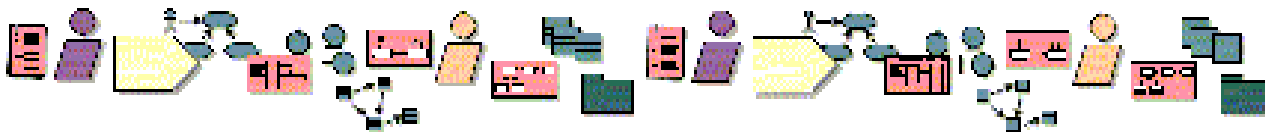
- **Действие** – это самая маленькая часть работы, которая относится к делу; его можно интерпретировать как «техническую операцию» работника. Далее, невозможно выполнить только часть действия, хотя в пределах действия может существовать некоторая необязательная операция. Такое разделение работы облегчает возможность контролировать разработку. Гораздо лучше (проще) знать, что в проекте реализованы три из пяти действий, чем то, что выполнено 60 % проекта.
- **Артефакты** – искусственные объекты (конструкции моделирования и документы), которые действия выделяют, поддерживают или используют как исходную информацию.

С каждым работником ассоциируется набор «связанных» действий; связанность означает, что действия будут лучше выполнены этим индивидуумом. Ответственности каждого работника обычно определяются относительно некоторых артефактов, например документов. Примеры работников: аналитик делового процесса, проектировщик, архитектор или инженер-технолог.



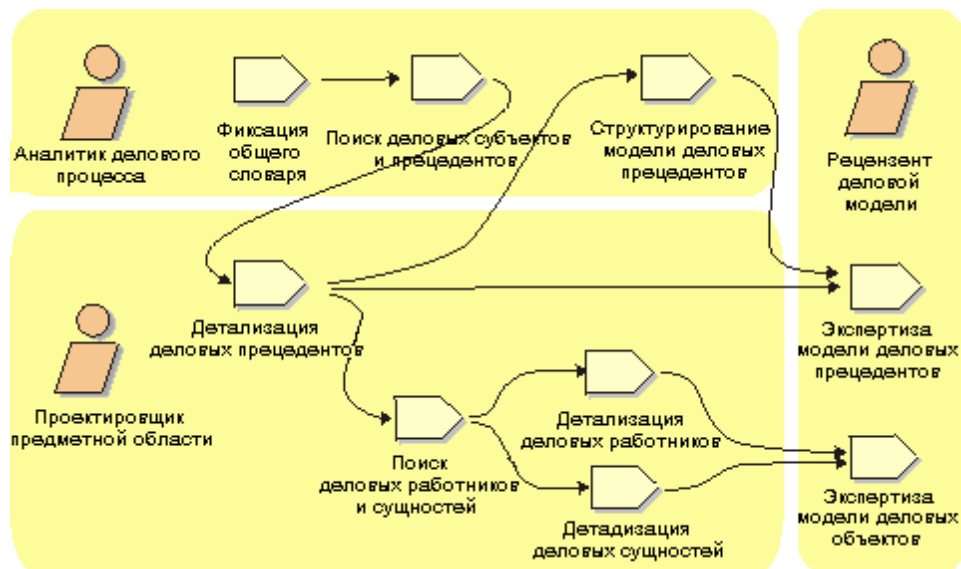
Каждый работник имеет свой собственный набор действий и артефактов

Примечание: В интерактивной версии Rational Unified Process щелчок на любой пиктограмме инициирует переход к описанию соответствующего объекта. Например, щелчок на пиктограмме артефакта Глоссарий (см. выше) откроет страницу с описанием структуры глоссария, требованиями к его содержанию, рекомендациями по использованию инструментальной поддержки и т.д.

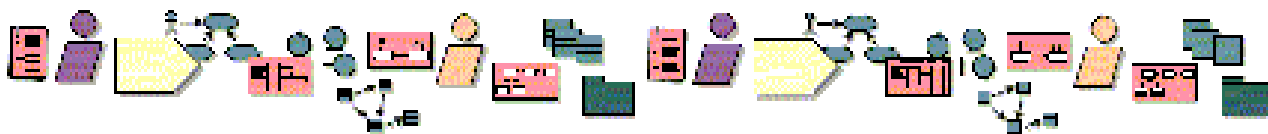


Через связанный набор действий работник неявно определяет также и навыки, необходимые для исполнения работ.

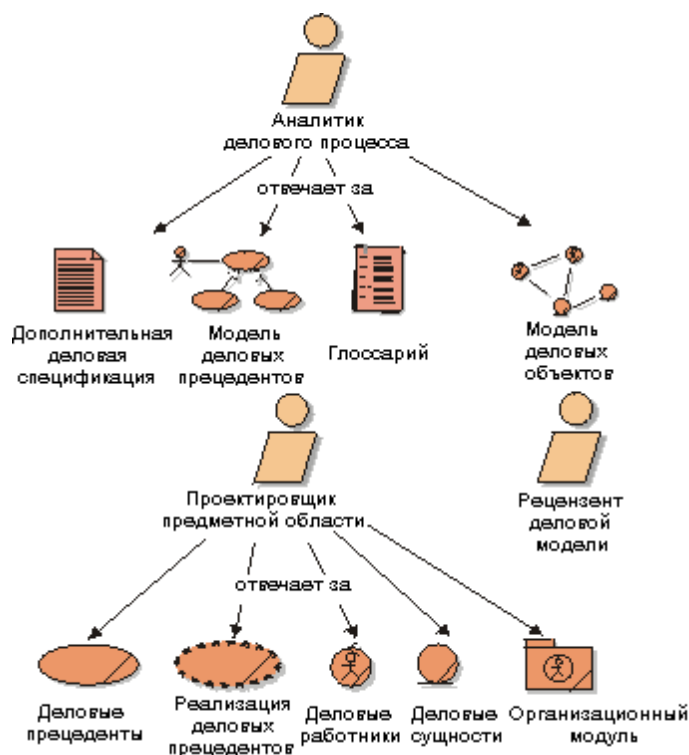
Для каждого основного потока работ представляется диаграмма **краткого обзора действий**. Эта диаграмма показывает все действия и всех работников, включенных в поток работ.



Типовая диаграмма краткого обзора действий (поток работ делового моделирования)



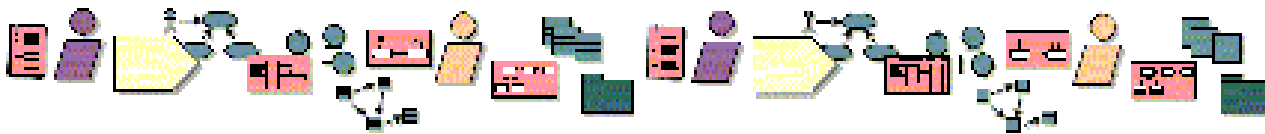
Для каждого основного потока работ представлена диаграмма **краткого обзора артефактов**. Эта диаграмма показывает все артефакты и всех работников, включенных в поток работ.



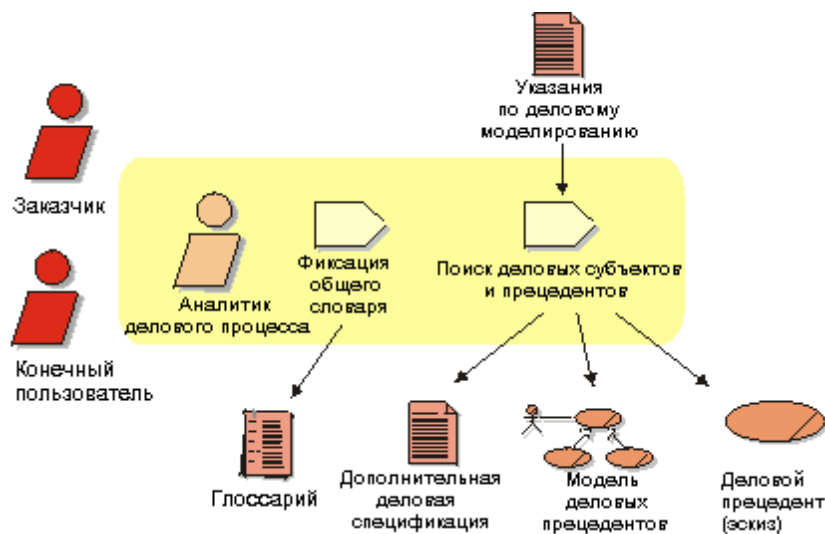
Типовая диаграмма краткого обзора артефактов (поток работ делового моделирования)

Для большинства из основных потоков работ Вы найдете также диаграммы **деталей потока работ**, которые показывают группировки действий, которые часто выполняются вместе. Эти диаграммы показывают соответствующих работников, артефакты ввода и вывода и выполняемые действия. Диаграммы деталей потоков работ представляются по следующим причинам:

- Действия потока работ не выполняются ни в строго определенной очередности, ни внезапно. Обычно Вы работаете параллельно больше, чем над одним действием и при этом рассматриваете больше, чем один артефакт. Диаграмма деталей потока работ показывает, как часто Вы выполняете поток работ или организуете совещания группы при выполнении потока работ. Обычно для основного потока работ представлено несколько диаграмм деталей потока работ.
- Бывает слишком сложно показать артефакты ввода и вывода для всех действий основного потока работ на одной диаграмме. Диаграмма деталей потока работ позволяет нам показать одновременно действия и артефакты для некоторой части потока работ.
- Основные потоки работ не полностью независимы друг от друга. Например, интеграция происходит в обоих потоках работ: и выполнения, и испытания. В действительности Вы



никогда не делаете одно без другого. Диаграмма деталей потока работ может показывать группу действий и артефактов в потоке работ вместе со связанными действиями из другого потока работ.



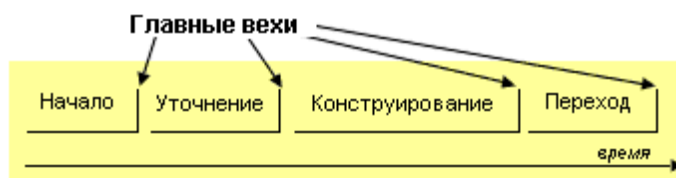
Типовая диаграмма деталей потока работ (поток работ делового моделирования)

Структура жизненного цикла

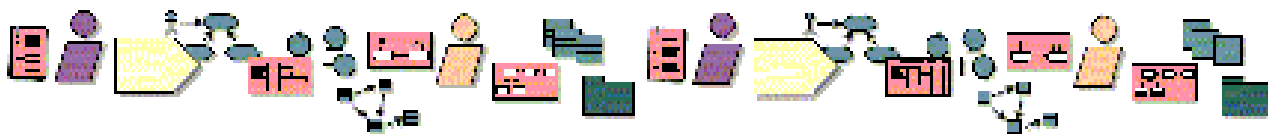
Когда мы рассматриваем динамическую организацию процесса во времени, жизненный цикл программы разбивается на циклы, каждый из которых работает над новым поколением изделия. Rational Unified Process делит один цикл развития на четыре последовательных стадии:

- Начало
- Уточнение
- Конструирование
- Переход

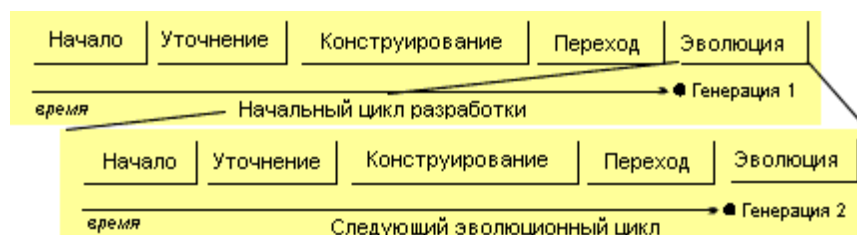
Каждая стадия заканчивается четкой определенной вехой – временной точкой, в которой должны быть приняты некоторые критические решения, а поэтому ключевые цели должны быть достигнуты.



Стадии и главные вехи процесса.



Первый цикл выполнения этих четырех стадий для данного изделия называют начальным циклом разработки. Если жизненный цикл изделия на этом не завершается, существующее изделие разовьется в своем следующем поколении, повторив ту же последовательность стадий: начала, уточнения, конструирования и перехода. Этот период называется «эволюцией». Циклы развития, которые следуют за начальным циклом развития, называются «эволюционными циклами».



Два последовательных цикла разработки

Начальная стадия

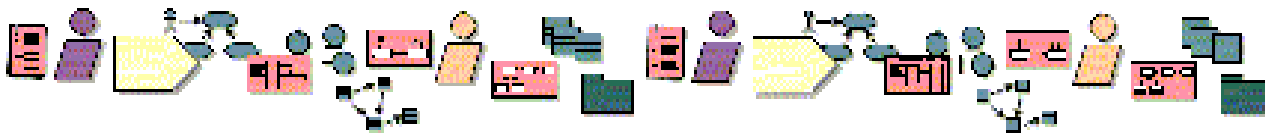
На начальной стадии Вы устанавливаете деловые применения системы и определяете рамки проекта. Чтобы сделать это, Вы должны идентифицировать все внешние объекты, с которыми взаимодействует система (субъекты), и определить характер этого взаимодействия на высоком уровне. Эта работа включает идентификацию всех прецедентов и описание нескольких наиболее существенных. Деловое применение включает критерии успеха, оценку риска, оценку необходимых ресурсов и укрупненный план с указанием дат завершения главных этапов.

В конце начальной стадии Вы исследуете требования жизненного цикла проекта и решаете, следует ли продолжать разработку.

Стадия уточнения

Цели стадии уточнения состоят в том, чтобы проанализировать прикладную область, создать нормальную архитектурную основу, разработать план проекта и устранить самые высокие элементы риска проекта. Архитектурные решения должны приниматься с пониманием целостной системы. Это подразумевает, что Вы описываете большинство прецедентов и принимаете во внимание некоторые из связей: дополнительные требования. Чтобы проверить архитектуру, Вы выполняете систему, которая демонстрирует архитектурные решения и выполняет существенный прецедент.

В конце стадии уточнения Вы детально исследуете цели и контекст системы, архитектурные решения и способы разрешения главных рисков.



Стадия конструирования

В ходе стадии конструирования Вы итерационно и с приращением разрабатываете законченное изделие, которое должно быть готово к передаче пользователям. Это подразумевает описание остающихся прецедентов, изложение деталей конструкции, завершение выполнения и проверку программного обеспечения.

В конце стадии конструирования Вы решаете, все ли программное обеспечение, рабочие места и пользователи готовы и работоспособны.

Стадия перехода

В процессе стадии перехода Вы передаете программное обеспечение пользователям. Как только изделие попадает в руки конечных пользователей, часто возникают новые проблемы, которые требуют дополнительной разработки по корректировке системы, исправлению необнаруженных ранее проблем или завершению реализации некоторых возможностей, которые, возможно, были отложены. Эта стадия обычно начинается с выпуска «бета-версии» системы.

В конце переходной стадии Вы решаете, достигнуты ли цели жизненного цикла, и возможно, запускаете другой цикл разработки. Это также та точка, в которой Вы можете проанализировать некоторые из уроков, полученных в процессе разработки проекта.

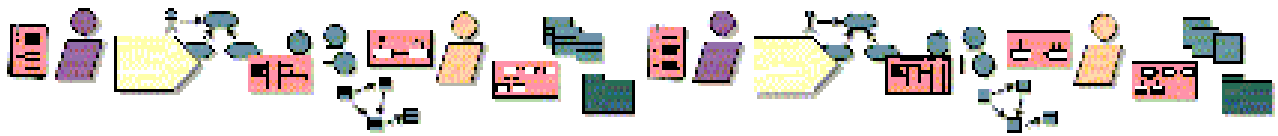
Итерации

Каждая стадия Rational Unified Process может быть в свою очередь разбита на итерации. Итерация – это законченный цикл разработки, приводящий к выпуску выполнимого изделия (внутренней или внешней версии) или подмножества конечного продукта, которое возрастает с приращением от итерации к итерации, чтобы стать законченной системой.

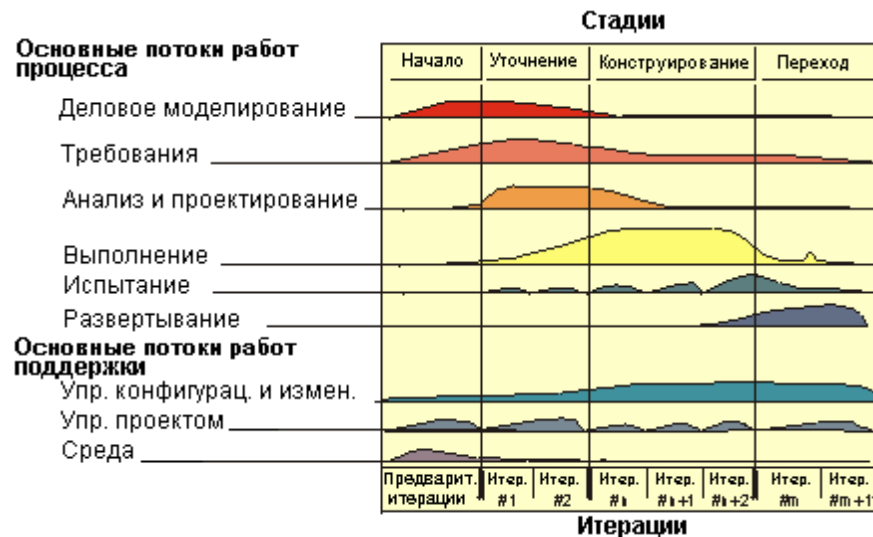


Каждая итерация в пределах стадии приводит к выпуску выполнимой системы.

Каждая итерация содержит все аспекты разработки программного обеспечения и повторяет все основные потоки работ. Но акценты на основных потоках работ различны, в зависимости от стадии разработки.

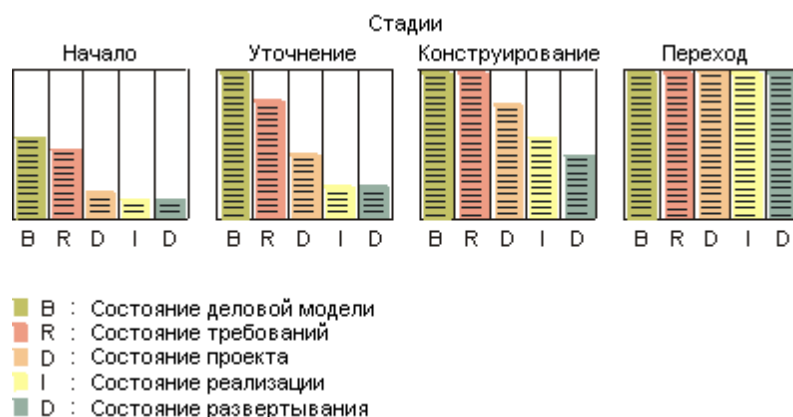


Это положение иллюстрируется следующей диаграммой, где показана трудоемкость каждого из основных потоков работ по мере Вашего продвижения от итерации к итерации через все четыре стадии.



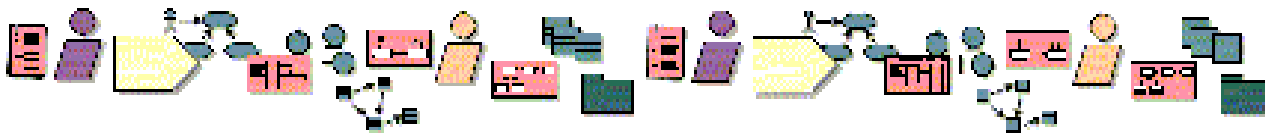
Стадии состоят из итераций, содержащих все основные потоки работ. Каждый из основных потоков работ ответственен за набор артефактов. Высота кривых отражает характер интенсивности потока работ на стадиях и итерациях.

Главное следствие такого итерационного подхода – артефакты, описанные ранее, обогащаются и через какое-то время становятся полностью зрелыми, как это показано на следующей диаграмме.



Эволюция состояния информации по стадиям разработки.





служащий в системе управления персоналом.

Модель, правильно разработанная с использованием объектной технологии:

- проста для понимания. Она ясно отражает действительность.
- проста для изменения. Изменение в конкретном явлении касается только объекта, представляющего это явление.

В этом разделе Вы познакомитесь с понятиями субъектов и прецедентов. Мы обсудим также другие объектно-ориентированные понятия и общие идеи моделирования. Более детальные определения Вы найдете, если у Вас хватит терпения, в главах, посвященных основным потокам работ.

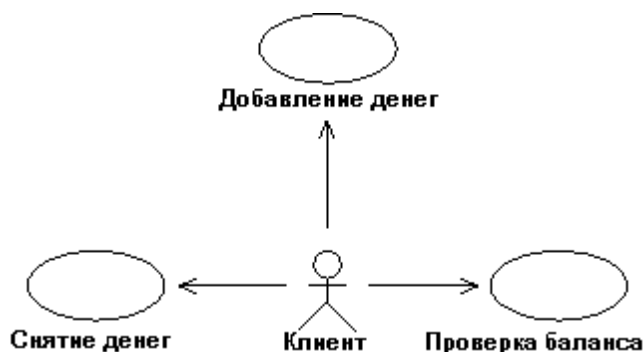
Субъекты

Для полного понимания назначения системы Вы должны знать, **для кого** предназначена эта система и **кто** будет ее использовать. В Rational Unified Process различные типы пользователей представлены **субъектами**. Субъектом представляется также и любая другая система, которая взаимодействует с нашей системой; таким образом, **субъекты определяют границы системы**.

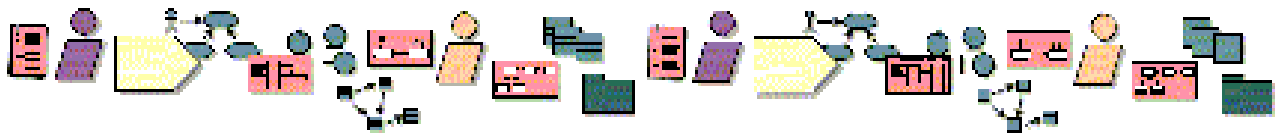
Прецеденты

Функциональные возможности системы определяются **прецедентами**, каждый из которых представляет определенный способ использования системы. Описание прецедента определяет то, что произойдет в системе, когда будет выполнен прецедент. Таким образом, каждый прецедент соответствует последовательности действий, выполняемых системой, которая выдает наблюдаемый результат, имеющий ценность для некоторого субъекта.

Действие – это атомарный набор операций, который выполняется полностью или частично.



Подойдя к банкомату, клиент может, например, снять деньги со счета, добавить деньги на счет или проверить состояние счета. Эти действия соответствуют потокам событий в системе, которые могут быть представлены прецедентами.



Объекты

Объект – это абстракция чего-либо в прикладной области или в выполняемой системе. Вы помните, что объектами могут быть, например, счет в банковской системе или служащий в системе управления персоналом.

Объект инкапсулирует данные и поведение. Данные объекта представляются **атрибутами**, а его поведение – **операциями**.

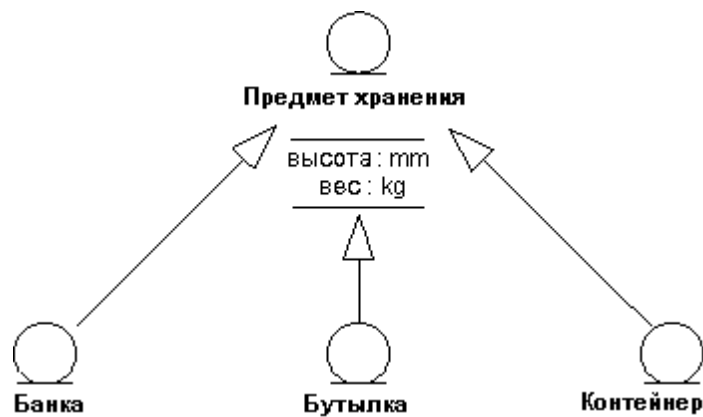
Объект определяется в **классе**.

Классы

Класс определяет шаблон структуры и поведение всех его объектов. Объекты, созданные в классе, называются также **экземплярами** класса.

Обобщения

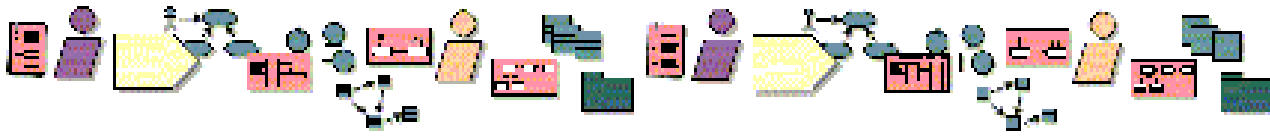
Обобщения показывают, как один класс наследуется от другого. Наследованный класс называется **потомком**. Класс, от которого происходит наследование, называется **предком**. Наследование означает, что определение предка – включая любые свойства типа атрибутов, связей или операций его объектов – является правильным и для объектов его потомка. Обобщение выводится от класса-потомка к его классу-предку.



Банки, бутылки и контейнеры имеют общие свойства высоты и веса. Для работника склада каждое из понятий является уточнением обобщённого понятия «предмет хранения».

Модели

Разработка моделей широко принята во всех технических дисциплинах в значительной степени потому, что она позволяет представить себе предмет моделирования прежде, чем он будет создан фактически. Моделирование позволяет в самом начале увидеть проблемы, установить и устранить которые позже было бы чрезвычайно дорого или вообще невозможно.



В Rational Unified Process используется несколько моделей системы. Это связано с тем, что в процессе жизненного цикла системы модели используются разными специалистами, каждый из которых имеет свои интересы. Когда Вы обсуждаете систему и требования к ней, например, с конечным пользователем или заказчиком, модель должна описывать **то, что** система должна делать, и абстрагироваться от подробностей выполнения. Позже, при проектировании, модель должна фокусироваться на потребностях проектировщиков, которые должны иметь возможность обсуждать проблемы декомпозиции, абстракции и иерархии на уровне выше уровня языка программирования.

Выполнение и тестирование также требуют различных характеристик моделей, приемлемых для разработчиков и тестировщиков. Таким образом, двигаясь через жизненный цикл приложения, Вы должны будете разработать несколько моделей, которые описывают различные аспекты разрабатываемой системы.

Инструментальная поддержка

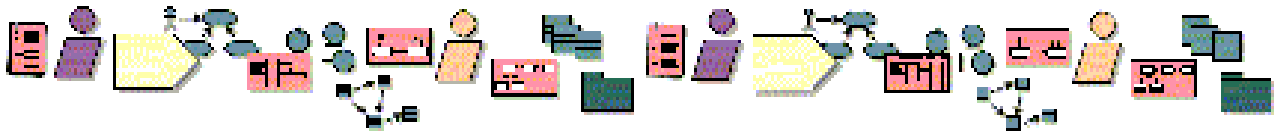
Раньше уже говорилось о том, что модели в Rational Unified Process используют UML как общую систему обозначений. Но при разработке больших систем этого мало. Нужно иметь инструмент, который поддерживал бы не только рисование составляющих модели диаграмм, но и контролировал бы их непротиворечивость и предоставлял возможность прямой и обратной разработки. Это означает, что Вы должны иметь возможность выполнять прямое проектирование и перепроектирование кода без того, чтобы отменять изменения, которые были сделаны в моделях или в коде, начиная с предыдущей генерации.

Rational Unified Process в рамках Rational Suite интегрирован с Rational Rose, который представляет собой именно такой инструмент моделирования.

Управление прецедентами

Положа руку на сердце, скажу - меня поразила бессмысленность использования прецедентов, когда я услышал о них впервые. Тем более, что прецеденты не являются частью «традиционного» объектного программирования. Как же я был не прав!

В традиционной объектно-ориентированной модели системы часто бывает трудно указать, как система будет делать то, что она должна делать. Эта трудность возникает из-за отсутствия «красной нити», объединяющей систему при выполнении отдельных задач. В Rational Unified Process такой «красной нитью» являются **прецеденты**, которые определяют поведение системы. Другие объектно-ориентированные методы также используют прецеденты, применяя для них разные названия.



Что такое прецедент?

Понятие прецедента было введено на [стр. 22](#) при обсуждении основных концепций моделирования. Приведем определения:

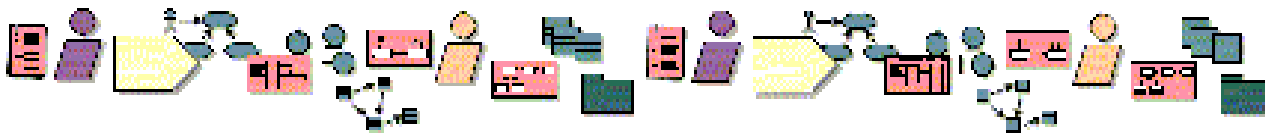
Экземпляр прецедента – это последовательность действий, выполняемых системой; эта последовательность имеет наблюдаемый результат, представляющий ценность для конкретного субъекта.

Прецедент – это набор экземпляров прецедента.

В этом определении есть несколько ключевых слов:

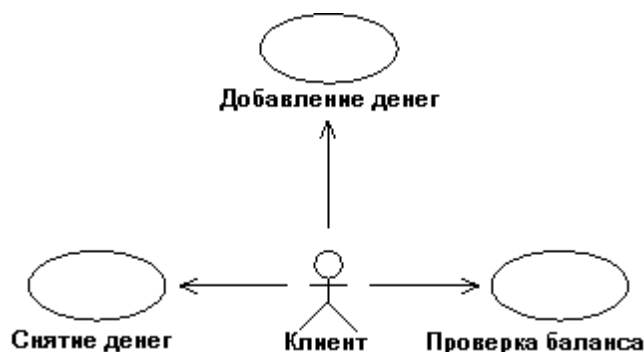
- **Экземпляр прецедента.** Последовательность, упомянутая в определении, действительно является конкретным потоком событий в системе, или экземпляром. Возможно множество потоков событий, и многие из них могут быть очень похожи. Чтобы сделать модель прецедентов понятной, можно сгруппировать подобные потоки событий в один прецедент. Идентификация и описание прецедента в действительности означает идентификацию и описание группы связанных потоков событий.
- **Система выполняет.** Это означает, что система обеспечивает прецедент. Субъект связывается с экземпляром прецедента системы.
- **Наблюдаемый результат, представляющий ценность.** Вы можете оценивать успешность выполнения прецедента. Прецедент следует производить, если в результате субъект сможет решить задачу, которая имеет реальный результат. Это очень важно для определения правильного уровня и глубины детализации прецедента. Правильным является стремление производить не слишком маленькие прецеденты. В некоторых обстоятельствах Вы можете использовать прецедент как элемент планирования.
- **Действия.** Действие – это вычислительная или алгоритмическая процедура; оно выполняется, когда субъект дает системе соответствующий сигнал или когда происходит временное событие. Действие может подразумевать передачу сигнала вызывавшему субъекту или другим субъектам. Действие атомарно; это означает, что оно или выполняется полностью, или не выполняется вовсе.
- **Конкретный субъект.** Субъект является ключевой фигурой при поиске правильного прецедента, особенно потому, что субъект помогает Вам избегать прецедентов, которые слишком велики. Например, рассмотрите инструмент визуального моделирования. В этом приложении имеются два реальных субъекта: разработчик - кто-то, кто разрабатывает системы, используя инструмент для поддержки, и администратор системы - кто-то, кто управляет этим инструментом. Каждый из этих субъектов имеет свои собственные запросы к системе, и ему будут нужны свои наборы прецедентов.

Функциональные возможности системы определяются набором прецедентов, каждый из которых представляет некоторый поток событий. Описание прецедента определяет то, что произойдет в



системе, когда прецедент будет выполнен.

Каждый прецедент имеет собственную задачу, которую он должен выполнить. Набор прецедентов устанавливает все возможные пути использования системы. Вы можете понять задачу прецедента, просто посмотрев на его название.

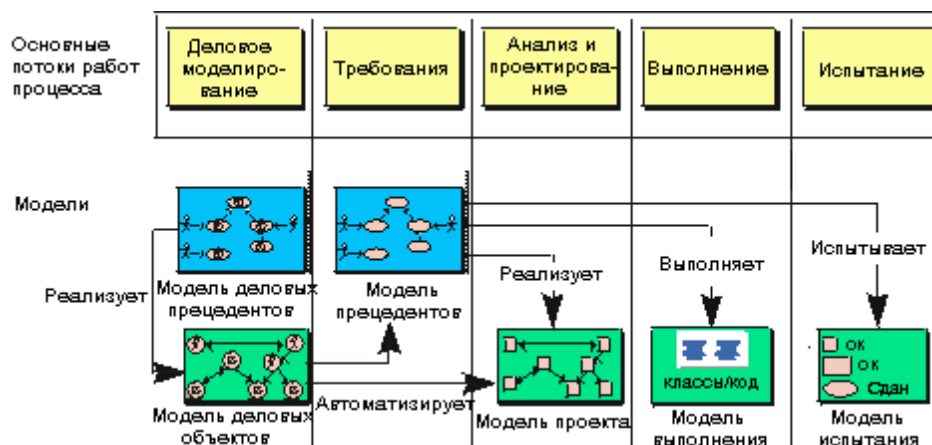


Вам уже знаком этот рисунок. Здесь указаны все прецеденты, которые выполняет банкомат по требованию клиента.

Использование прецедентов при разработке

Итак, в Rational Unified Process определенный для системы прецедент является основанием для всего процесса разработки.

Прецеденты играют роль в каждом из пяти основных потоков работ процесса: Деловое моделирование, Требования, Анализ и проектирование, Выполнение и Испытание.

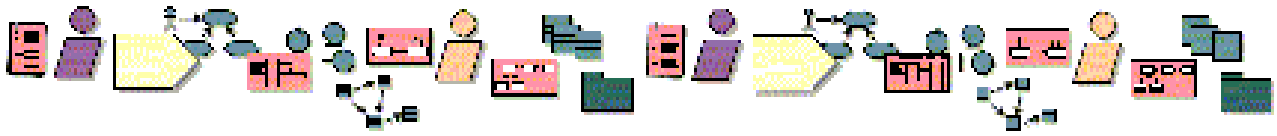


«Поток» прецедентов через различные модели

■ Деловые прецеденты определяются в ходе делового моделирования.

В управляемом прецедентами проекте Вы разрабатываете два представления деловой сферы.

Сам деловой прецедент отражает внешнее представление деловой сферы, которое определяет,



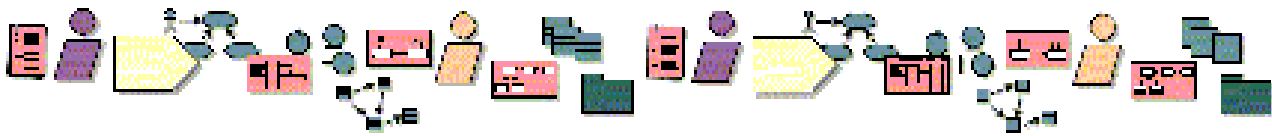
что существенно исполнить в деловой сфере, чтобы обеспечить субъекту нужные результаты. Оно определяет также, **какое** взаимодействие должны иметь деловая сфера и субъекты при выполнении делового прецедента. Такое представление должно быть разработано, когда Вы договариваетесь о том, что должно быть сделано в каждом деловом прецеденте. Совокупность деловых прецедентов **устанавливает границы системы**.

Реализация делового прецедента, с другой стороны, дает внутреннее представление делового прецедента, которое определяет, **как** должна быть организована работа, чтобы достичь нужных результатов. Реализация охватывает деловых работников и деловые сущности, которые участвуют в выполнении делового прецедента, и связи между ними, необходимые для выполнения задания. Такие представления необходимо разработать, чтобы решить, как должна быть организована работа в каждом деловом прецеденте для достижения нужных результатов.

- **Модель прецедентов** создается в потоке работ Требования. В этом потоке работ Вы нуждаетесь в прецеденте для моделирования того, что должна делать система с точки зрения пользователей. Таким образом, прецедент представляет важную фундаментальную концепцию, которая должна быть приемлема и для заказчика и для разработчиков системы.
- В потоке работ Анализ и проектирование прецеденты реализуются в модели проекта. В этой модели Вы создаете **реализации прецедента**, которые в терминах взаимодействующих объектов описывают, как прецедент выполняется. В терминах объектов проекта эта модель описывает различные части реализуемой системы и то, как должны взаимодействовать эти части, чтобы выполнить прецеденты.
- В потоке работ Выполнение **модель проекта** реализует технические требования. Поскольку прецеденты являются фундаментом модели проекта, они реализуются в терминах классов проекта.
- В потоке работ Испытания прецеденты составляют основу для идентификации **контрольных примеров** и **методов испытаний**. То есть система проверяется при выполнении каждого прецедента.

Прецеденты исполняют также и другие роли:

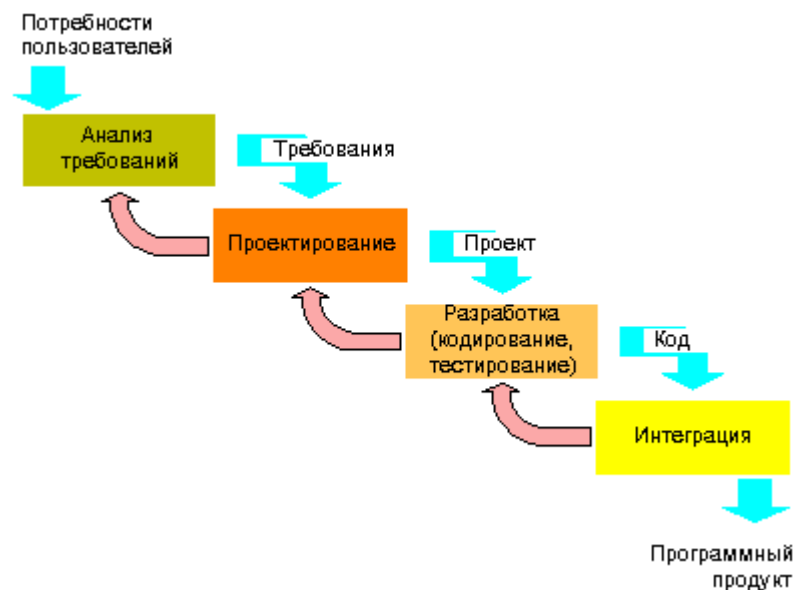
- Они используются для планирования итерационной разработки.
- Они составляют основу при написании руководств пользователей.
- Они могут использоваться как единицы упорядочения. Например, заказчик может конфигурировать систему с индивидуальным набором прецедентов.



Итеративная разработка

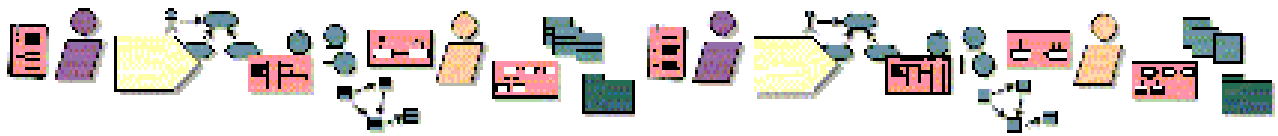
Процесс «водопада»

Сегодня стало модным винить во всех проблемах и неудачах программных проектов последовательный процесс разработки (или «водопад»). Это может показаться довольно странным. Ведь, на первый взгляд, этот подход кажется очень разумным! Вот шаги, которым Вы должны следовать:



В процессе водопада разработка ведется сверху вниз. В случае неудачи возможно возвращение к предыдущему этапу.

1. Исследуйте проблему, которая должна быть решена, все требования и их связи. Зафиксируйте их в виде задания на разработку и заставьте все заинтересованные стороны согласиться, что это именно то, что они хотят получить.
2. Разработайте проектные решения, удовлетворяющие всем требованиям и связям. Тщательно исследуйте эту конструкцию и удостоверьтесь, что все заинтересованные стороны согласны с правильностью Ваших решений.
3. Осуществите проектные решения, используя лучшие методы кодирования.
4. Убедитесь, что все заявленные требования выполнены и ваша работа удовлетворяет заказчика.
5. Передайте заказчику готовый продукт. Проблема решена!



Именно так были построены небоскребы и космические корабли. Этот способ очень хорош, но только потому, что прикладная область давно известна: за плечами инженеров сотни лет проектирования и конструирования.

Специалисты по программному обеспечению не имели такой возможности. Не имея своего, они не могли не воспользоваться положительным опытом специалистов из смежных областей.

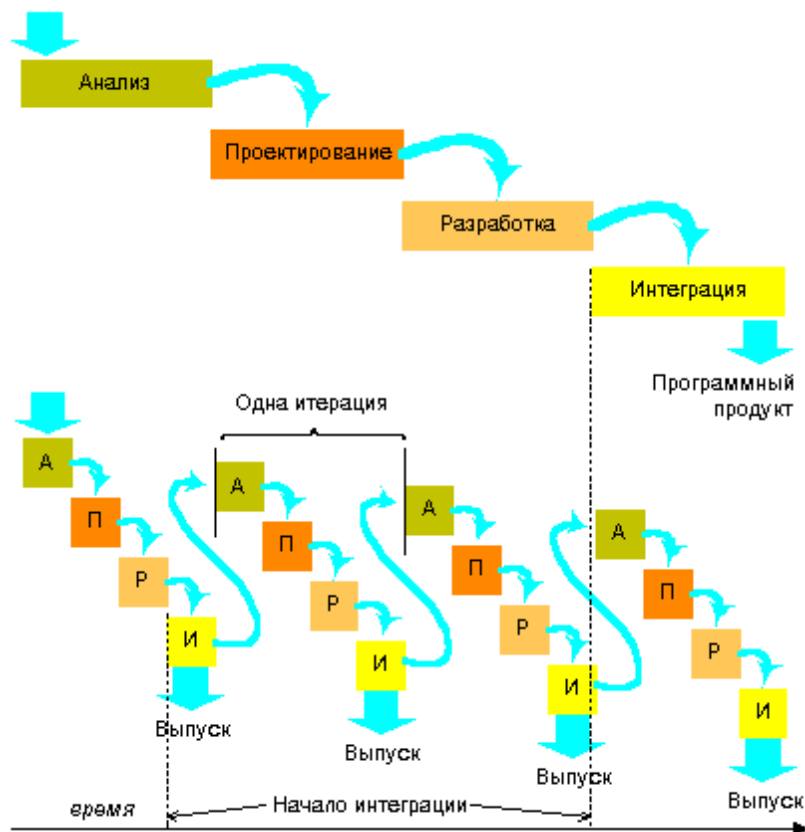
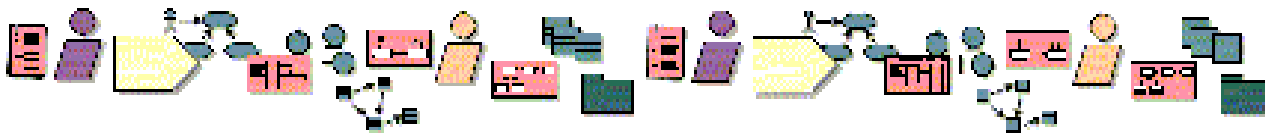
Несколько лет назад многие из нас и представить себе не могли, что процесс «водопада» не является единственным разумным подходом. А если бы и смогли, это не изменило бы ситуацию: метод «водопада» заложен во все ГОСТы и методики проектирования программных систем.

Если процесс «водопада» идеален, тогда почему большинство проектов, особенно проектов крупных систем, заканчиваются неудачей? Тому есть несколько причин.

- Контекст программирования существенно отличается от такового других технических дисциплин.
- Мы не сумели учесть некоторые факторы, связанные с человеком.
- Мы пытаемся использовать подход, четко работающий в определенных обстоятельствах, вне этих обстоятельств.
- Исторически мы все еще находимся в периоде исследований способов разработки программного обеспечения. Мы не имеем сотен лет опыта экспериментов и опыта ошибок, который превратил создание космического корабля в механический процесс. И в этом состоит главная причина неудач.

Преимущества итераций

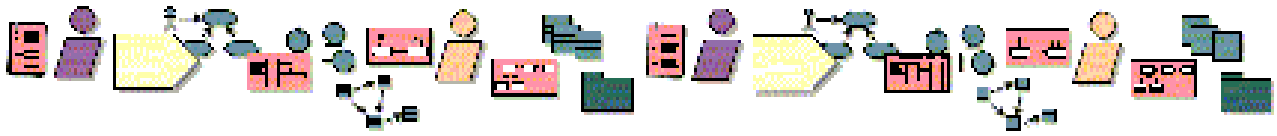
Определение итерации, как ее представляет Rational Unified Process, приводилось раньше, на [стр. 18](#). Вспомним его еще раз: «Итерация – это законченный цикл разработки, приводящий к выпуску **выполнимого изделия** (внутренней или внешней версии) или **подмножества конечного продукта**, которое возрастает с приращением от итерации к итерации, чтобы стать законченной системой.



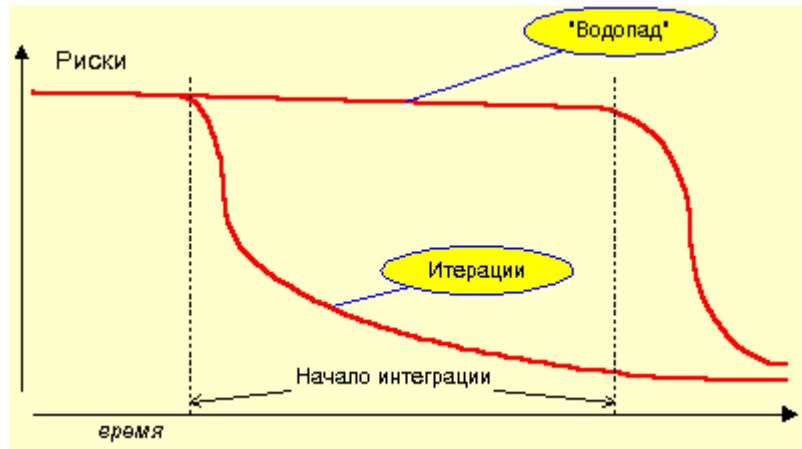
Сравните «водопад» (вверху) с итерационным подходом. Во втором случае каждая итерация начинается анализом и заканчивается выпуском изделия

Итерационный подход превосходит «водопад» по следующим причинам:

- Он позволяет принимать во внимание изменяющиеся требования. Действительность состоит в том, что требования изменяются. Изменение требований и их «ползучесть» всегда были первоисточниками неудач проектов; они ведут к опозданиям поставок, нарушениям планов, неудовлетворенности заказчиков и бесполезности разработки. Как писал Fred Brooks еще 25 лет назад: «Планируйте бросать так далеко, куда Вы добегите наверняка».
- Элементы интегрируются постепенно: интеграция - не один «большой восклицательный знак» в конце. Итерационный подход - почти непрерывная интеграция. То, что всегда было продолжительным, сомнительным и неприятным, занимая до 40 % всех усилий в конце проекта, теперь разбито на в шесть-девять раз меньшие интеграции, которые начинаются с меньшего количества элементов.
- Он позволяет Вам раньше смягчать риски, потому что интеграция, как правило, есть единственное время обнаружения или адресации рисков. Когда Вы выполняете начальные итерации, Вы проходите все основные потоки работ процесса, выявляя многие аспекты проекта: пригодность инструментальных средств и имеющегося в наличии программного



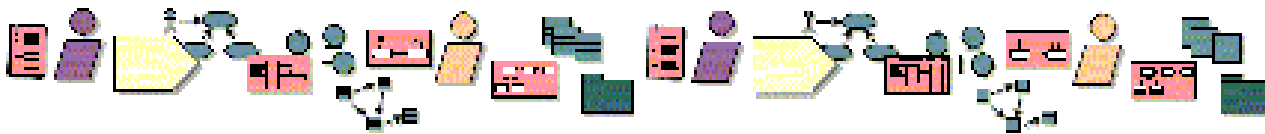
обеспечения, навыки людей и т.д. Некоторые предполагаемые риски могут не проявиться, и могут появляться новые, не предполагавшиеся ранее.



Больше всего мы рискуем, начиная проект. По мере движения вперед ожидаемый риск уменьшается. В процессе интеграции это происходит с наибольшей скоростью. В отличие от «водопада», при итеративном подходе интеграция выполняется уже на первой итерации.

Примечание: Два последних рисунка выполнены в одном временном масштабе.

- Он обеспечивает такое управление, при котором возможно делать тактические изменения в изделии; например, чтобы конкурировать с существующими изделиями. Вы можете принять решение выпустить изделие раньше с уменьшенными функциональными возможностями, чтобы противостоять продвижению конкурента, или Вы можете использовать продукт другого продавца для решения своих проблем.
- Он облегчает многократное использование компонентов - проще идентифицировать общие части, когда они частично разработаны или реализованы, чем идентифицировать все общие части с самого начала. Идентификация и разработка многократно используемых частей трудна. Критический анализ проектных решений на начальных итерациях позволяет архитекторам идентифицировать потенциальное многократное использование и разрабатывать и совершенствовать общий код в последующих итерациях.
- Он приводит к более устойчивой архитектуре - в нескольких итерациях Вы исправите больше ошибок. Слабые места обнаруживаются даже в ранних итерациях. Одновременно обнаруживаются и еще могут быть легко исправлены критические параметры эффективности, отпадает необходимость делать это накануне развертывания.
- Разработчики учатся по ходу и более полно используют различные навыки и специальные знания в течение всего жизненного цикла. Испытатели раньше запускают тесты, технические специалисты раньше делают свои записи и т.д. При не итерационной разработке те же самые люди ожидали бы начала своей работы. Это время можно было бы использовать для оттачивания своих навыков. Но потребности в обучении или в дополнительной (возможно внешней) помощи в это время, в процессе предварительной оценки, еще покрыты мраком.



- Процесс может быть улучшен и усовершенствован по ходу дела. Оценка в конце каждой итерации предусматривает не только анализ состояния изделия и перспектив выполнения графика продвижения проекта, но и того, что должно быть изменено в организации и непосредственно в процессе, чтобы улучшить его в следующей итерации.

При подходе «водопада» все выглядит высококачественным почти до конца проекта, иногда вплоть до середины интеграции. Затем все разваливается. При итерационном подходе очень трудно скрывать правду очень долго.

Организаторы проекта часто сопротивляются итерационному подходу, видя в нем своего рода бесконечное «перемалывание» сделанного. В Rational Unified Process интерактивный подход очень управляем; итерации планируются по числу, продолжительности и цели. Задачи и ответственности участников определены. Объективные критерии продвижения зафиксированы. Некоторая доработка от одной итерации до другой имеет место, но она также тщательно управляется.

Управление требованиями

Управление требованиями – это систематический подход к обнаружению, документированию, организации и сопровождению изменяющихся требований к системе.

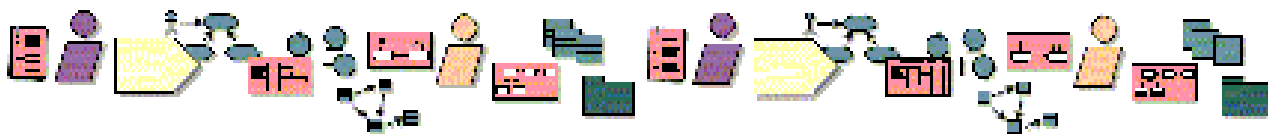
Требование – это условие или возможность, которой должна соответствовать система.

Наше формальное определение управления требованиями предопределяет систематический подход

- к обнаружению, организации и документированию требований к системе, и
- к установлению и поддержанию соглашений между заказчиком и проектной группой об изменяющихся требованиях к системе.

Сбор требований может показаться довольно тривиальной задачей. Однако, в реальных проектах Вы столкнетесь с существенными трудностями потому, что:

- Требования не всегда очевидны и могут исходить из разных источников.
- Требования не всегда удастся ясно выразить словами.
- Имеется много различных типов требований на различных уровнях детализации.
- Число требований может стать неуправляемым, если ими не управлять.
- Требования связаны друг с другом, а также с другими представлениями процесса разработки программного обеспечения.
- Требования имеют уникальные свойства или значения свойств. Например, они не являются ни одинаково важными, ни одинаково простыми для согласования.
- Есть много заинтересованных сторон и это означает, что требования должны управляться



перекрестно-функциональными группами людей.

- Требования изменяются.

Какими навыками нужно обладать, чтобы справиться этими трудностями? Наиболее важные из них это умение:

- Анализировать проблемы
- Определять потребности совладельцев
- Определять систему
- Управлять контекстом проекта
- Уточнять определение системы
- Управлять изменением требований

Кто такие совладельцы?

На интуитивном уровне это понятие можно прокомментировать следующим образом:

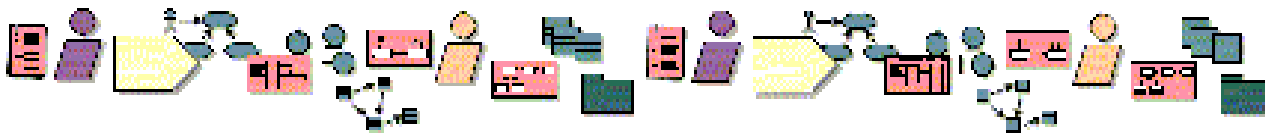
Эффективное решение любой сложной проблемы предполагает удовлетворение потребностей некоторой группы совладельцев. Совладельцы обычно имеют различные точки зрения на проблему и различные потребности, которые должны быть удовлетворены принимаемыми решениями. Многие совладельцы являются пользователями системы. Другие совладельцы - только косвенные пользователи системы или только используют результаты, на которые влияет система. К совладельцам могут быть отнесены покупатели или сторонники приобретения системы. Понимание того, кто является совладельцами и их специфических потребностей, является важным элементом при выборе эффективного решения. Примерами совладельцев являются:

- Представитель заказчика
- Представитель пользователя
- Инвестор
- Руководитель производства
- Покупатель

Анализ проблемы

Анализ проблемы выполняется для того, чтобы понять проблему, начальные потребности совладельцев и предложить решения высокого уровня.

Анализ проблемы - это акт рассуждения и анализа с целью найти «проблему позади проблемы». В ходе анализа достигается взаимное согласие по реальной проблеме и о том, кто заинтересован

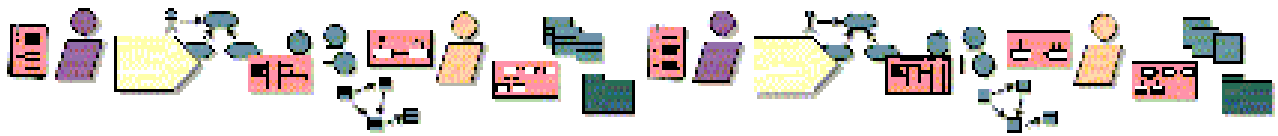


Определение потребностей совладельцев

Действия по выявлению информации могут использовать методы интервью, мозговой атаки, концептуального прототипирования, анкетных опросов и анализа конкурентоспособности. Результатом выявления потребностей может быть список запросов или потребностей, которые описаны в текстовом или графическом виде, с указанием приоритетности относительно друг друга.

Определение системы формируется путем обработки и реорганизации потребностей совладельцев. Сначала собираются требования, затем выполняется их высокоуровневый анализ и формализация. При этом уточняется специфика требований (что должно быть сделано, с какой степенью детализации, каковы приоритеты, оценки трудоемкости, технические и управленческие риски и начальный контекст). Часть этих требований, непосредственно связанных с наиболее важными запросами совладельцев, может быть представлена в виде черновых прототипов и моделей проекта.

Управление контекстом проекта



приложения. Чтобы быть уверенным, что Вы разрешили или смягчили риски в проекте как можно раньше, Вы должны разрабатывать вашу систему с приращениями, тщательно выбирая к каждому приращению требования, смягчающие известные риски в проекте. Для этого Вы должны согласовывать контекст каждой итерации с совладельцами проекта. Обычно это требует хороших навыков в управлении предполагаемыми результатами в различных стадиях разработки проекта. Вы должны также контролировать требования (как внешнее представление проекта) и непосредственно процесс разработки.

Уточнение определения системы

Детальное определение системы должно быть представлено таким образом, чтобы ваши совладельцы могли понимать его и соглашаться или не соглашаться с ним. Оно должно фиксировать не только функциональные возможности, но также и соответствие всем юридическим или нормативным требованиям, применимость, достоверность, эффективность, пригодность и надежность в эксплуатации.

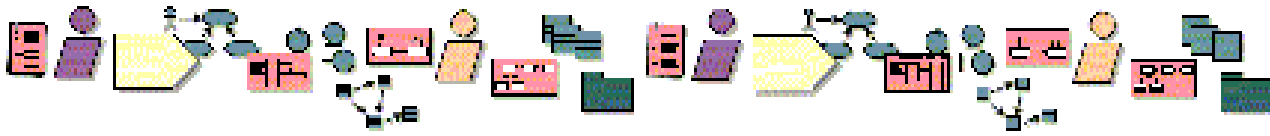
Часто совершаемая ошибка состоит в том, что Ваше ощущение комплексности разработки толкает Вас на формирование комплексного определения. Это ведет к трудностям в объяснении цели проекта и системы. Исполнители могут быть искренне увлечены работой, но они не дадут хорошей отдачи, так как у них нет полного понимания, что за продукт они производят. Вы должны приложить максимум усилий для понимания аудиторией тех документов, которые Вы производите для описания системы. Зачастую возникает потребность производить различные виды описания применительно к аудитории.

Мы видели, что методология прецедента, часто в комбинации с простыми визуальными прототипами, является очень эффективным способом указания цели системы и определения ее подробностей. Прецеденты помогают помещать требования в контекст, они сообщают историю того, как будет использоваться система.

Другой компонент детального определения системы - это констатация того, как система должна быть испытана. Планы проведения и определение предметов испытания говорят о том, какие возможности системы будут проверены.

Управление изменением требований

Независимо от того, насколько Вы осторожны при определении ваших требований, всегда будут вещи, которые потребуют изменения. Что делает изменяющиеся требования сложными для управления? Изменение требования означает не только то, что должно быть потрачено больше или меньше времени на реализацию новой конкретной возможности, но также и то, что изменение в одном требовании может вступить в конфликт с другими требованиями. Вы должны удостовериться, что Вы придаете вашим требованиям структуру, которая гибка к изменениям, и что Вы используете ссылки трассируемости для представления зависимостей между требованиями и другими артефактами (производимыми Вами искусственными объектами)



разработки. Управление изменением включает действия подобные базированию, определению зависимостей, которые важно проследить, устанавливая трассируемость между связанными предметами и управлением изменением.

Инструментальная поддержка

Rational Unified Process в рамках Rational Suite интегрирован с Rational Requisite Pro, который представляет собой инструмент для **управления требованиями**. С его помощью выполняются фиксация, организация, расположение по приоритетам и прослеживание всех требований.

Акцент на архитектуре

Rational Unified Process от начала до конца жизненного цикла управляется прецедентами, но действия проектирования сосредоточены вокруг понятия **архитектуры** – архитектуры системы, или для преимущественно программных систем, архитектуры приложения. Основной упор на ранних итерациях процесса - главным образом в стадии уточнения – делается на производство и проверку правильности **архитектуры приложения**. В начальном цикле развития архитектурные решения материализуются в форме выполняемого архитектурного прототипа, который постепенно развивается, чтобы на более поздних итерациях стать законченной системой.

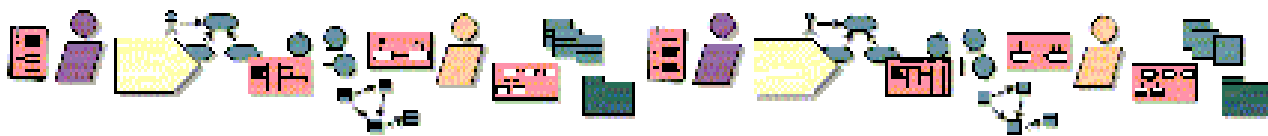
Что такое архитектура приложения?

Архитектура приложения – это понятие, которое интуитивно чувствует большинство проектировщиков, особенно опытных, и которое достаточно сложно точно определить. В частности, трудно провести четкую границу между проектом и архитектурой. Архитектура - один из аспектов проекта, который концентрируется на некоторых определенных особенностях.

Во «Введении в архитектуру приложений» David Garlan и Mary Shaw определяют архитектуру приложения как уровень проекта, отражающий «... проблемы проектирования и определения структуры комплексной системы вне алгоритмов вычислений и структур данных. Структурные проблемы включают общую организацию и структуру глобального управления, протоколы связи, синхронизации и доступа к данным, распределение функциональных возможностей между модулями, физическое распределение, композицию элементов проекта и т.п.»

Другие считают, что архитектура – это больше, чем только структура. Рабочая группа IEEE по архитектуре определяет ее как «высокоуровневую концепцию системы **в ее среде**». Это определение связывает «пригодность» с системной целостностью, экономическими связями, эстетическим восприятием и со стилем. Но оно не ограничено взглядом внутрь, а рассматривает систему в целом в ее операционной среде и среде разработки – взгляд наружу.

В Rational Unified Process архитектура системы программного обеспечения (в данном случае) – это организация или структура существенных компонентов системы, взаимодействующих через интерфейсы с компонентами, составленными из последовательно меньших компонентов и



интерфейсов.

Почему так нужен проект архитектуры?

- Наличие проекта архитектуры позволяет Вам иметь интеллектуальный контроль над проектом, управлять его сложностью и поддерживать целостность системы.

Сложная система – это нечто большее, чем сумма ее частей, чем последовательность маленьких независимых тактических решений. Она должна иметь некоторую объединяющую структуру, позволяющую организовать эти части и обеспечивать точные правила роста. Система, если не учитывать ее сложность, «взрывается» вне человеческого понимания.

Архитектура закладывает основу для понимания всего проекта, устанавливая общий набор справочников, общий словарь и т.д.

- Архитектура является эффективной базой для крупномасштабного многократного использования.

Ясно показывая крупные компоненты и критические интерфейсы между ними, архитектура позволяет Вам рассуждать о многократном использовании: о внутреннем многократном использовании - идентифицировать общие части, и о внешнем многократном использовании – объединять готовые изделия и имеющиеся в наличии компоненты. Она позволяет также многократное использование в большем масштабе: многократное использование самой архитектуры в потоке производства приложений, - т.е. в производстве, которое объединяет различные функциональные возможности в одной системе.

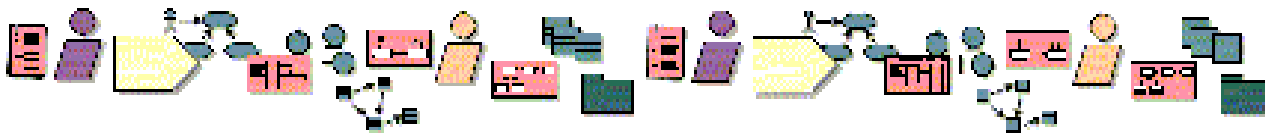
- Архитектура является базой для руководства проектом.

Планирование и персонал организуются по линиям крупных компонентов. Фундаментальные структурные решения принимаются небольшой централизованной архитектурной группой. Разработка разбивается на разделы среди маленьких групп, каждая из которых отвечает за одну или несколько частей системы.

Описание архитектуры

Архитектура приложения описывается множеством ее **представлений**. Каждое представление архитектуры отражает некоторый аспект, интересующий группу совладельцев проекта: конечных пользователей, проектировщиков, администраторов, системотехников, эксплуатационщиков и т.д.

Представления фиксируют главные решения проектирования структуры, показывая, как приложение разделено на **компоненты**, и как связаны эти компоненты для получения полезной **конфигурации**. Эти решения должны вытекать из **требований** функциональности и других факторов. С другой стороны, эти решения устанавливают дальнейшие **ограничения** на требования и на будущие проектные решения нижнего уровня.



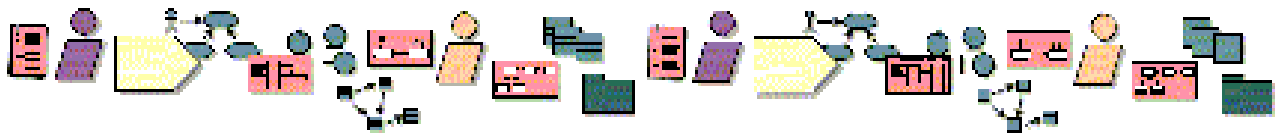
Представления архитектуры по существу являются выжимками, иллюстрирующими «архитектурно существенные» элементы моделей. В Rational Unified Process Вы начинаете с типичного набора представлений, называемого моделью представления «4+1».



Модель представления архитектуры «4+1»

Этот набор включает:

- **представление прецедентов**, которое содержит прецеденты и сценарии, охватывающие архитектурно существенное поведение, классы или технические риски. Оно является подмножеством модели прецедентов
- **логическое представление**, которое содержит наиболее важные классы проекта, их организацию в пакеты и подсистемы различных уровней. Здесь содержится уже некоторая реализация прецедента. Это представление является подмножеством модели проекта
- **представление выполнения**, которое содержит краткий обзор модели выполнения в терминах модулей пакетов и уровней. Здесь описывается также распределение пакетов и классов (из логического представления) в пакетах и модулях представления выполнения. Это представление является подмножеством модели выполнения
- **представление процесса**, которое содержит описание задач (процессов и нитей), их взаимодействия и конфигурации и распределения объектов и классов по задачам. Потребность этого представления возникает только, если система имеет существенную степень параллелизма. В Rational Unified Process 5.1 представление процесса - это подмножество модели проекта



- **представление развертывания**, которое содержит описание различных физических узлов для наиболее типичных конфигураций платформы, и расположение задач (из представления процесса) по физическим узлам. Потребность этого представления возникает только, если система распределена.

В Rational Unified Process представления архитектуры регистрируются в документе **Архитектура приложения**.

Вы можете предлагать дополнительные представления для выражения различных специальных потребностей: представление интерфейса пользователя, представление защиты, представление данных и так далее. Для простых систем Вы можете опустить некоторых из представлений, содержащихся в «модели вида 4+1».

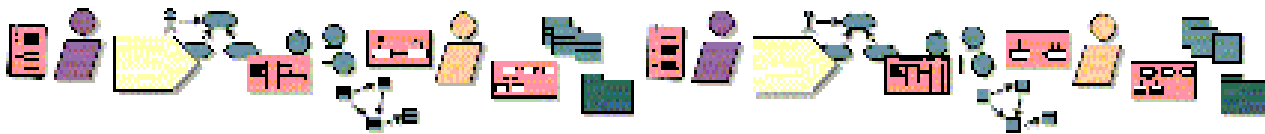
Круг интересов архитектуры

Перечисленные выше представления могут изобразить проект системы в целом. Но архитектура интересуется только некоторыми определенными аспектами:

- **Структура** модели - организационные элементы, например, иерархическое представление.
- **Существенные элементы** - критические прецеденты, главные классы, общие механизмы и так далее, а не все элементы, представленные в модели.
- Несколько **ключевых сценариев**, показывающих главный поток управления в системе.
- **Сервис**, чтобы зафиксировать модульность, необязательные особенности, аспекты производственной линии.

В сущности, архитектурные представления - это **абстракции** или упрощения полного проекта, в которых важные характеристики сделаны более яркими, подробности оставлены в стороне. Эти характеристики важны при рассуждении относительно

- развития системы – переходу к следующему циклу разработки
- многократного использования архитектуры или ее частей в контексте производственной линии
- оценки дополнительных качеств типа эффективности, доступности, взаимозаменяемости и безопасности
- распределения проектных работ между группами или субподрядчиками
- решения о включении имеющихся в наличии компонентов
- включения в более широкую систему.



Методическая поддержка

Rational Unified Process предлагает систематический способ проектирования, разработки и проверки правильности архитектуры. Он предоставляет шаблоны для описания архитектуры и для сбора данных, касающихся архитектуры. Поток работ проектирования содержит специальные действия, нацеленные на идентификацию архитектурных связей и существенных архитектурных элементов, а также рекомендации по принятию архитектурных решений. Поток работ управления проектом показывает, как планирование ранних итераций учитывает проектирование архитектуры и ликвидацию главных технических рисков.

Шаблоны архитектуры – это готовые формы, которые решают повторяющиеся архитектурные проблемы. **Архитектурный каркас** или **архитектурная инфраструктура** - набор компонентов, из которых Вы можете формировать некоторый вид архитектуры. Многие из архитектурных трудностей были преодолены путем использования каркасов или инфраструктур, обычно связанных с определенной областью: управление и контроль, административные информационные системы и т.д.

Архитектура приложения или только какое-то представление архитектуры может иметь атрибут под названием **архитектурный стиль**, который предоставляет набор возможных форм, что позволяет применить к архитектуре некоторую степень унификации. Стиль может быть определен набором образцов или выбором определенных компонентов или разъемов как базовых строительных блоков. Для данной системы некоторый стиль может быть зафиксирован как раздел описания архитектуры в **руководстве по архитектурному стилю** – отдельной части документации по руководящим принципам проектирования в Rational Unified Process. Стиль играет важную роль в понятности и целостности архитектуры.

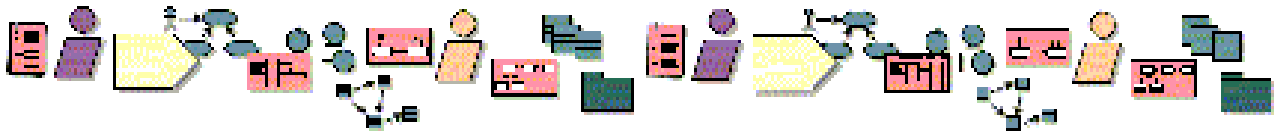
Архитектура в Rational Unified Process – это, прежде всего, результат потока работ анализа и проектирования. Так как проект повторяет этот поток работ, итерацию за итерацией, архитектура развивается, усовершенствуется и приглаживается. Поскольку каждая итерация включает интеграцию и тестирование, ко времени поставки архитектура будет весьма надежна. Архитектура является главным делом итераций стадии разработки, в конце которой архитектура обычно бывает сформирована.

Инструментальная поддержка

Выше уже упоминалось, что Rational Unified Process в рамках Rational Suite интегрирован с Rational Rose, который представляет собой инструмент моделирования с использованием системы обозначений UML.

Для графического описания поясненных выше представлений архитектуры используются следующие диаграммы UML:

- **Представление прецедентов:** диаграммы прецедентов, изображающие прецеденты, субъекты и обычные классы проекта; диаграммы последовательности, изображающие



объекты проекта и их взаимодействия.

- **Логическое представление:** диаграммы классов, диаграммы состояний и диаграммы объектов.
- **Представление выполнения:** диаграммы компонентов.
- **Представление процесса:** диаграммы классов и диаграммы объектов, затрагивающих задачу - процессы и нити.
- **Представление развертывания:** диаграммы развертывания.

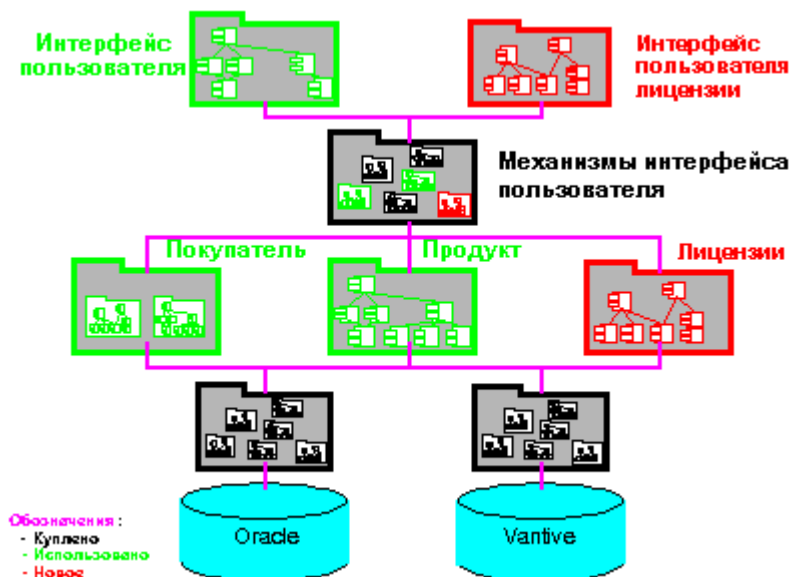
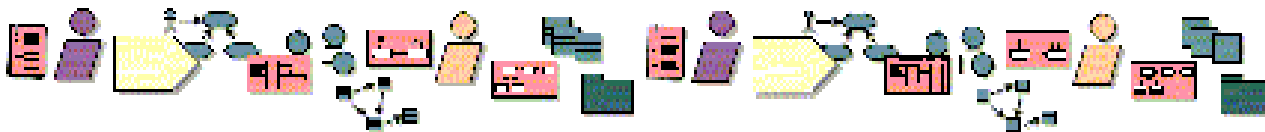
Разработка на базе компонентов

Программный компонент может быть определен как нетривиальная часть программного обеспечения, модуля, пакета или подсистемы, которая выполняет ясную функцию, имеет ясную границу и может быть интегрирована в четко определенной архитектуре. Это - физическая реализация абстракции в вашем проекте.

Компоненты появляются из различных источников:

- Внутри модульной архитектуры Вы идентифицируете, изолируете, проектируете, разрабатываете и тестируете правильно построенные компоненты. Эти компоненты могут быть индивидуально проверены и постепенно интегрированы в цельную систему.
- Кроме того, некоторые из этих компонентов могут быть разработаны так, чтобы их можно было перенастраивать, особенно компоненты, обеспечивающие общие решения в широких пределах общих проблем. Эти перенастраиваемые компоненты, которые могут быть больше, чем просто совокупностями утилит или библиотеками классов, образуют базу для многократного использования в пределах организации, увеличивая производительность и качество программирования.
- Недавнее появление большого числа коммерчески успешных компонентных инфраструктур, типа CORBA, Internet, ActiveX или JavaBeans, вызвало бурный рост индустрии производства компонентов для различных прикладных областей, позволяя Вам покупать и интегрировать компоненты вместо разработки их внутри организации.

Первое появление эксплуатирует старые концепции модульности и инкапсуляции, на шаг далее продвигая концепции, лежащие в основе объектно-ориентированной технологии. Два последних переориентируют разработку программ от программирования к сборке программного обеспечения (трансляции компонентов).



В архитектуре этого приложения используются вновь разработанные, модифицированные старые и покупные компоненты (они обозначены разными цветами)

Rational Unified Process поддерживает компонентно-ориентированную разработку в нескольких направлениях:

- Итерационный подход позволяет Вам постепенно идентифицировать компоненты, решая какой из них разрабатывать, какой многократно использовать и который купить.
- Акцент на архитектуре программы позволяет Вам соединять структуру (компоненты) и способы их интеграции (фундаментальные механизмы и характер их взаимодействия).
- Концепции пакетов, подсистем и уровней используются в процессе анализа и проектирования для организации компонентов и определения их интерфейсов.
- Тестирование организуется сначала для одиночных компонентов, затем для все более крупных наборов интегрируемых компонентов.

Настраиваемый процесс

Rational Unified Process является достаточно общим и совершенным, чтобы использоваться широким кругом разработчиков программного обеспечения. Во многих случаях этот процесс должен быть изменен, откорректирован, расширен и приспособлен к определенным характеристикам, связям и традициям конкретной организации.

Элементы процесса, которые могут изменяться, настраиваться, добавляться или исключаться, включают: артефакты, действия, потоки работ и работников.

С возможностями настройки процесса к нуждам конкретного проекта и конкретной организации мы познакомимся более подробно при рассмотрении потока работ «Среда».