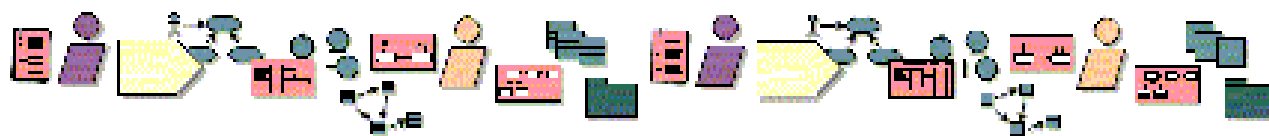


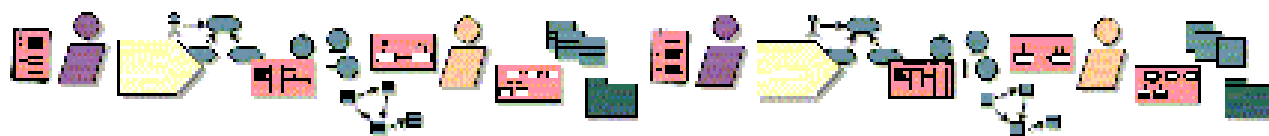
Содержание

Предисловие	1-1
Что такое Rational Unified Process?	2-1
Rational Unified Process как технология	2-1
Rational Unified Process как продукт	2-3
Архитектура процесса	3-1
Два измерения	3-1
Статическое содержание процесса	3-2
Структура жизненного цикла	3-6
Начальная стадия	3-7
Стадия уточнения	3-7
Стадия конструирования	3-8
Стадия перехода	3-8
Итерации	3-8
Особенности Rational Unified Process	4-1
Визуальное моделирование	4-1
Субъекты	4-2
Прецеденты	4-2
Объекты	4-3
Классы	4-3
Обобщения	4-3
Модели	4-3
Инструментальная поддержка	4-4
Управление прецедентами	4-4
Что такое прецедент?	4-5
Использование прецедентов при разработке	4-6

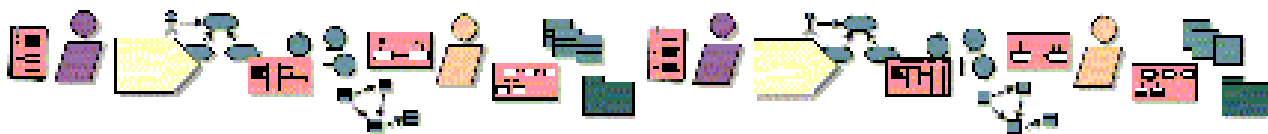


Итеративная разработка	4-8
Процесс «водопада»	4-8
Преимущества итераций	4-9
Управление требованиями	4-12
Кто такие совладельцы?	4-13
Анализ проблемы	4-13
Определение потребностей совладельцев	4-14
Определение системы	4-14
Управление контекстом проекта	4-14
Уточнение определения системы	4-15
Управление изменением требований	4-15
Инструментальная поддержка	4-16
Акцент на архитектуре	4-16
Что такое архитектура приложения?	4-16
Почему так нужен проект архитектуры?	4-17
Описание архитектуры	4-17
Круг интересов архитектуры	4-19
Методическая поддержка	4-20
Инструментальная поддержка	4-20
Разработка на базе компонентов	4-21
Настраиваемый процесс	4-22
Начало работы с Rational Unified Process	5-1
Средства навигации Rational Unified Process	5-1
Кнопка на поле страницы	5-2
Окно просмотра древовидной структуры	5-3
Кнопки поиска	5-3
Таблица краткого обзора	5-3
Изучение процесса	5-4
Внедрение Rational Unified Process	5-5
Инструментальные средства поддержки	5-6
Поток работ «Управление проектом»	6-1
Краткий обзор действий	6-3

Работник: Руководитель проекта	6-3
Действие: Изложение деловых обстоятельств	6-3
Действие: Идентификация рисков	6-4
Действие: Разработка плана проекта	6-4
Действие: Формирование проектной группы	6-5
Действие: Разработка плана итерации	6-7
Действие: Выполнение плана итерации	6-8
Действие: Оценка итерации	6-8
Действие: Ревизия списка рисков	6-9
Артефакты управления проектом	6-11
Концепция рисков в Rational Unified Process	6-11
Введение	6-11
Определения	6-12
Инициализация списка рисков	6-12
Группировка и ранжирование рисков	6-13
Стратегии поведения	6-14
Концепция метрик в Rational Unified Process	6-14
Зачем нам метрики?	6-14
Что такое метрика?	6-16
Действия с метриками	6-17
Поток работ «Управление проектом» (продолжение)	7-1
О планирование итерационного проекта	7-1
Двухуровневое планирование	7-1
План проекта	7-2
План итерации	7-2
Разработка плана проекта	7-3
Определение главных вех проекта	7-4
Определение целей вех	7-5
Определение количества и длины итераций в пределах стадий	7-7
Уточнение сроков вех и контекста	7-12
Определение плана измерений	7-12
Разработка плана итерации	7-13
Определение рамок итерации	7-13
Определение критериев оценки итерации	7-15
Определение действий итерации	7-16



Распределение ответственности	7-19
Инструментальная поддержка управления проектом	7-19
Шаблоны Microsoft Word	7-19
Использование Microsoft Project	7-20
Использование Rose Planner Link	7-22
Выводы	7-22
 Поток работ «Деловое моделирование»	 8-1
Введение	8-1
Цели делового моделирования	8-1
Сферы применения делового моделирования	8-2
Язык моделирования	8-3
Основные понятия	8-4
Модель деловых прецедентов	8-5
Деловой субъект	8-5
Деловой прецедент	8-6
Связь ассоциации в модели деловых прецедентов	8-6
Обобщение субъектов в модели деловых прецедентов	8-7
Связь включения в модели деловых прецедентов	8-8
Связь расширения в модели деловых прецедентов	8-9
Обобщение прецедентов в модели деловых прецедентов	8-10
Модель деловых объектов	8-10
Деловой работник	8-11
Деловая сущность	8-11
Связь ассоциации в модели деловых объектов	8-12
Связь объединения в модели деловых объектов	8-12
Связь обобщения в модели деловых объектов	8-13
Организационный модуль	8-15
Реализация делового прецедента	8-15
Диаграмма прецедентов	8-16
Диаграмма действий	8-16
Диаграмма классов	8-18
Диаграмма сотрудничества	8-18
Диаграмма последовательности	8-19
Диаграммы состояний	8-20
Краткий обзор действий	8-21
Работник: Аналитик делового процесса	8-21



Действие: Фиксация общего словаря	8-21
Действие: Поиск деловых субъектов и прецедентов	8-22
Действие: Структурирование модели деловых прецедентов	8-22
Работник: Проектировщик предметной области	8-22
Действие: Детализация делового прецедента	8-23
Действие: Поиск деловых работников и сущностей	8-23
Действие: Детализация делового работника	8-23
Действие: Детализация деловой сущности	8-24
Работник: Рецензент деловой модели	8-24
Действия: Экспертиза моделей деловых прецедентов и деловых объектов	8-24

Поток работ «Деловое моделирование» (продолжение)..... 9 -1

Пример моделирования: начало	9 -1
------------------------------------	------

Модель деловых прецедентов 9 -3

Поиск деловых субъектов	9 -3
Пример моделирования: деловые субъекты	9 -5
Поиск деловых прецедентов	9 -7
Пример моделирования: деловые прецеденты	9 -10
Описание схемы потока работ делового прецедента	9 -11
Пример моделирования: описания потоков работ	9 -11

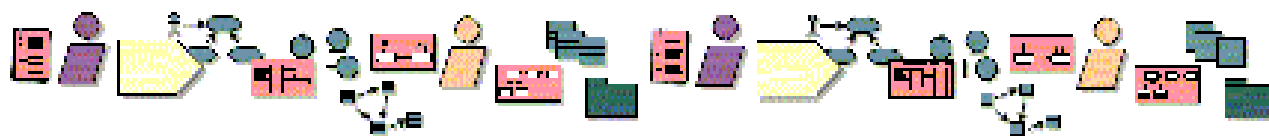
Поток работ «Деловое моделирование» (окончание) 10-1

Модель деловых объектов 10-1

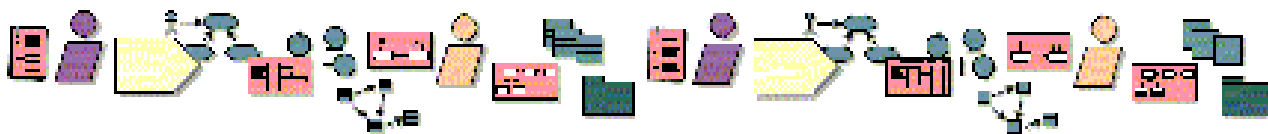
Идентификация организационных модулей	10-1
Пример моделирования: Структура модели деловых объектов	10-2
Поиск деловых работников	10-3
Пример моделирования: Деловые работники	10-5
Идентификация деловых сущностей	10-8
Пример моделирования: Деловые сущности	10-9
Определение реализаций деловых прецедентов	10-14
Деловые объекты в отношении к деловым прецедентам	10-15
Модель деловых объектов и информационные системы	10-15
Подробности информационной системы в модели деловых объектов	10-16

Переход от деловых моделей к системным 10-17

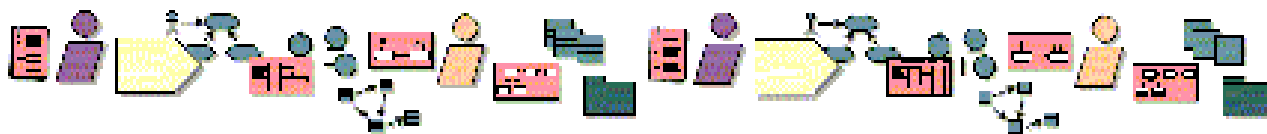
Деловые модели и архитектура системы	10-18
Деловые модели и субъекты в системе	10-19
Деловые модели и классы сущностей в модели анализа	10-20
Трансляция взаимодействий между работниками в требования к системе	10-20

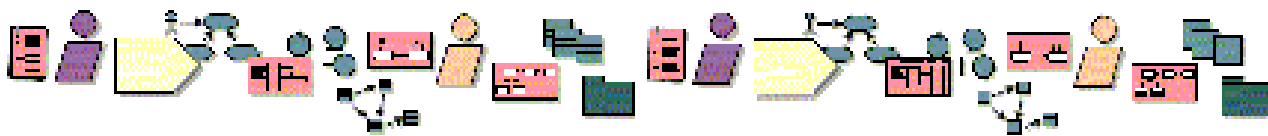


Различные контексты делового моделирования	10-20
Итоговая таблица	10-21
Выводы	10-22
 Поток работ «Требования»	 11-1
Введение	11-1
Цели потока работ	11-1
Требования	11-2
Трассируемость требований	11-4
Краткий обзор потока работ	11-6
Анализ проблемы	11-7
Понимание потребностей совладельцев	11-9
Определение системы	11-11
Управление контекстом системы	11-14
Уточнение определения системы	11-16
Управление изменением требований	11-18
 Поток работ «Требования» (продолжение)	 12-1
Артефакты управления требованиями	12-1
Работник: Системный аналитик	12-2
Артефакт: Глоссарий	12-2
Пример артефакта: Введение в RUP. Глоссарий	12-5
Артефакт: Видение	12-5
Пример артефакта: Система обработки и выполнения заказов.	
Видение	12-12
 Поток работ «Требования» (продолжение 2)	 13-1
Артефакт: Запросы совладельцев	13-1
Артефакт: Модель прецедентов	13-3
Артефакт: Дополнительная спецификация	13-15
Артефакт: Атрибуты требований	13-18
 Поток работ «Требования» (продолжение 3)	 14-1
Артефакт: Субъект	14-1
Артефакт: План управления требованиями	14-9
Работник: Автор прецедента	14-11
Артефакт: Прецедент	14-11



Артефакт: Пакет прецедентов	14-27
Поток работ «Требования» (окончание)	15-1
Артефакты интерфейса пользователя	15-1
Работник: Проектировщик интерфейса пользователя	15-1
Артефакт: Иллюстрированный сценарий прецедента	15-3
Артефакт: Граничный класс	15-15
Артефакт: Прототип интерфейса пользователя	15-20
Выводы	15-24
Поток работ «Среда»	16-1
Введение	16-1
Цели потока работ	16-1
Действия внедрения среды	16-2
Внедрение процесса в организации	16-7
Пошаговая процедура	16-7
Артефакты	16-13
Различные подходы к внедрению	16-13
Организация работы	16-17
Трактовка внедрения процесса как проекта	16-17
Рекомендации по планированию внедрения среды	16-19
Поток работ «Среда» (Продолжение)	17-1
Внедрение процесса в проекте	17-1
Приспосабливание процесса	17-2
Общие рекомендации по планированию	17-4
Подходы к внедрению процесса и инструментальных средств	17-5





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.

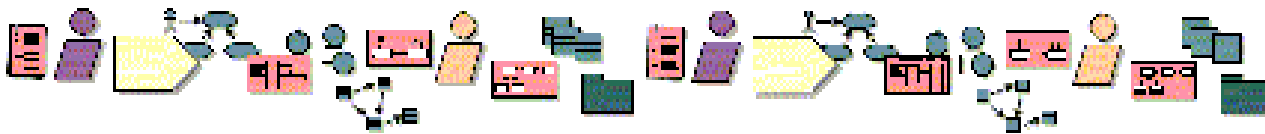
Традиционные способы разрешения проблемы сложности, такие как увеличение коллективов разработчиков, их специализация, распределение работ в чистом виде, ведут к еще большим трудностям согласования результатов и сборки готовых систем.

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.

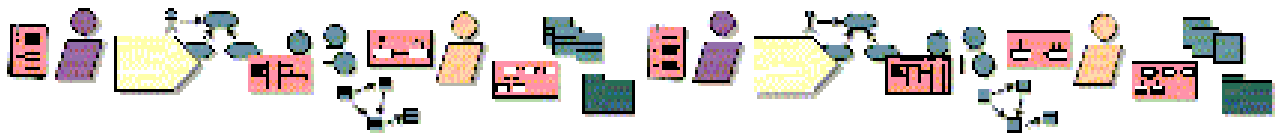
Статьи цикла предполагается публиковать приблизительно раз в две недели. Формат публикации (.pdf) позволяет устанавливать гиперссылки внутри и между статьями. Поэтому, при появлении новых статей, возможно появление новых версий уже опубликованных. Это может быть связано как с появлением ссылок, так и 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...

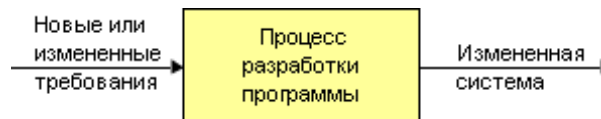


Что такое Rational Unified Process?

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.

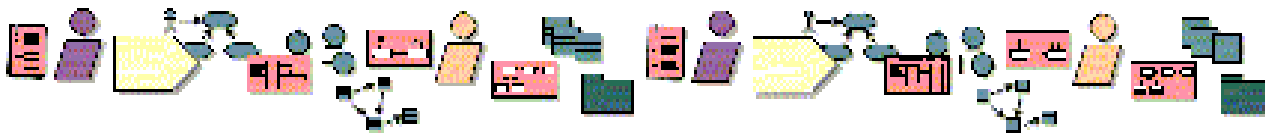


Процесс разработки программного обеспечения - процесс разработки системы из требований, новых (начальный цикл развития) или измененных (цикл эволюции).

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**.

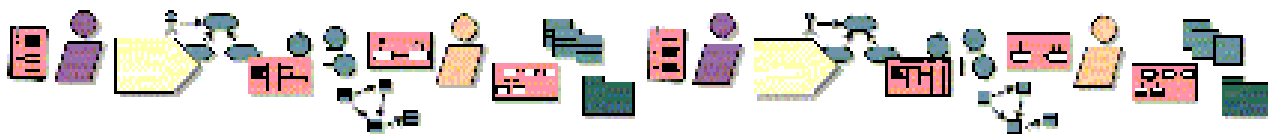
Rational Unified Process обеспечивает строгий подход к назначению задач и ответственности в пределах группы разработки. Его **цель** состоит в том, чтобы гарантировать высокое качество программного продукта, отвечающего потребностям конечных пользователей, в пределах предсказуемого временного графика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

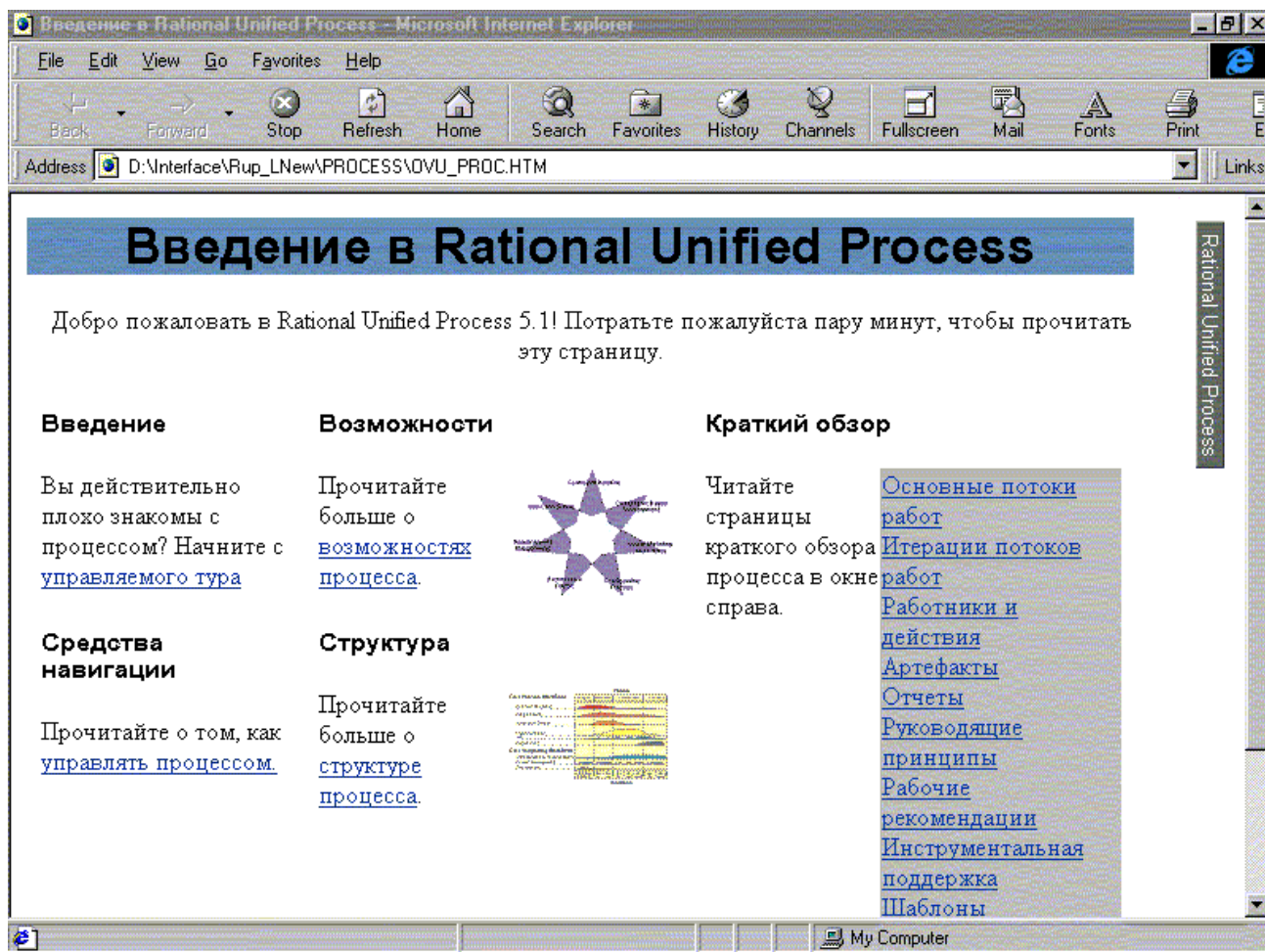
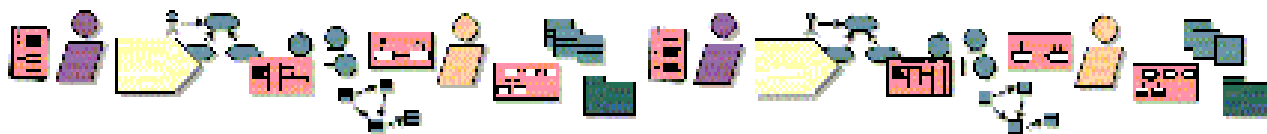
Продукт Rational Unified Process входит в состав Rational Suite всех комплектаций. В поставку входит:

- 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.

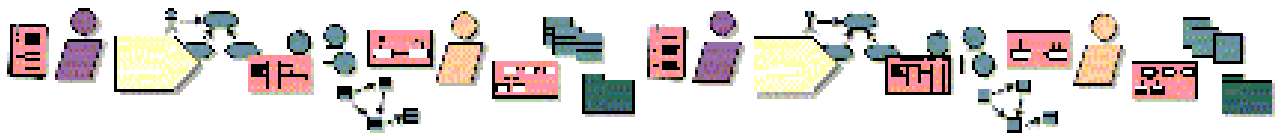
Интерактивная версия содержит настраиваемые шаблоны, которые позволяют использовать информацию, порожденную в процессе работы над проектом, для создания отчетных документов с использованием Microsoft Word, Microsoft Project и Microsoft FrontPage.

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.

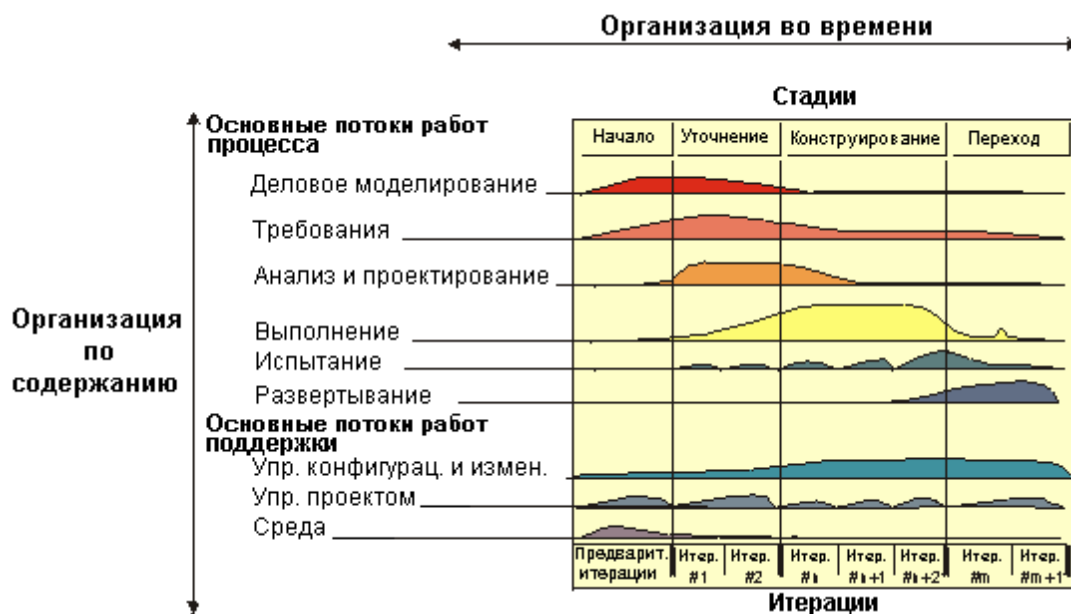


Архитектура процесса

Два измерения

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

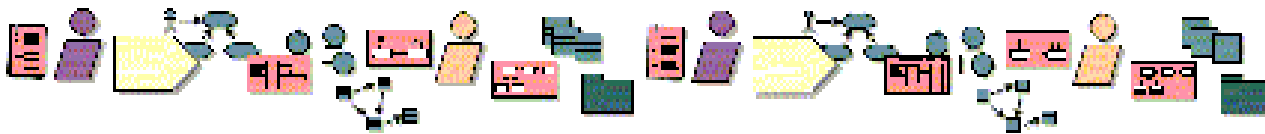
- По содержанию действий участников групп разработки (по основным потокам работ).
- Во времени (по стадиям жизненного цикла разрабатываемой системы).



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:

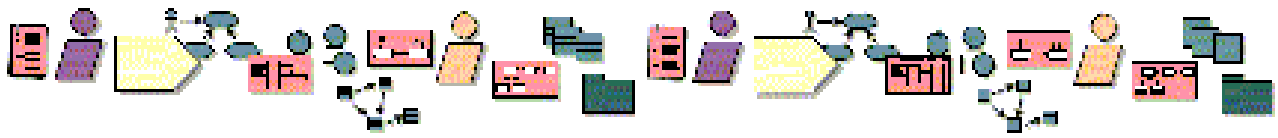
- Деловое моделирование
- Требования
- Анализ и проектирование
- Выполнение
- Испытание
- Развертывание

и три потока работ поддержки:

- Управление конфигурацией и изменением
- Управление проектом
- Среда

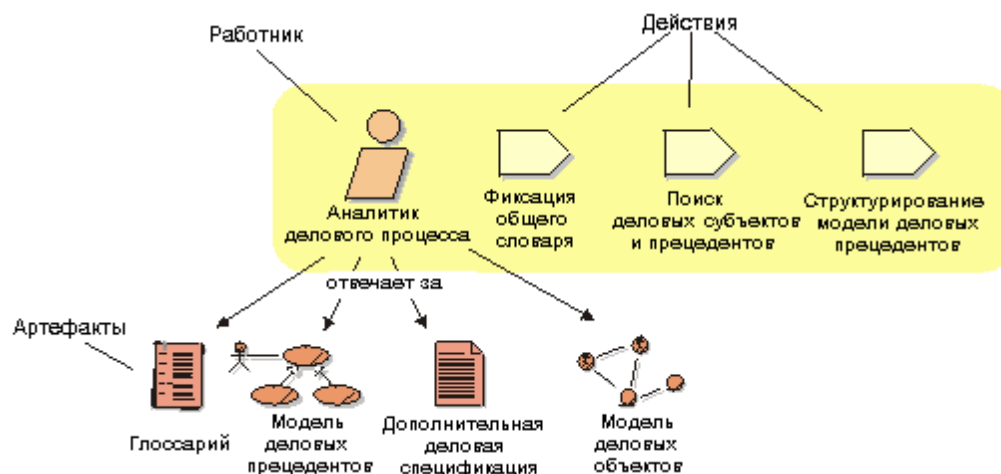
Основные потоки работ описаны в терминах работников, действий и артефактов.

- **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.



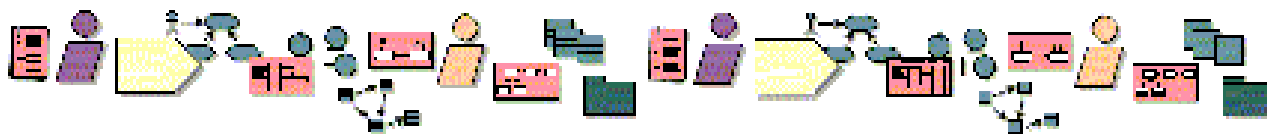
- **Действие** – это самая маленькая часть работы, которая относится к делу; его можно интерпретировать как «техническую операцию» работника. Далее, невозможно выполнить только часть действия, хотя в пределах действия может существовать некоторая необязательная операция. Такое разделение работы облегчает 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.



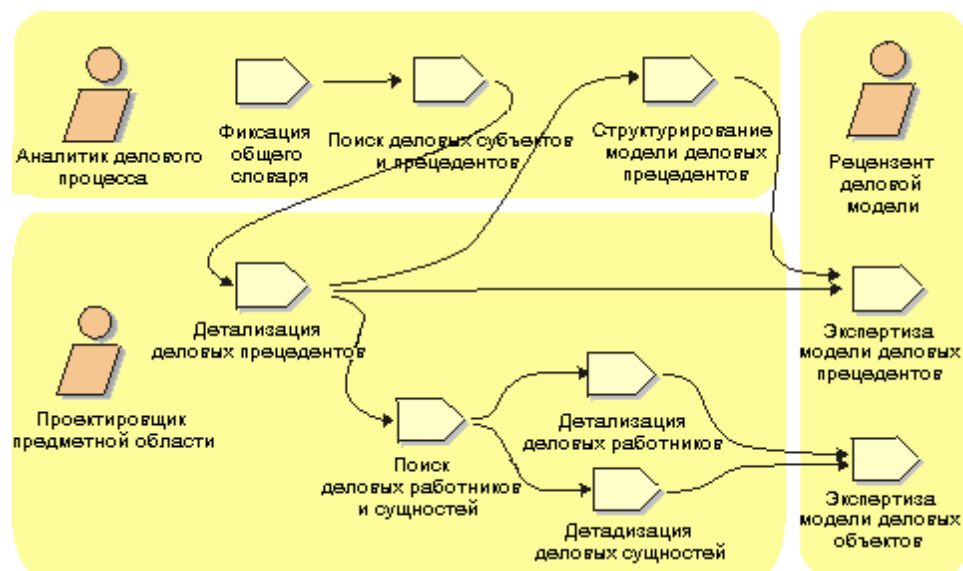
Каждый работник имеет свой собственный набор действий и артефактов

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.

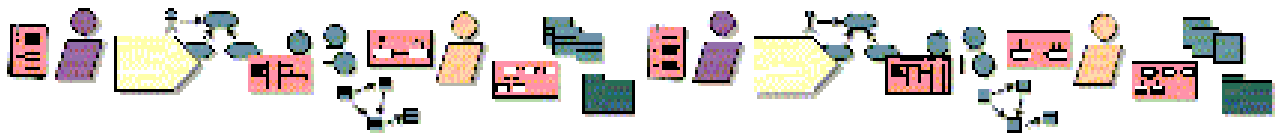


Через связанный набор действий работник неявно определяет также и навыки, необходимые для исполнения работ.

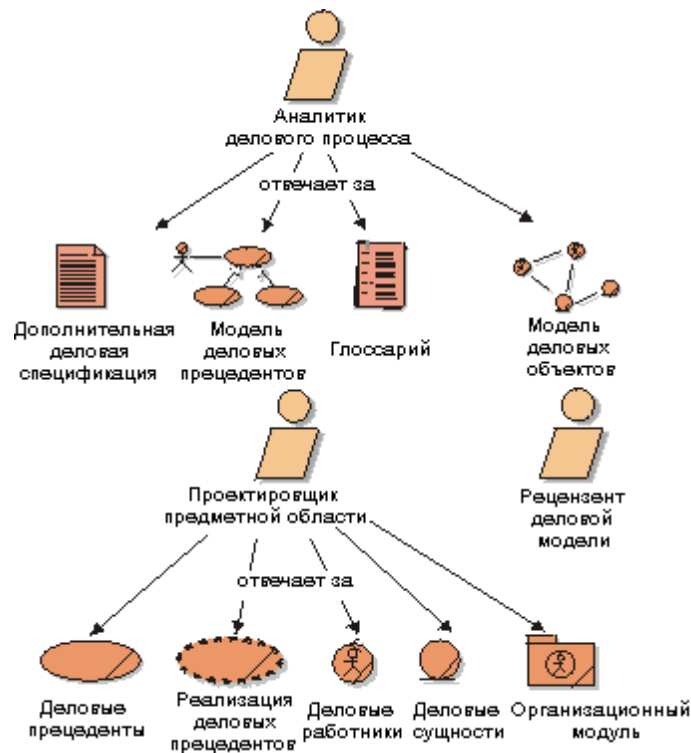
Для каждого основного потока работ представляется диаграмма **краткого обзора действий**. Эта диаграмма показывает все действия и всех работников, включенных в поток работ.



Типовая диаграмма краткого обзора действий (поток работ делового моделирования)



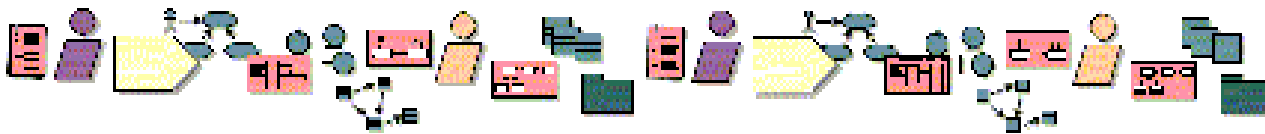
Для каждого основного потока работ представлена диаграмма **краткого обзора артефактов**. Эта диаграмма показывает все артефакты и всех работников, включенных в поток работ.



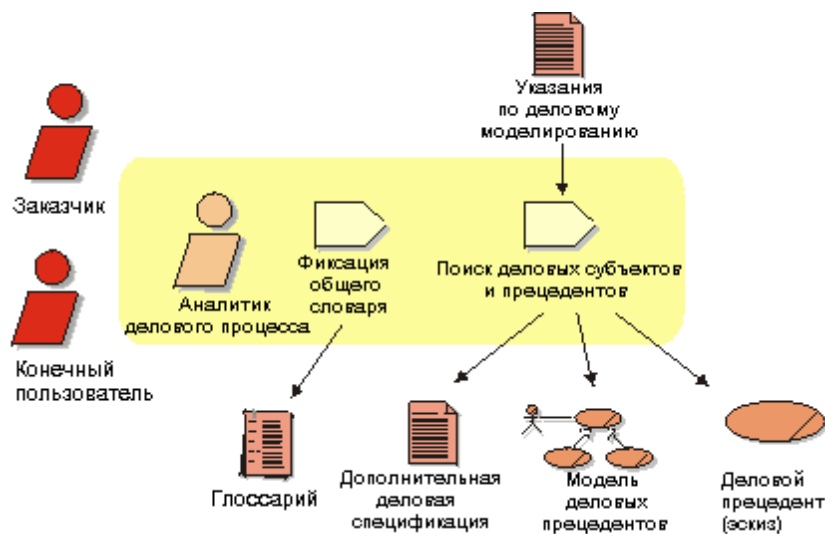
Типовая диаграмма краткого обзора артефактов (поток работ делового моделирования)

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.



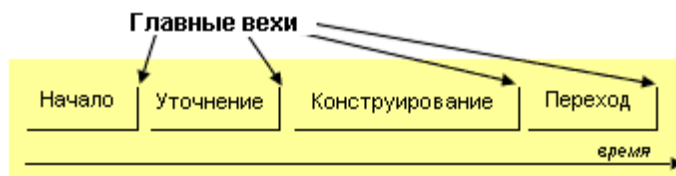
Типовая диаграмма деталей потока работ (поток работ делового моделирования)

Структура жизненного цикла

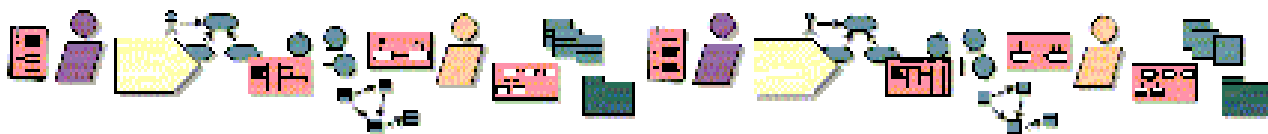
Когда мы рассматриваем динамическую организацию процесса во времени, жизненный цикл программы разбивается на циклы, каждый из которых работает над новым поколением изделия. Rational Unified Process делит один цикл развития на четыре последовательных стадии:

- Начало
- Уточнение
- Конструирование
- Переход

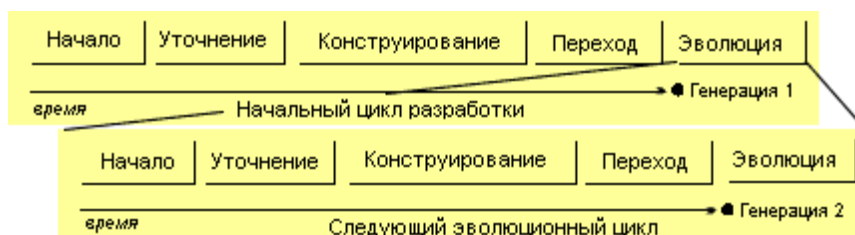
Каждая стадия заканчивается четко определенной вехой – временной точкой, в которой должны быть приняты некоторые критические решения, а поэтому ключевые цели должны быть достигнуты.



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

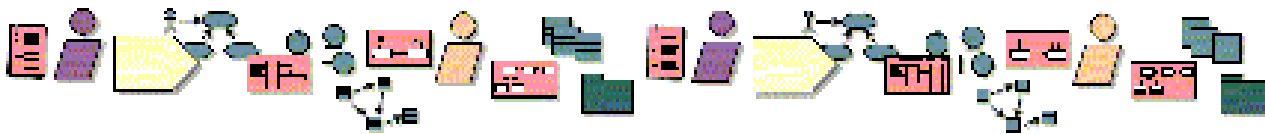
На начальной стадии Вы устанавливаете деловые применения системы и определяете рамки проекта. Чтобы сделать это, Вы должны идентифицировать все внешние объекты, с которыми взаимодействует система (субъекты), и определить характер этого взаимодействия на высоком уровне. Эта работа включает идентификацию 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.

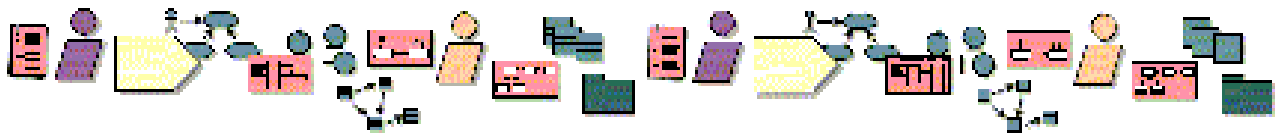
Итерации

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.

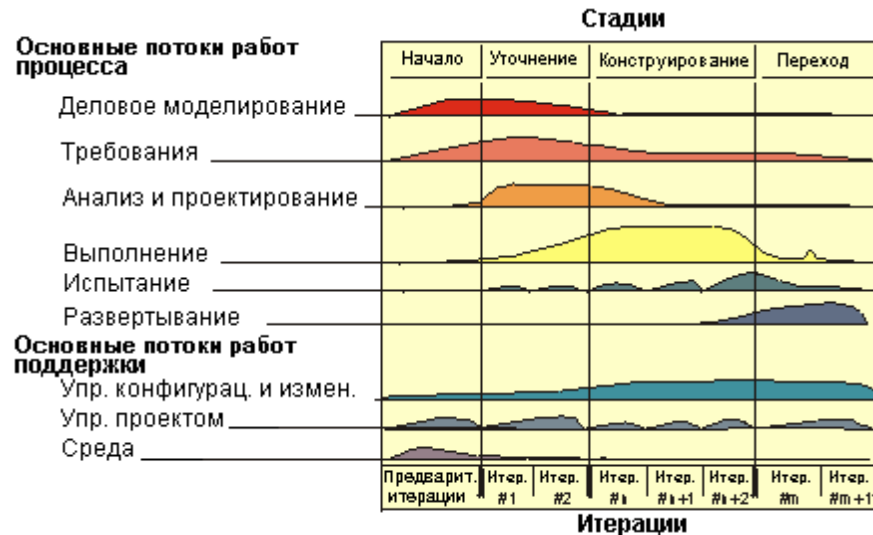


Каждая итерация в пределах стадии приводит к выпуску выполнимой системы.

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.

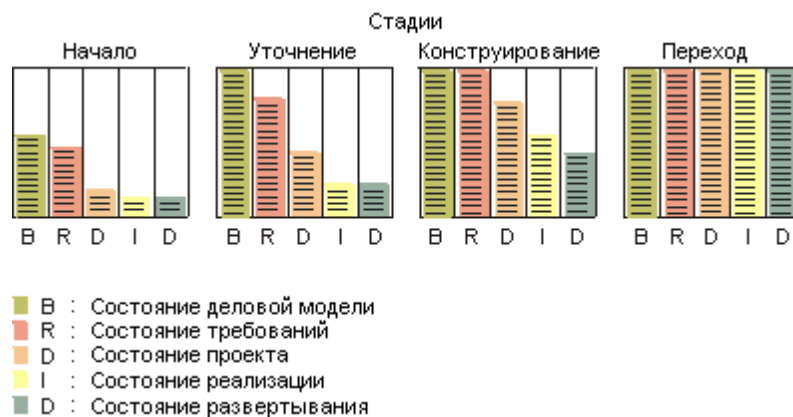


Это положение иллюстрируется следующей диаграммой, где показана трудоемкость каждого из основных потоков работ по мере Вашего продвижения от итерации к итерации через все четыре стадии.



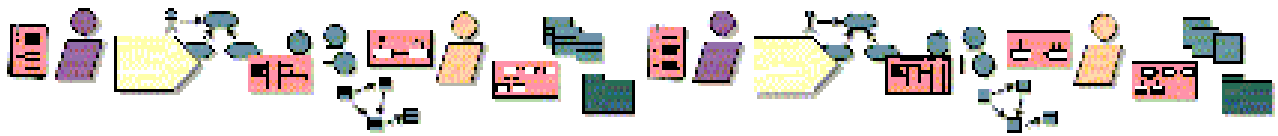
Стадии состоят из итераций, содержащих все основные потоки работ. Каждый из основных потоков работ ответственен за набор артефактов. Высота кривых отражает характер интенсивности потока работ на стадиях и итерациях.

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.





Особенности Rational Unified Process

Мы уже обсуждали кратко особенности технологии Rational Unified Process ([см. стр. 2-1 – 2-3](#)). Теперь рассмотрим их более подробно.

Рисунок внизу иллюстрирует основные идеи Rational Unified Process. Описание каждой из них поможет Вам понять основание для соответствующей концепции Rational Unified Process и способы ее реализации. Это, в свою очередь, должно помочь Вам впоследствии более эффективно использовать Rational Unified Process.



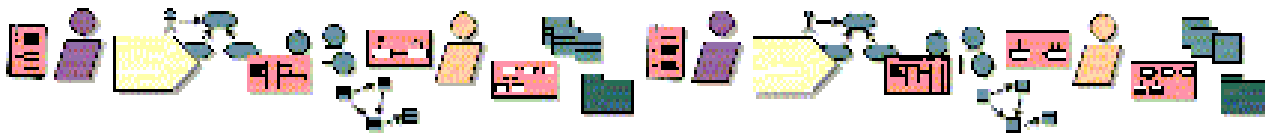
Каждый луч этой звезды представляет основополагающую идею Rational Unified Process.

Примечание: Не забывайте об интерактивных возможностях наших иллюстраций!

Визуальное моделирование

Множество проектов используют сегодня объектно-ориентированные языки программирования для получения перенастраиваемых, допускающих изменения и при этом устойчивых систем. Но чтобы получить эти преимущества, гораздо важнее использовать объектную технологию проектирования. Rational Unified Process производит объектно-ориентированную модель проекта, на которой базируется вся дальнейшая работа.

Объектно-ориентированная модель стремится к отражению мира, который мы наблюдаем в действительности. Поэтому объекты часто соответствуют вполне реальным вещам, которые должна обработать система. Например, объектами могут быть счет в банковской системе или



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.

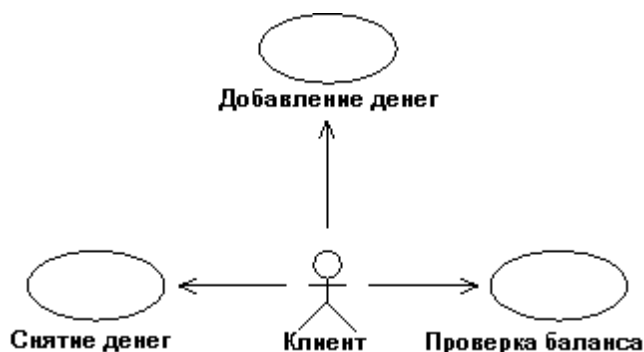
Субъекты

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**.

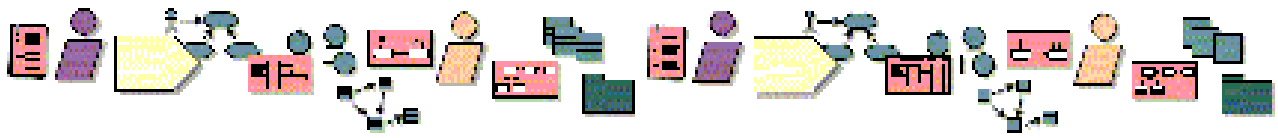
Прецеденты

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**.

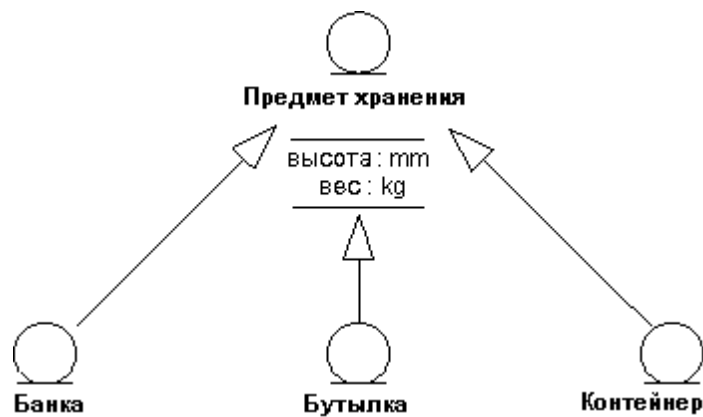
Объект определяется в **классе**.

Классы

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

Обобщения

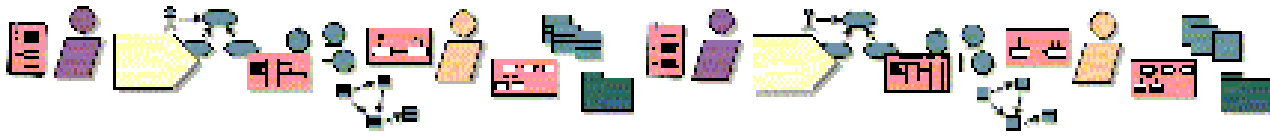
Обобщения показывают, как один класс наследуется от другого. Наследованный класс называется **потомком**. Класс, от которого происходит наследование, называется **предком**. Наследование означает, что определение предка – включая любые свойства типа атрибутов, связей или операций его объектов – является 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

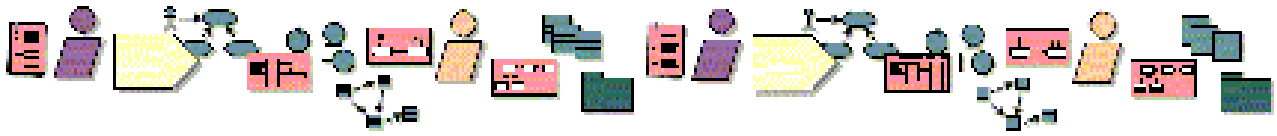
Раньше уже говорилось о том, что модели в Rational Unified Process используют UML как общую систему обозначений. Но при разработке больших систем этого мало. Нужно иметь инструмент, который поддерживал бы не только рисование составляющих модели диаграмм, но и контролировал бы их непротиворечивость и 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. 4-2](#) при обсуждении основных концепций моделирования. Приведем определения:

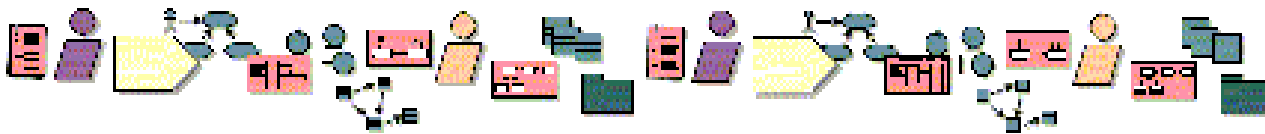
Экземпляр прецедента – это последовательность действий, выполняемых системой; эта последовательность имеет наблюдаемый результат, представляющий ценность для конкретного субъекта.

Прецедент – это набор экземпляров прецедента.

В этом определении есть несколько ключевых слов:

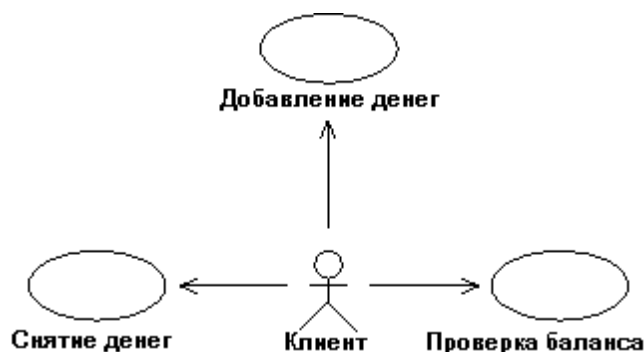
- **Экземпляр прецедента.** Последовательность, упомянутая в определении, действительно является конкретным потоком событий в системе, или экземпляром. Возможно множество потоков событий, и многие из них могут быть очень похожи. Чтобы сделать модель прецедентов понятной, можно сгруппировать подобные 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.

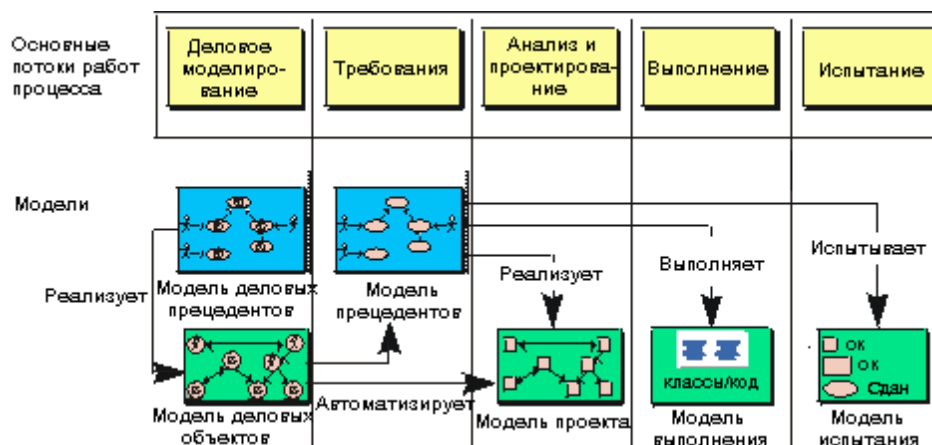


Вам уже знаком этот рисунок. Здесь указаны все прецеденты, которые выполняет банкомат по требованию клиента.

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.

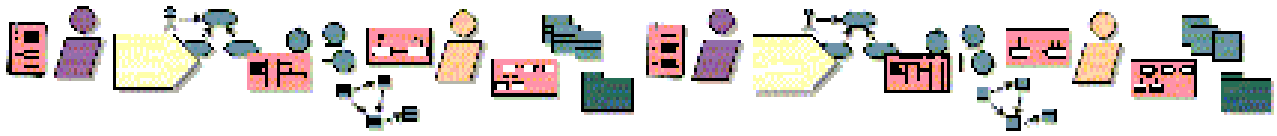


«Поток» прецедентов через различные модели

■ 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,



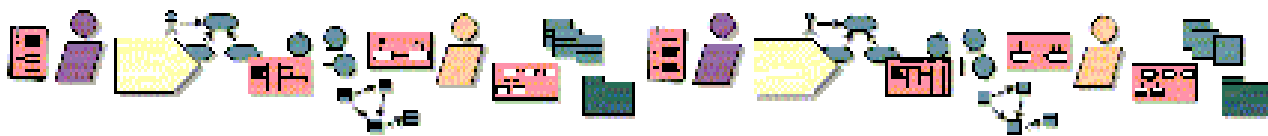
что существенно исполнить в деловой сфере, чтобы обеспечить субъекту нужные результаты. Оно определяет также, **какое** взаимодействие должны иметь деловая сфера и субъекты при выполнении делового прецедента. Такое представление должно быть разработано, когда Вы договариваетесь о том, что должно быть 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.



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.

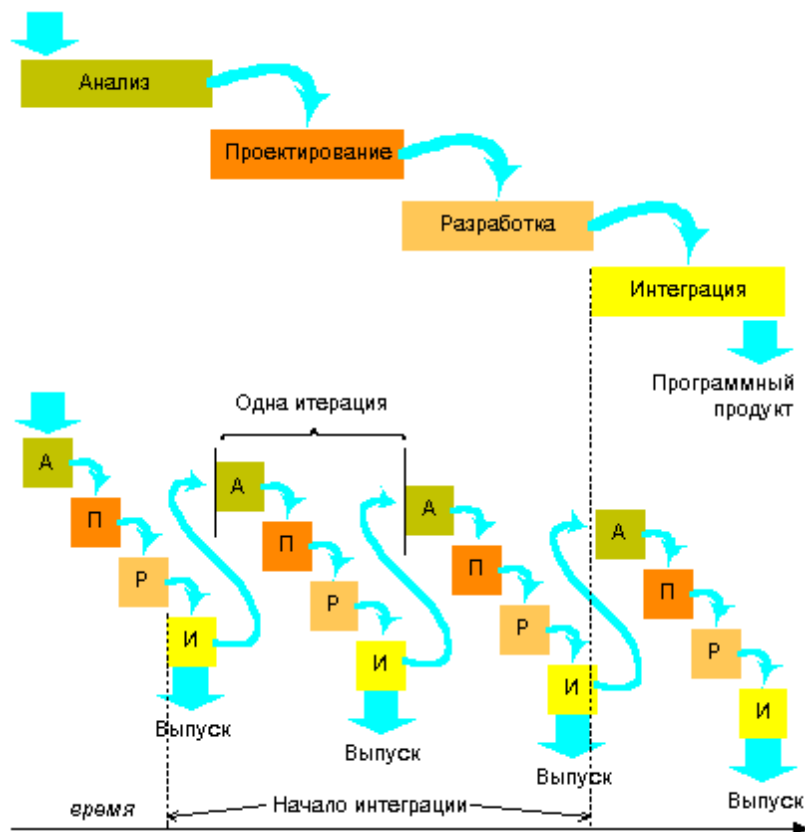
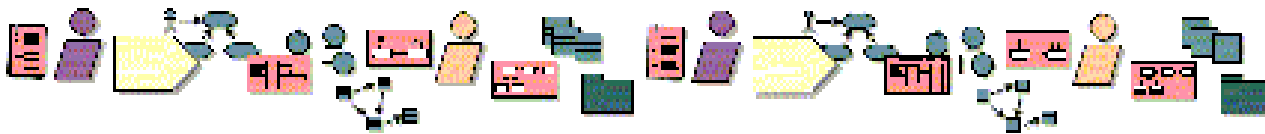
Несколько лет назад многие из нас и представить себе не могли, что процесс «водопада» не является единственным разумным подходом. А если бы и смогли, это не изменило бы ситуацию: метод «водопада» заложен во все ГОСТы и методики проектирования программных систем.

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

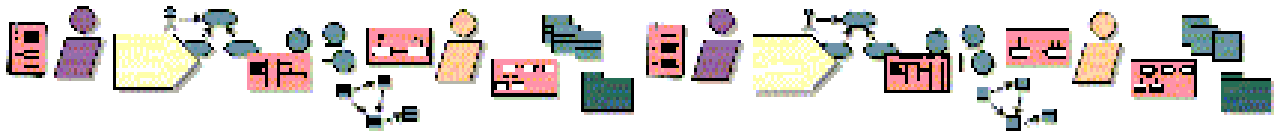
Определение итерации, как ее представляет Rational Unified Process, приводилось раньше, на [стр. 3-8](#). Вспомним его еще раз: «Итерация – это законченный цикл разработки, приводящий к выпуску **выполнимого изделия** (внутренней или внешней версии) или **подмножества конечного продукта**, которое возрастает с приращением от итерации к итерации, чтобы стать законченной системой.



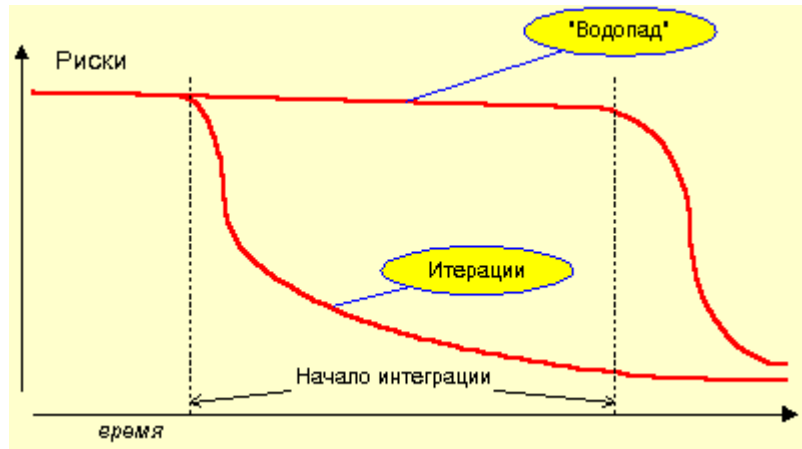
Сравните «водопад» (вверху) с итерационным подходом. Во втором случае каждая итерация начинается анализом и заканчивается выпуском изделия

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



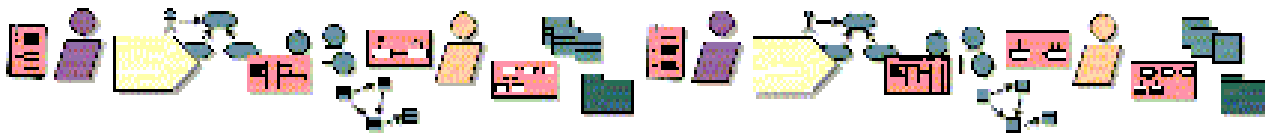
обеспечения, навыки людей и т.д. Некоторые предполагаемые риски могут не проявиться, и могут появляться новые, не предполагавшиеся ранее.



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.

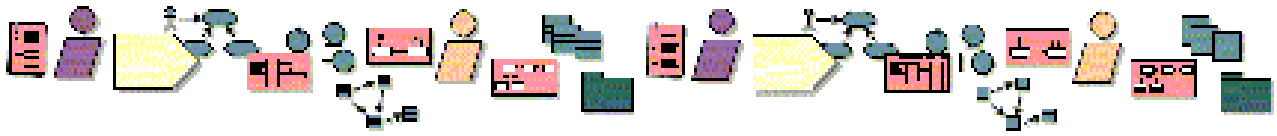
Требование – это условие или возможность, которой должна соответствовать система.

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:

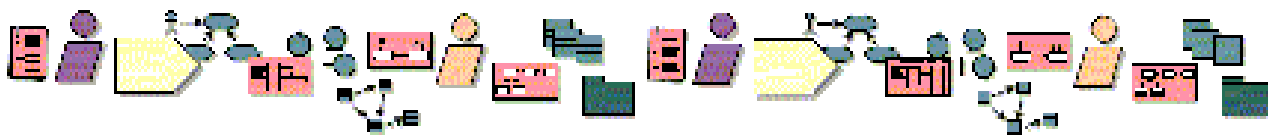
Эффективное решение любой сложной проблемы предполагает удовлетворение потребностей некоторой группы совладельцев. Совладельцы обычно имеют различные точки зрения на проблему и различные потребности, которые должны быть удовлетворены принимаемыми решениями. Многие совладельцы являются пользователями 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



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.

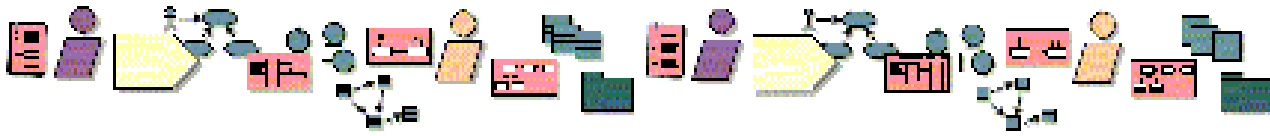
Часто совершаемая ошибка состоит в том, что Ваше ощущение комплексности разработки толкает Вас на формирование комплексного определения. Это ведет к трудностям в объяснении цели проекта и системы. Исполнители могут быть искренне увлечены работой, но они не дадут хорошей отдачи, так как у них нет полного 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.

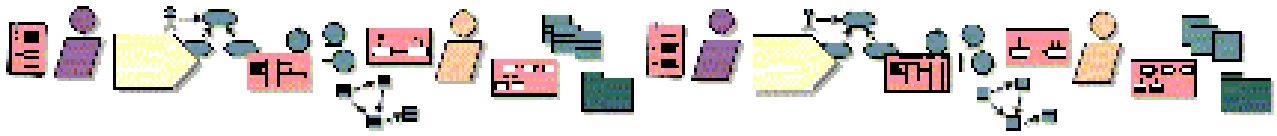
Что такое архитектура приложения?

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.

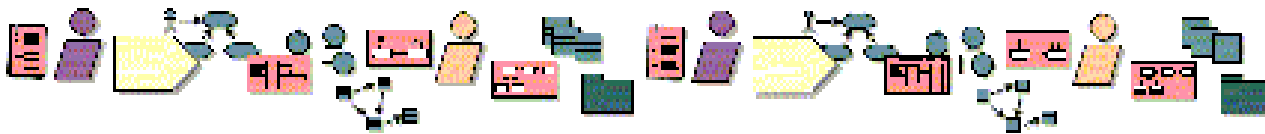
- Архитектура является базой для руководства проектом.

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.



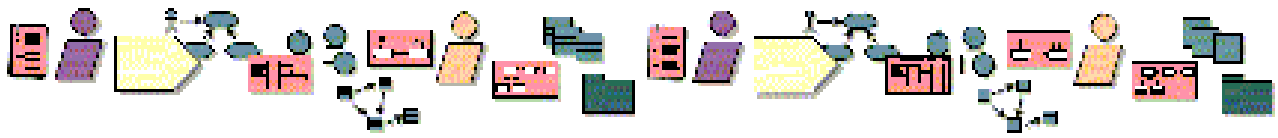
Представления архитектуры по существу являются выжимками, иллюстрирующими «архитектурно существенные» элементы моделей. В Rational Unified Process Вы начинаете с типичного набора представлений, называемого моделью представления «4+1».



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».

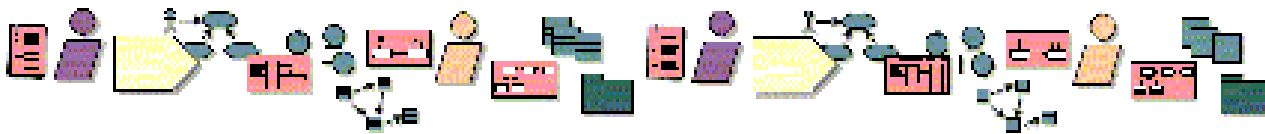
Круг интересов архитектуры

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.

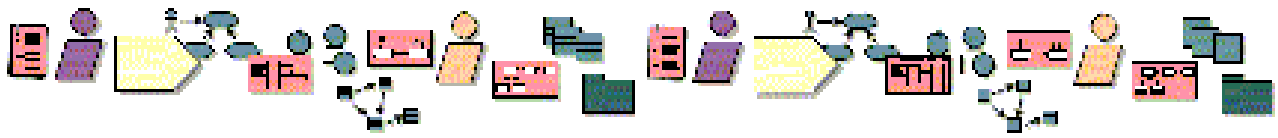
Архитектура в Rational Unified Process – это, прежде всего, результат потока работ анализа и проектирования. Так как проект повторяет этот поток работ, итерацию за итерацией, архитектура развивается, усовершенствуется и приглаживается. Поскольку каждая итерация включает интеграцию и тестирование, ко 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.

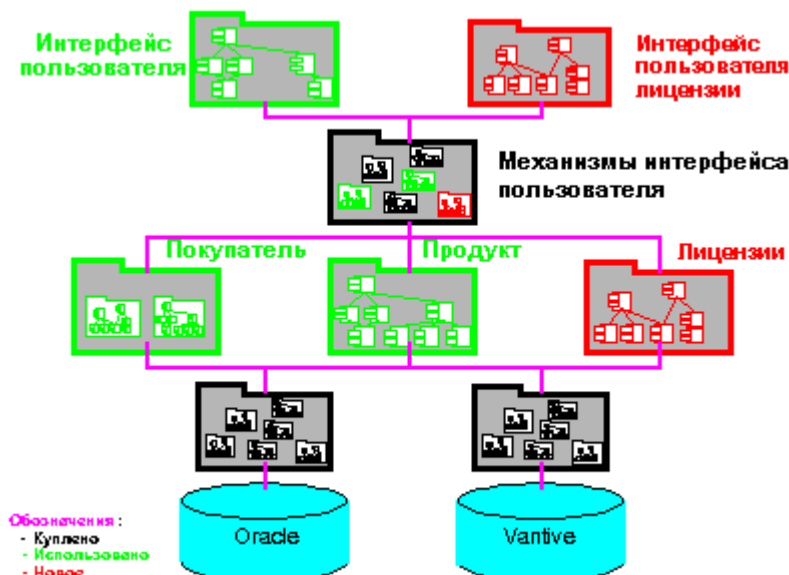
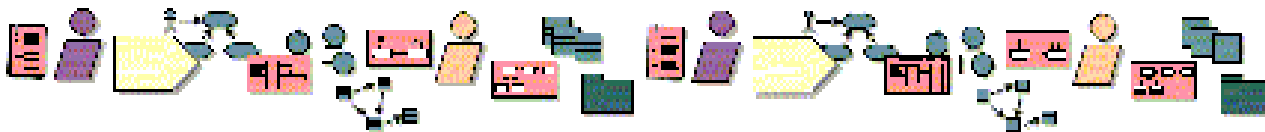
Разработка на базе компонентов

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:

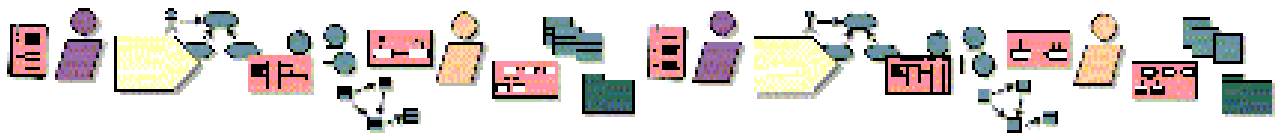
- Итерационный подход позволяет Вам постепенно идентифицировать компоненты, решая какой из них разрабатывать, какой многократно использовать и который купить.
- Акцент на архитектуре программы позволяет Вам соединять структуру (компоненты) и способы их интеграции (фундаментальные механизмы и характер 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.

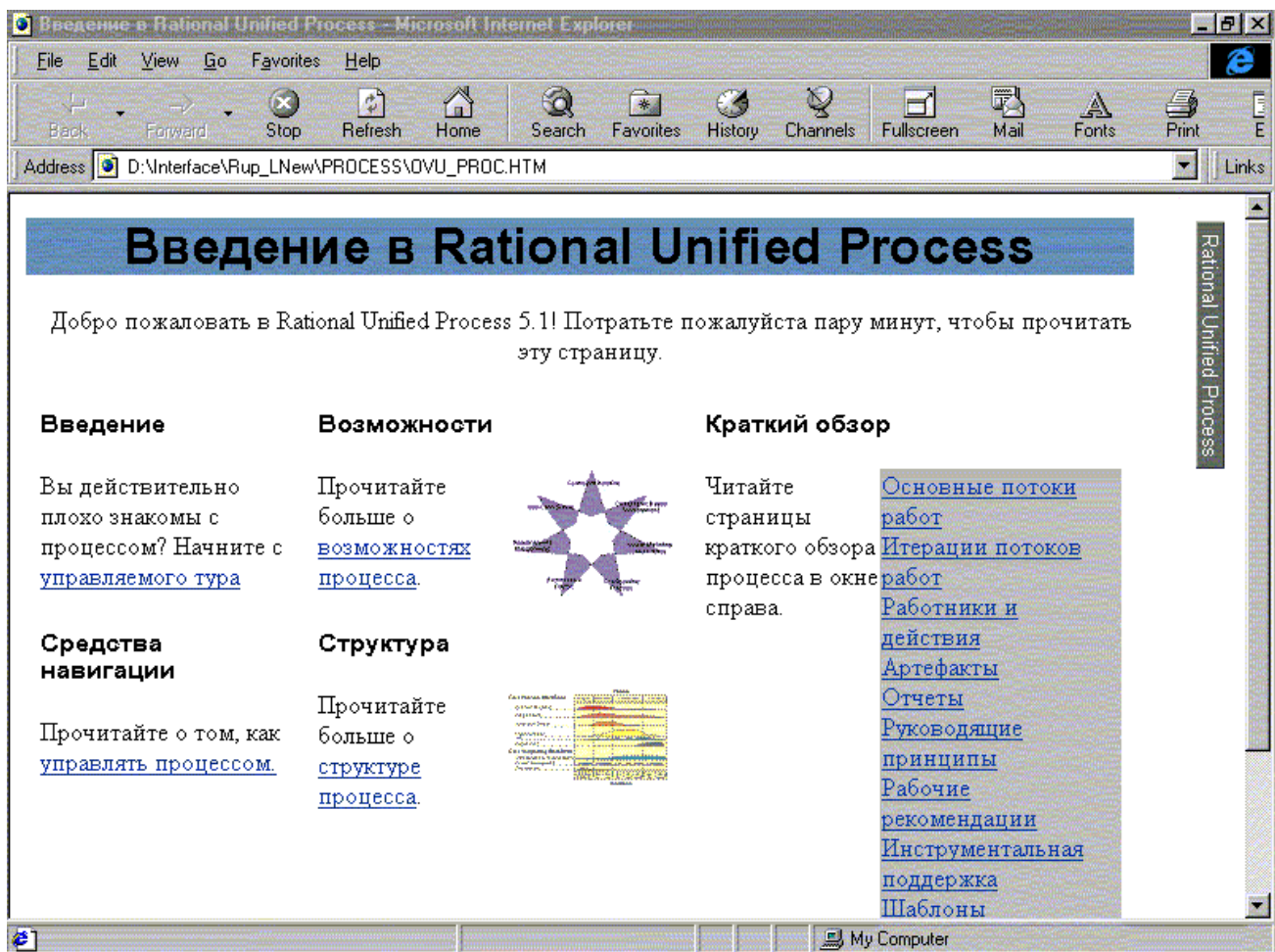
С возможностями настройки процесса к нуждам конкретного проекта и конкретной организации мы познакомимся более подробно при рассмотрении потока работ «Среда».



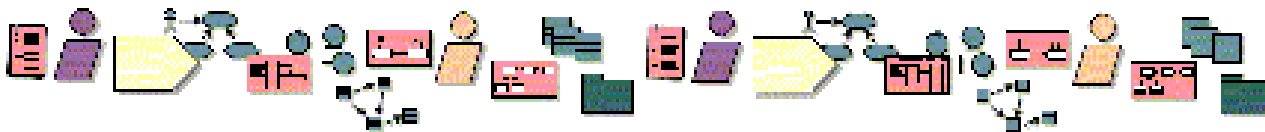
Начало работы с Rational Unified Process

Средства навигации Rational Unified Process

Раньше уже упоминалось, что интерактивная версия Rational Unified Process предоставляет возможность просматривать содержание базы знаний любым из популярных сетевых браузеров, типа Netscape® Navigator™ или Microsoft® Internet Explorer™. Интерактивность Rational Unified Process поддерживается множеством гипертекстовых ссылок и интерактивных изображений.



Вы можете интерактивно просматривать базу знаний Rational Unified Process.



Для тех, кто пользуется Интернетом, путешествие по базе знаний Rational Unified Process не составит никакого труда. Тем не менее, укажем на некоторые специфические для Rational Unified Process средства навигации.

Кнопка на поле страницы

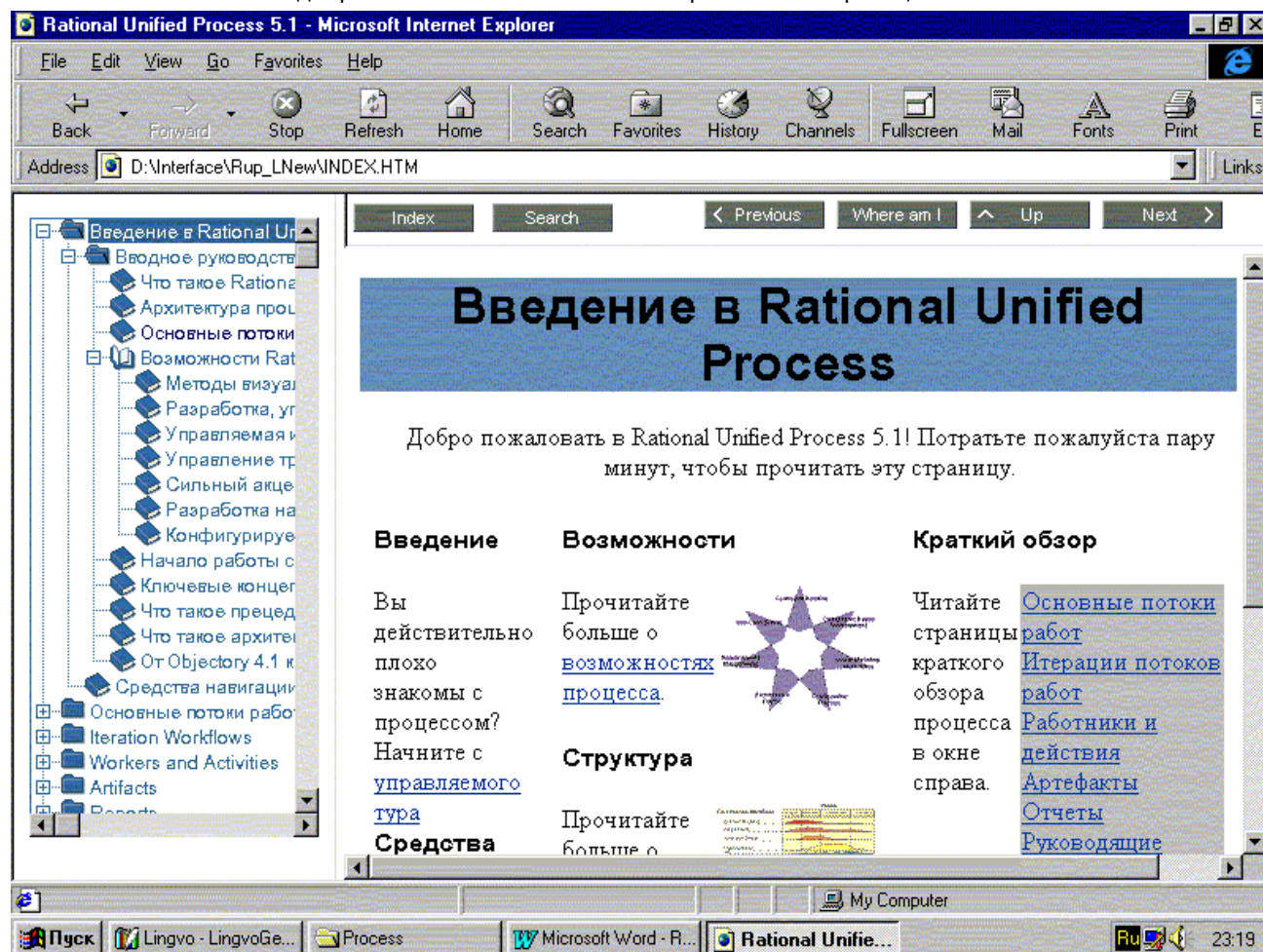
Большинство страниц имеет кнопку на правом поле (как показано справа от этого текста:

Назначение этой кнопки состоит в том, чтобы дать возможность перейти к любой нужной странице Rational Unified Process. Щелчок на кнопку инициирует появление окна просмотра древовидной структуры слева и кнопок навигации в верхней части окна.

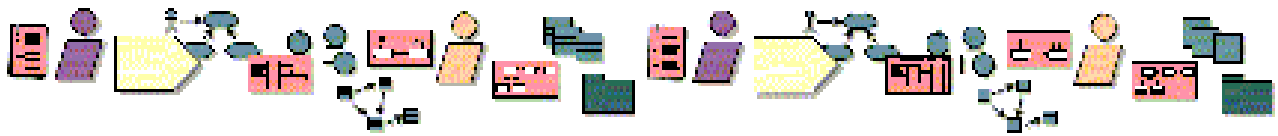
Обратите внимание: Если Вы уже воспользовались этой кнопкой, ее повторное нажатие не приводит ни к каким изменениям.

Rational Unified Process

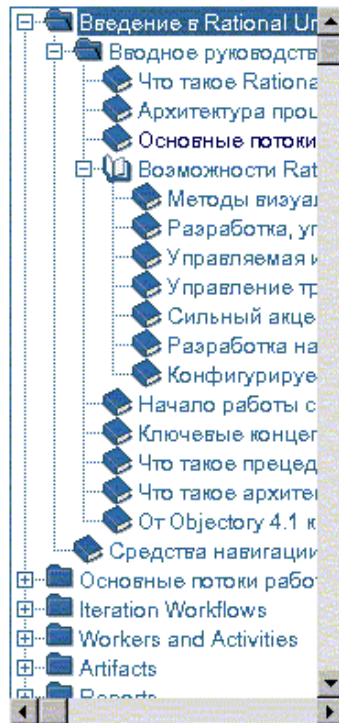
Вид экрана после нажатия кнопки на правом поле страницы



Окно просмотра древовидной структуры



Окно слева обеспечивает связь с различными частями описания процесса. Нажимая на нужную



ссылку, Вы отображаете соответствующий раздел в основном окне.

Кнопки Previous, Up и Next отображаются в основном окне и являются контекстно-зависимыми. Эти кнопки обеспечивают



перемещение по древовидной структуре, представленной в левом окне. Не следует путать кнопки Previous и Next с аналогичными кнопками в браузере Интернета, где последние отображают просмотренные страницы в хронологическом порядке.

Кнопка Where am I высвечивает текущую страницу окна просмотра древовидной структуры и разворачивает дерево в случае необходимости.

Кнопки поиска



Кнопки Index и Search отображают каждая новое окно, позволяя Вам просмотреть алфавитный указатель или ввести строку для свободного текстового поиска, соответственно.

Таблица краткого обзора

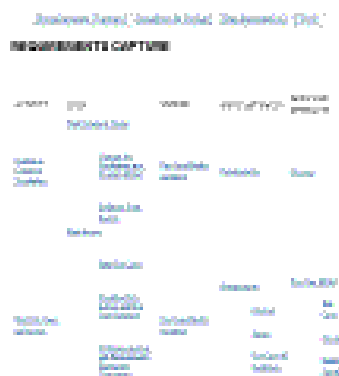
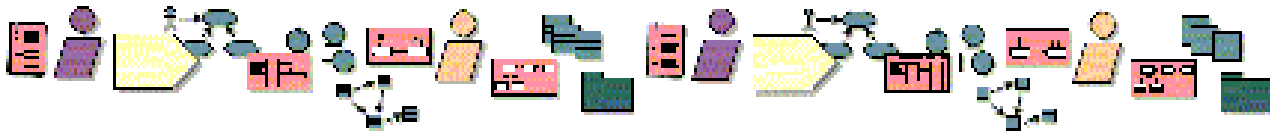


Таблица краткого обзора процесса обеспечивает ссылки ко всем объектам процесса. Она особенно полезна, если Вы точно знаете, что ищете.

Таблица краткого обзора вызывается в окне просмотра древовидной структуры.



Изучение процесса

В этом разделе даются некоторые советы о том, как изучать процесс, читая интерактивную версию Rational Unified Process. Помните о том, что самый эффективный путь изучения чего-либо – это фактическое выполнение предлагаемых рекомендаций. Старайтесь совмещать ваше чтение с осмыслением прочитанного. Посмотрите на выполняемую Вами работу по-новому, с учетом полученных знаний.

Тем, кто пока не имеют доступа к полной интерактивной версии Rational Unified Process (на английском языке), я рекомендую воспользоваться демонстрационной русской версией, которую Вы, наверное, уже списали.

Не старайтесь объять необъятное. На первый раз постарайтесь разобраться в структуре интерактивной версии Rational Unified Process. Демонстрационной версии для этого вполне достаточно. Мы намерены расширять возможности демонстрационной версии по мере продолжения нашей работы по изучению Rational Unified Process.

Примечание: В демонстрационной версии сохранены все ссылки оригинала. Однако не для всех ссылок обеспечены страницы-адресаты. Если Вы записали демонстрационную версию поверх полной версии Rational Unified Process, то при выборе ссылки, не имеющей в демонстрационной версии страницы адресата, Вы увидите английский текст. В противном случае Вы получите сообщение об ошибке. Продолжайте работать дальше.

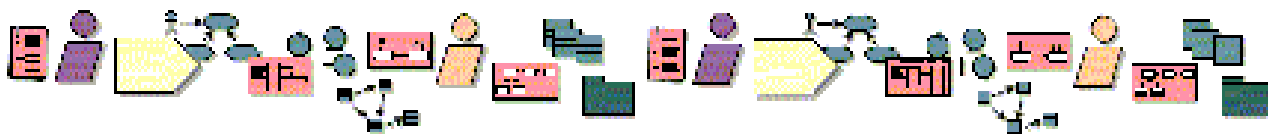
Для начала выберите Управляемый тур. Этот тур содержит Введение в Rational Unified Process и три путеводителя в соответствии со следующими ролями в проекте:

- **Руководитель проекта.**
- **Инженер-технолог.** Человек, который конфигурирует или приспособливает процесс к Вашим конкретным условиям организации и проекта. Тот, кто играет в проекте роль инженера-технолога, обычно отвечает за методы.
- **Разработчик.** Эта роль включает различных работников (проектировщиков прецедентов, архитектора и так далее).

Выберите Введение в Rational Unified Process. На этой странице доступны ссылки к разделам Введение, Архитектура процесса и Основные потоки работ процесса. Если Вы читали эту книжку с начала, то материал покажется Вам, в основном, знакомым.

Вернитесь к началу. Нажмите кнопку на правом поле страницы и воспользуйтесь для изучения структуры базы знаний Rational Unified Process окном просмотра древовидной структуры.

На первый раз этого достаточно. Итак, запускайте интерактивную версию Rational Unified Process...



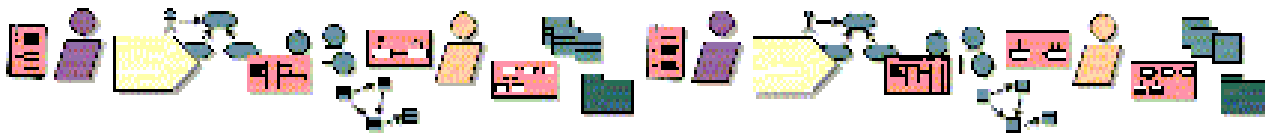
Внедрение Rational Unified Process

Для успешного внедрения нового процесса работы над проектом по разработке программного обеспечения (или над всеми проектами в организации), Вы должны как минимум рассмотреть следующие проблемы:

- **Поддержка со стороны администрации.** Без соответствующей поддержки администрации трудно вводить новый процесс.
- **Обучение участников проекта.** Каждый участник проекта должен понимать новый процесс.
- **Конфигурирование процесса для удовлетворения потребностей проекта.** При конфигурировании необходимо рассмотреть такие факторы как прикладная область, разновидность приложения, культура компании и инструментальные средства поддержки.
- **Инструментальные средства поддержки.** Процесс разработки программного обеспечения будет гораздо эффективнее, если все действия жизненного цикла системы будут поддержаны соответствующими инструментальными средствами. Вы должны выбрать набор инструментальных средств поддержки и приспособить их для работы друг с другом в пределах процесса.

Важно, чтобы введение нового процесса и, возможно, приобретение нового набора инструментальных средств поддерживала администрация. Руководство должно знать все, что требуется, чтобы ввести новый процесс:

- **Введение нового процесса всегда требует определенных капитальных затрат.** Вы должны обучить ваш персонал и, возможно, оплачивать консультации. Вы можете получить более низкую производительность в вашем первом проекте, по крайней мере, если Вы включите оплату необходимого обучения в стоимость проекта. Может быть, правильнее исключить расходы на обучение из первого проекта и получить более справедливую оценку сравнения с другими проектами. Через какое-то время Вы увидите увеличение производительности; проверьте свои возможности по повышению качества уже с первого проекта.
- **Большинство людей стремится делать дело способами, с которыми они уже знакомы.** Вы можете столкнуться с тем, что в критической ситуации персонал возвратится к старым процедурам. Как администратор, Вы должны быть готовы к этому. Стимулируйте членов группы придерживаться процесса. Вы и сами можете вернуться назад к старым процедурам, когда у Вас возникнут трудности. Попробуйте пройти по крайней мере один полный проект, если это возможно, и затем оцените то, что получилось хорошо, что не получилось и что может быть улучшено.
- **Когда вводится новый процесс, всегда есть определенный риск, но имеется также и большая награда.** Один из способов уменьшить риск состоит в том, чтобы обеспечить соответствующее обучение и выполнить пробный проект (см. ниже). Другой способ снижения риска состоит в привлечении квалифицированных консультантов. Они должны быть способны идентифицировать и устранить некоторые из проблем прежде, чем они станут



дорогостоящими проблемами. Кроме того, они обычно помогают увеличить производительность, предупреждая о возможных ловушках.

Очень рекомендуется вводить процесс начиная с маленького проекта. Цель проекта должна состоять в том, чтобы люди поняли процесс и научились с ним работать. Другая цель - разработать конфигурацию процесса (определить, как процесс должен использоваться). Рекомендуется также обучать людей в среде, которая является настолько реальной, насколько это возможно.

Есть несколько путей достижения этих целей. Все они представляют собой объединение обучения, обсуждения с консультантами и работы над проектом:

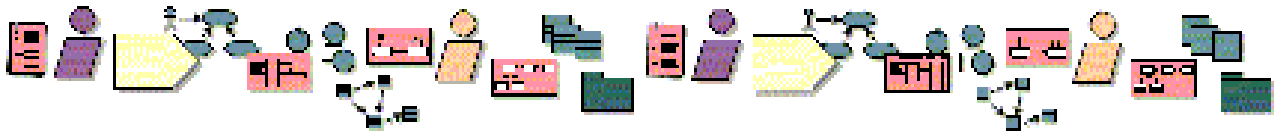
- Примените процесс к пробному проекту, в котором Вы работаете над фиктивным примером-прототипом. Цель состоит в том, чтобы изучить процесс без воздействия на фактическое изделие.
- Примените процесс только к первой итерации проекта. Вы работаете над частями программы, и у Вас вполне достаточно времени, чтобы установить, что что-то идет не так, как надо.
- Устройте «учебный лагерь новобранцев». Цель состоит в том, чтобы позволить разработчикам пройти все (или большинство) стадии процесса разработки за возможно короткое время. Система могла бы быть фиктивным прототипом или частью реальной системы. Преимущество «учебного лагеря новобранцев» состоит в том, что люди нацеливаются на изучение. Цель состоит прежде всего в том, чтобы пройти все стадии и научиться в максимально возможной степени, получая таким образом достаточно опыта в конфигурировании процесса для выполнения определенного набора потребностей.

При планировании обучения, старайтесь проводить его тогда, когда это необходимо. Это максимизирует мотивы и способность к обучению. Например, запланируйте обучение проектированию, когда люди собираются начать проектировать.

Инструментальные средства поддержки

Процесс разработки программного обеспечения требует, чтобы инструментальные средства поддерживали все действия жизненного цикла системы. Итерационный процесс развития предъявляет специальные требования к используемым Вами инструментальным средствам, типа их интеграции друг с другом и прямой и обратной разработки моделей и кодов. Вы нуждаетесь также в инструментальных средствах для автоматизации документирования и, возможно, автоматизации испытаний. Rational Unified Process можно использовать совместно с рядом инструментальных средств фирмы Rational или других продавцов. Однако Rational предоставляет много хорошо интегрированных инструментальных средств, которые эффективно поддерживают Rational Unified Process.

Ниже перечислены инструментальные средства, которые Вам потребуются, и некоторые примеры инструментальных средств Rational, отвечающих этим потребностям.



- **Инструмент моделирования для разработки различных моделей, типа модели прецедентов или модели проекта. Инструмент должен иметь возможность прямой и обратной разработки, чтобы Вы могли выполнять прямое проектирование и перепроектировать код без того, чтобы отменять изменения, которые Вы сделали в моделях или в коде начиная с предыдущей генерации.**

Пример: Rational Rose

- **Инструмент управления требованиями для фиксации, организации, расположения по приоритетам и прослеживания всех требований.**

Пример: Rational Requisite Pro

- **Инструмент документирования для поддержки проектной документации. Вы должны иметь возможность извлекать информацию из инструмента моделирования и из других источников, например, кода и создавать документы, которые представляют модели. Если Вы не имеете средств автоматизированного порождения документов, то скорее всего Вы будете иметь документацию, которая отличается от ваших моделей (или никакой документации вообще). Инструмент документирования должен позволять Вам вносить изменения в документ вручную и не отменять эти изменения при автоматическом восстановлении документа.**

Пример: Rational SoDA

- **Средства программирования, чтобы помочь разработчикам; редакторы, трансляторы, отладчики и так далее. Они должны быть интегрированы со средой моделирования и условиями проведения испытаний.**

Примеры: Rational Apex/Ada, Rational Apex/C++ (Java ready)

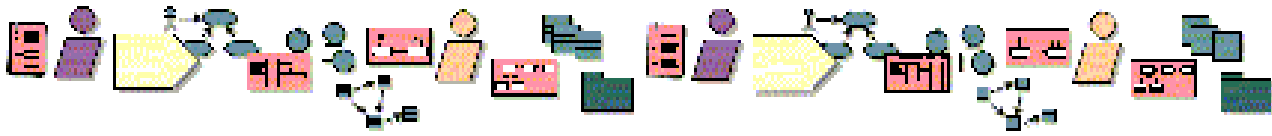
- **Инструментальные средства, которые поддерживают руководителя проекта при планировании и управлении проектом.**

Пример: Microsoft Project

- **В большом проекте Вам нужно автоматизировать распределение задач между участниками проекта. Инструмент управления задачами может помочь организаторам проекта знать, какие задачи находятся в работе, привести их исполнение в соответствие некоторому процессу, и в то же самое время автоматизировать утомительные задачи, связанные с управлением конфигурацией. Он может также помочь руководству проекта непрерывно контролировать его продвижение.**

Примеры: Rational ClearQuest

- **Инструмент управления конфигурацией может помочь Вам следить за всеми произведенными артефактами и их различными версиями. Обе модели, и особенно кодирование, должны управляться конфигурацией. Очень важна интеграция среды кодирования, инструментальных средств моделирования и инструментальных средств**

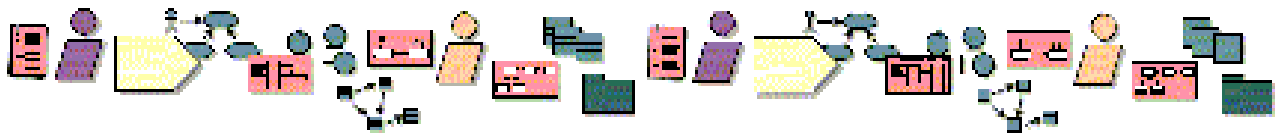


управления конфигурацией.

Пример: ClearCase.

- В процессе итерационной разработки Вы выполняете испытания на протяжении всего жизненного цикла. Важно использовать средства тестирования для автоматизации испытаний так, чтобы Вы могли легко повторять испытания кода (возвратное тестирование) при минимизации затрат и увеличении качества. Более специализированные инструментальные средства позволяют выполнять испытания загрузки.

Примеры: Rational SQA Suite, Rational TestMate, Rational Visual Test



Поток работ «Управление проектом»

Мы переходим ко второй (условно) части нашей книжки – к описанию потоков работ разработки программного обеспечения, как они представлены в Rational Unified Process.

Основные потоки работ, составляющие статическое содержание процесса, уже перечислялись в главе «Архитектура процесса». Не будет лишним перечислить их еще раз. Это шесть потоков работ непосредственно разработки:

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

и три потока работ поддержки:

- Управление конфигурацией и изменением
- Управление проектом
- Среда

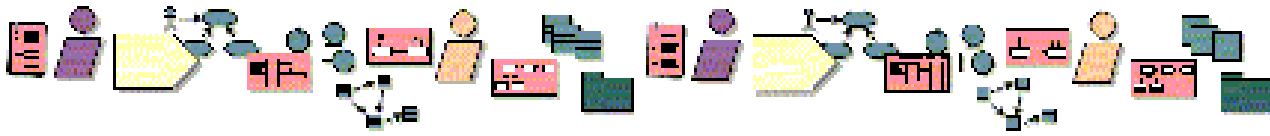
Знакомство с потоками работ мы начнем с потока из второй части списка, с управления проектом.

Во-первых, именно в этом потоке работ кто-то (видимо, руководитель проекта) произносит: «Все! Хватит болтать! Начинаем...», - и далее определяет что, кому, когда и как делать. А значит и понадобится описание этого потока работ прежде других.

Во-вторых, именно эта часть технологии разработки слабо (очень мягко сказано!) отражена в отечественной программистской литературе. Не все, кто начал, дочитают эту книжку до конца. Поэтому наиболее важную информацию хотелось бы поместить как можно раньше.

Руководство проектом программного обеспечения - искусство балансирования конкурирующих целей, управления рисками и преодоления ограничений ради поставки изделия, отвечающего потребностям обеих категорий заказчиков - оплачивающих счета и пользователей. Тот факт, что бесспорно успешных проектов так немного, объясняется чрезвычайной сложностью задачи.

Цель потока работ управления проектом программного обеспечения в Rational Unified Process



состоит в том, чтобы упростить задачу, предоставляя некоторый контекст руководства проектом. Он не является панацеей для успеха, но он обеспечивает тот подход к управлению проектом, который заметно увеличит вероятность поставки удачного программного обеспечения.

С этой целью поток работ управления проектом Rational Unified Process:

- Предлагает структуру управления сложными (преимущественно программными) проектами.
- Предлагает практические рекомендации по планированию, укомплектованию персоналом, выполнению и мониторингу проектов.
- Предлагает структуру управления рисками.

Однако этот поток работ не пытается охватить все аспекты руководства проектом. Например, он **не** включает проблемы типа:

- Руководство людьми: наем, обучение, тренировка
- Управление бюджетом: определение, распределение и т.д.
- Управление контрактами с поставщиками и заказчиками

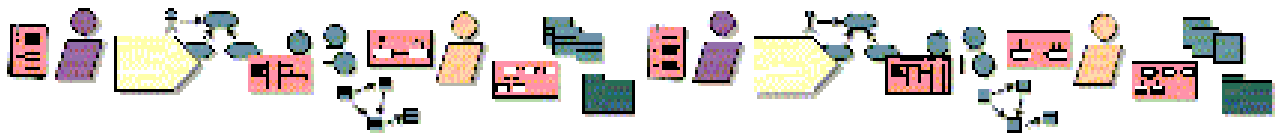
Этот поток работ нацелен, главным образом, на самые важные аспекты итерационного процесса разработки:

- Управление рисками
- Планирование итерационного проекта: всего жизненного цикла и конкретной итерации
- Мониторинг продвижения итерационного проекта, метрики

Описание потока работ «Управление проектом» будет представлено в двух главах (выпусках).

В этой, первой главе, посвященной управлению проектом, будут кратко описаны все действия и артефакты, а также введены концепции рисков и метрик – двух основных понятий планирования и управления итеративным процессом.

Вторая глава (следующий выпуск) будет посвящена способам планирования и мониторинга итеративного процесса разработки проекта программного обеспечения.



Краткий обзор действий

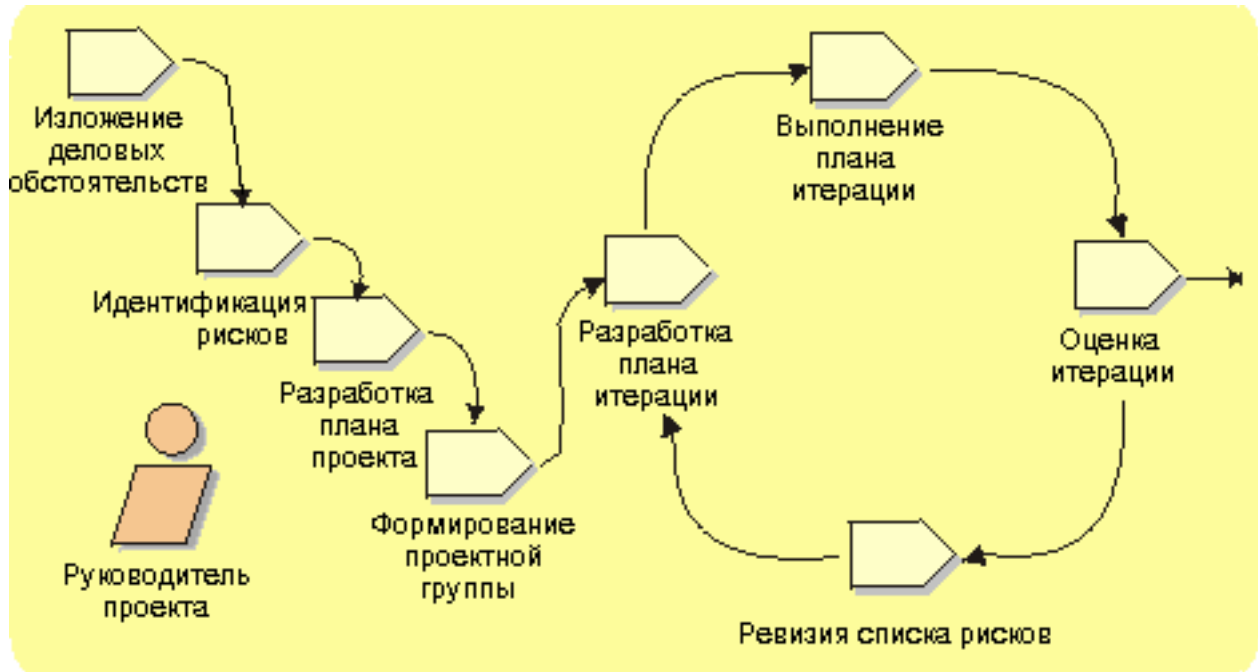


Диаграмма краткого обзора действий потока работ показывает все действия управления проектом

Работник: Руководитель проекта

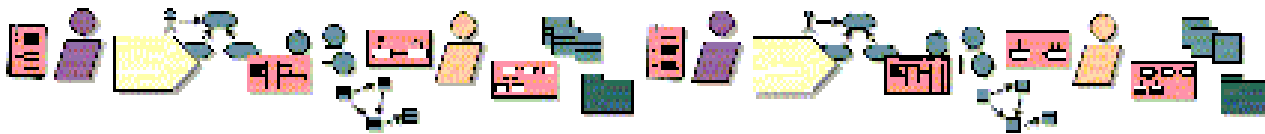
Работник «Руководитель проекта» является ключевым действующим лицом этого потока работ.

Он распределяет ресурсы, формирует приоритеты, координирует взаимодействия с заказчиками и пользователями и вообще руководит проектной группой, направляя ее действия в правильное русло. Кроме того, руководитель проекта устанавливает порядок действий, который гарантирует целостность и качество артефактов проекта. И, наконец, руководитель проекта отвечает за эффективное выполнение процесса проверки изменения изделия.

Действие: Изложение деловых обстоятельств

Описание деловых обстоятельств документирует экономическую эффективность изделия. Это инструмент, посредством которого обосновывается размер капиталовложений в проект. Плохое документирование деловых обстоятельств может потопить лучшие проектные идеи, в то время как грамотное документирование деловых обстоятельств может гарантировать соответствующее финансирование достойных изделий.

Действие «Изложение деловых обстоятельств» включает следующие шаги:



-
- Описание изделия.
 - Определение контекста предметной области.
 - Определение цели создания изделия.
 - Разработка финансового прогноза.
 - Описание проектных ограничений.

Действие выполняется один раз в каждой итерации. Результатом действия является артефакт «Деловые обстоятельства».

Инструментальная поддержка действия в Rational Unified Process – шаблон MSWord.

Действие: Идентификация рисков

Целью этого действия является идентификация рисков и прогнозирование их воздействия на проект.

Концепция рисков является одной из важнейших для управления итеративным проектом. Мы познакомимся с ней позже в этой главе.

Действие «Идентификация рисков» включает следующие шаги:

- Идентификация потенциальных рисков
- Группировка и сортировка рисков
- Идентификация стратегий предотвращения риска
- Идентификация стратегий уменьшения риска
- Идентификация стратегий локализации риска

Действие выполняется один раз в каждой стадии. Результатом действия является артефакт «Список рисков».

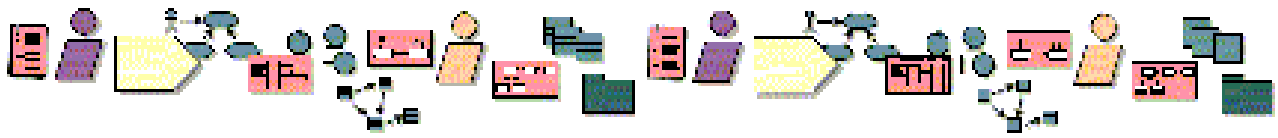
Инструментальная поддержка действия в Rational Unified Process – шаблоны краткого и детального списков MSWord.

Действие: Разработка плана проекта

В этом действии производится крупномодульный план проекта, фокусирующийся на главных вехах и ключевых выпусках в течение всего жизненного цикла.

Действие «Разработка плана проекта» включает следующие шаги:

- Определение вех проекта
- Определение целей вех



-
- Определение количества и длины итераций в пределах стадий
 - Уточнение сроков вех и контекста
 - Определение плана измерений

Действие выполняется один раз в проекте. Результатами действия являются артефакты «План проекта» и «План измерений».

Инструментальная поддержка действия в Rational Unified Process – шаблоны MSWord и MSPROJECT.

Действие: Формирование проектной группы

Цели этого действия состоят в планировании человеческих ресурсов по наборам навыков, необходимых для проекта, и в группировке доступных ресурсов в относительно независимые, но сотрудничающие группы.

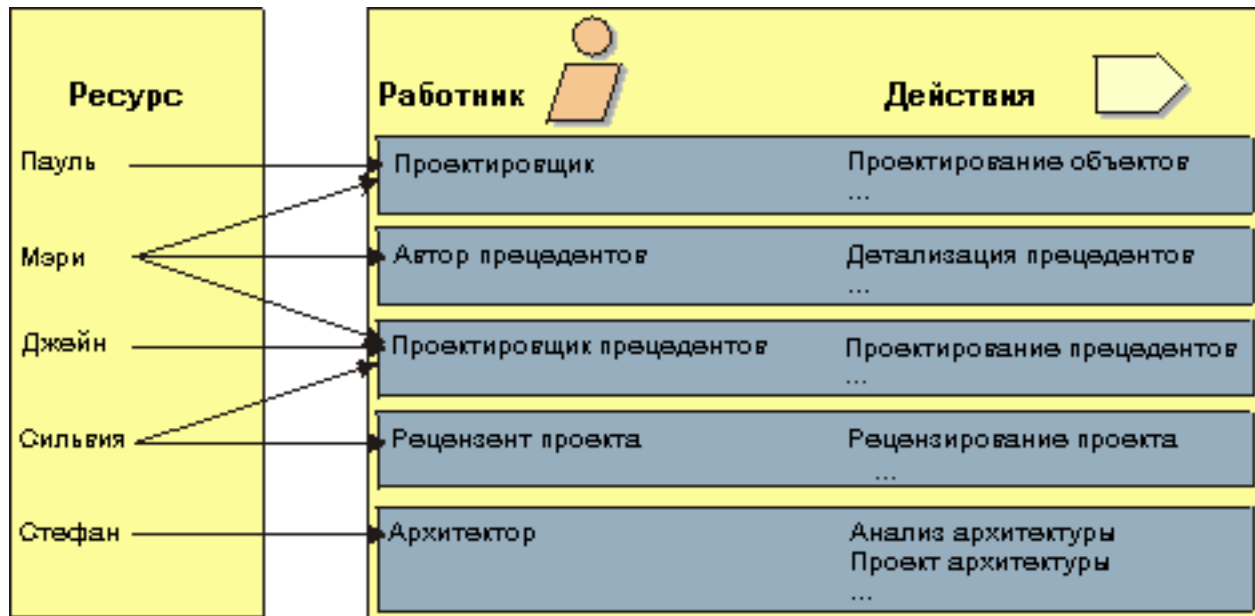
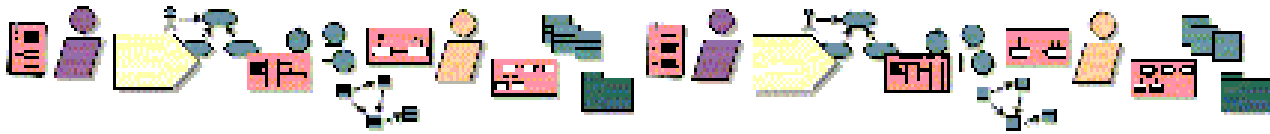
Действие «Формирование проектной группы» включает следующие шаги:

- Рассмотрение потребностей в персонале по стадиям
- Распределение работы среди работников
- Формирование групп
- Обучение персонала

Rational Unified Process описывает потоки работ в терминах работников и действий.

Заметьте, что работники – это не конкретные личности. Эти понятия описывают, как личности должны вести себя в работе, и какие обязанности эти личности имеют.

Обычно проект имеет в своем распоряжении некоторых конкретных людей, имеющих определенные навыки. Например, Пауль, Мэри, Джейн и Сильвия – люди, имеющие различные пересекающиеся навыки. Используя работников, определенных для процесса, отобразите доступные ресурсы проекта, в качестве которых может быть задействован работник.



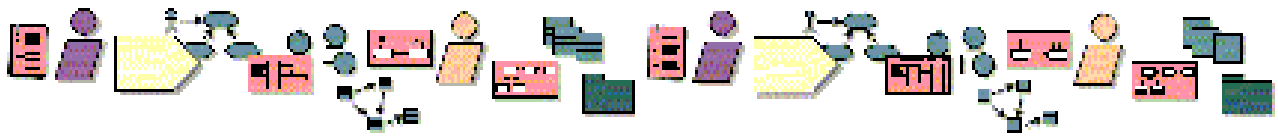
Каждый человек - участник проектной группы - ассоциирован с одним или несколькими работниками

Связь индивидуума с работником через какое-то время может измениться под влиянием стадии жизненного цикла проекта и выполняемой работы.

- Индивидуум может действовать как несколько различных работников в течение одного и того же дня: Сильвия может быть рецензентом проекта утром и проектировщиком прецедента днем.
- Индивидуум может действовать как несколько работников одновременно: Джейн может быть и архитектором и проектировщиком некоторого класса, а также владельцем пакета, который содержит этот класс.
- Несколько личностей могут действовать группой как один работник для совместного выполнения некоторых действий: и Пауль и Мэри могут быть проектировщиками одного и того же прецедента.

Попытайтесь распределить обязанности так, чтобы минимизировать передачу ответственности за артефакты от одного индивидуума к другому: имейте одного и того же человека или группу для проектирования и выполнения подсистемы, чтобы они не должны были изучать работу, уже сделанную другими.

Когда в функции группы входит и проектирование, и выполнение, то переход от конструирования к выполнению будет плавным. Кроме того, это удобно для проектировщиков: изучив, что работает, а что нет, они определяют то лучшее, что следует использовать. Подобно скульптору, хороший проектировщик должен понять выразительное средство, которое для программного обеспечения является средой выполнения.



Профессиональный и количественный состав участников проектных групп зависит от стадии разработки.

Если над проектом работает больше 5-7 человек, то необходимо разделить их на группы. За каждый артефакт (прецедент, подсистему, компонент) должен отвечать один человек или группа. Типичная структура:

- Одна группа сбора требований
- Одна архитектурная группа
- Одна или более групп конструирования (их функции - и проектирование, и выполнение, и тестирование)
- Одна группа интеграции

Группы должны состоять минимально из двух и максимально из семи человек; группы более семи человек обычно естественным путем распадаются на подгруппы, так что лучше сделать это заранее.

Результатом действия является модифицированный артефакт «План проекта», в котором указано распределение человеческих ресурсов по работам во времени.

Действие: Разработка плана итерации

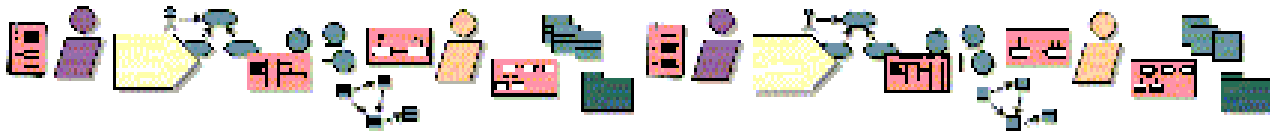
Целью этого действия является разработка подробного плана одиночной итерации, состоящего из детальной схемы разбиения работы на действия, распределения ответственности, внутренних вех итерации и выпусков, оценочных критериев для итерации.

Сама по себе итерация - это набор очень точно выделенных для создания выполнимой программы действий. Для всех итераций, кроме последней итерации перехода, это - полуфабрикат, производимый с целью смягчения рисков и доведения проекта до успешного завершения. Акцент на выполнимости программы вынуждает к почти непрерывной интеграции и позволяет рано обнаруживать технические риски и уменьшать сопутствующие риски проекта.

Выполнение итерации подразумевает некоторое количество доработки и сопровождается изменениями в характере доработки. Короче говоря, чтобы поставить изделие высокого качества, требуется некоторая его доработка. Создание полуфабрикатов и оценка пригодности архитектуры **раньше и чаще** повышают качество конечного продукта, при этом делать изменения дешевле и приспособлять их проще.

Действие «Разработка плана итерации» включает следующие шаги:

- Определение рамок итерации
- Определение критериев оценки итерации
- Определение действий итерации



■ Распределение ответственности

Действие выполняется один раз в каждой итерации. Результатом действия является артефакт «План итерации».

Инструментальная поддержка действия в Rational Unified Process – шаблоны MSWord и MSPProject.

Действие: Выполнение плана итерации

Цель этого действия - произвести выпуск выполнимой программы.

Выполнение плана итерации просто: выполняйте действия, определенные непосредственно в плане итерации. Периодическая оценка продвижения и результатов в контрольных точках с большой вероятностью гарантируют, что проект идет по графику. Малые отклонения от графика можно просто не учитывать, они являются результатом неточной оценки, но накопление сдвигов ведет к большим отклонениям от графика.

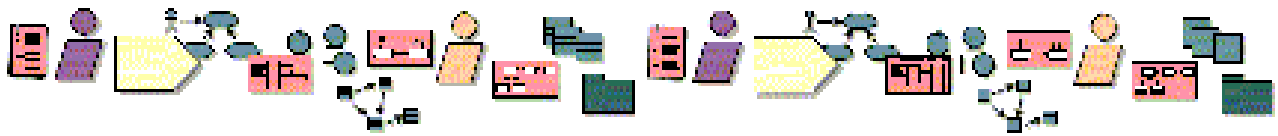
Контролируйте риски, отслеживая наиболее разрушительные.

Результатом действия является артефакт «Модель выполнения».

Действие: Оценка итерации

Оценка итерации фиксирует результат итерации, степень достижения оценочных показателей, полученные уроки и изменения, которые должны быть сделаны.

Одно из основных достоинств итерационного подхода состоит в том, что итерации обеспечивают естественные вехи для оценки прогресса и ограничения риска. В пределах итерации необходимо продолжать оценивать прогресс и риск (возможно неформально), чтобы гарантировать, что трудности не разрушат проект.



Оценка итерации заключается в определении успеха или неудачи итерации и в изучении собранных данных с целью изменения проекта или улучшения процесса его разработки.

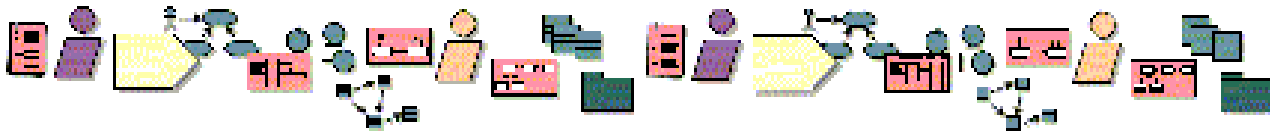
Действие «Оценка итерации» включает следующие шаги:

- Сбор результатов измерений
- Оценка результатов итерации
- Рассмотрение внешних изменений
- Исследование критериев оценки
- Создание запросов изменений

Действие выполняется один раз в каждой итерации. Результатами действия являются артефакт «Оценка итерации» и предложения по корректировке артефактов «Список рисков», «План проекта» и «План итерации» для следующей итерации.

Действие: Ревизия списка рисков

Цель этого действия - модифицировать список рисков для отражения текущего состояния проекта.



Оценка рисков - скорее один непрерывный процесс, чем некоторое разовое действие, которое происходит только в определенных ключевых точках в ходе проекта. Как минимум Вы должны:

- Ежеженедельно ревизовать ваш список, чтобы отслеживать изменения.
- Сделать верхнюю часть списка из десяти пунктов видимыми для всего проекта в целом и настойчиво выполнять все необходимые для них действия.

В конце итерации пересмотрите список рисков с точки зрения целей итерации. Особенно важно:

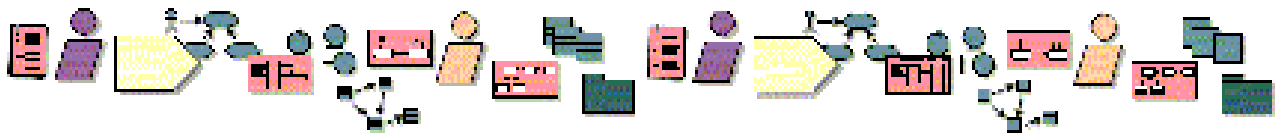
- Исключить риски, которые были полностью смягчены.
- Ввести новые обнаруженные риски.
- Пересмотреть важность и переупорядочить список рисков.

Не беспокойтесь, если Вы обнаружите, что список рисков растет в течение стадий начала и уточнения: начав интеграцию, Вы можете обнаружить некоторые скрытые трудности. Однако, риски должны устойчиво уменьшаться по мере продолжения стадии уточнения и в течение конструирования. Если этого не происходит, значит, Вы не можете обрабатывать риски соответствующим образом, или ваша система слишком сложна, или Вы не можете сформировать систематический и предсказуемый стиль.

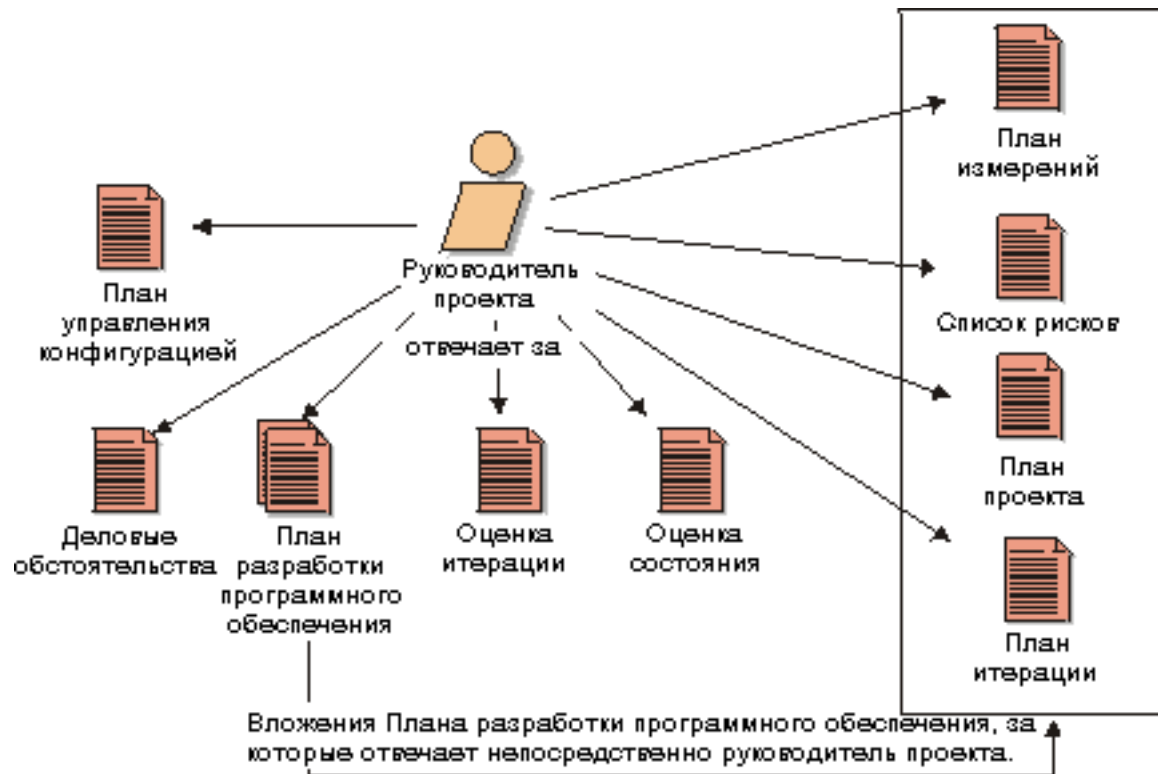
Действие «Ревизия списка рисков» включает следующие шаги:

- Ревизия рисков в течение итерации
- Ревизия рисков в конце итерации

Действие выполняется один раз в каждой итерации. Результатом действия является модифицированный артефакт «Список рисков».



Артефакты управления проектом



Работник «Руководитель проекта» и производимые им артефакты

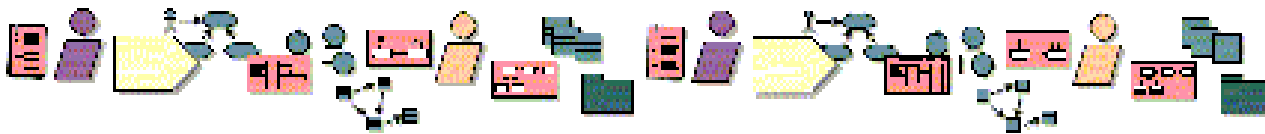
Артефакты управления проектом перечислены в предыдущем разделе при описании действий, в которых они производятся.

Интерактивная версия Rational Unified Process содержит подробные рекомендации по составу соответствующих документов, указания, на что нужно обратить особое внимание и т.д. Инструментальная поддержка производства артефактов в Rational Unified Process – это шаблоны документов MSWord и MSProject.

Концепция рисков в Rational Unified Process

Введение

Процесс программирования имеет дело, главным образом, с **известными** аспектами программирования. Вы можете достаточно точно описывать, планировать, назначать и обозревать то, что, как Вы считаете, должно быть сделано. Управление рисками занимается **неизвестными** аспектами. Многие организации в своей деятельности все еще исходят из «отрицания риска»: оценка и планирование выполняются так, как будто все переменные известны, а это предполагает, что работа механическая, персонал взаимозаменяем и т.д. Но все больше организаций, по крайней



мере, оплачивают «иммитацию» управления риском. Посмотрев на все это глубже, Вы часто обнаруживаете, что это всего лишь поверхностная, слабая попытка минимизации риска.

Определения

Многие решения в итерационном жизненном цикле управляются **рисками**. Чтобы делать это, Вы должны иметь хорошее понимание о рисках в проекте и иметь ясные стратегии того, как смягчать риски и как вообще иметь с ними дело.

В повседневной жизни риск – это подверженность потере или ущербу; фактор, вещь, элемент или направление движения, заключающий вероятную опасность. В разработке программного обеспечения определение более конкретно:

- **Риск** – это переменная, которая в пределах своего нормального распределения может принимать значение, подвергающее опасности или исключающее успех проекта. Проще говоря, риск состоит в том, что что-то может стоять на нашем пути к успеху, это что-то в настоящее время неизвестно или трудно поддается определению.
- **Успех** – это достижение всех требований и ограничений, ожидаемых от проекта.

Мы можем далее квалифицировать риски как прямые или косвенные:

- **Прямой риск**: риск, который в большой степени может управляться проектом
- **Косвенный риск**: риск, который мало или вообще никак не может управляться проектом

Атрибуты рисков:

- **Вероятность** наступления
- **Влияние** на проект (серьезность)

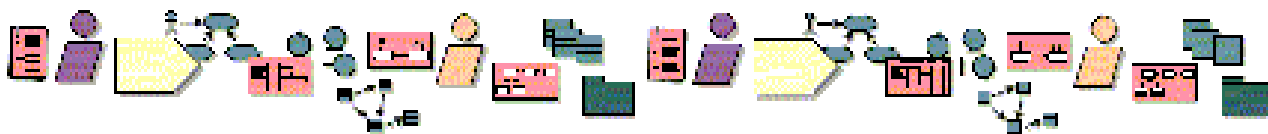
Оба атрибута часто объединяют в единый показатель **значительности** риска: высокий, существенный, умеренный, незначительный, низкий.

Инициализация списка рисков

Когда мы идентифицируем риски, мы рассматриваем, «что может идти неправильно». В самом широком смысле, конечно, все может идти не так, как надо. Однако, это не должно внушать пессимистического представления о проекте; мы хотим идентифицировать потенциальные барьеры к успеху для того, чтобы иметь возможность уменьшить или устранить их.

Говоря более точно, мы ищем те события, при совершении которых уменьшится вероятность поставки проекта с заданными свойствами, необходимого уровня качества, вовремя и в пределах бюджета.

Используйте метод мозговой атаки.



Соберите проектную группу (в начале работы эта группа очень невелика; если в группе больше 5-7 человек, процессом оценки рисков должны заняться руководители). Попросите каждого участника указать риски проекта. Позволяется задавать вопросы, но предложения не должны оцениваться или комментироваться группой. Заполняйте таблицу, пока не убедитесь, что больше рисков не может быть идентифицировано.

Для раскочки предложите ознакомиться со списками рисков из предыдущих проектов (интерактивная версия Rational Unified Process содержит ссылки на опубликованные материалы по этой теме). Распространите указанные списки рисков: часто наблюдение того, что уже было обнаружено, помогает обнаружить новое.

Включите в этот процесс все стадии, и не слишком волнуйтесь относительно формы или повторений; Вы сможете подчистить список позже. Используйте однородные группы специалистов (заказчики, пользователи, техники и так далее). Это облегчает процесс сбора рисков: люди меньше скованы с равными себе по специальности и по положению в иерархии.

Объясните, что участники, говоря о рисках, не обязаны их обрабатывать. Если возникнет необходимость заниматься этим риском, будет и лицо, ответственное за его обработку. Иначе никто не идентифицирует никакие риски (или риски, о которых они говорят, будут тривиальны).

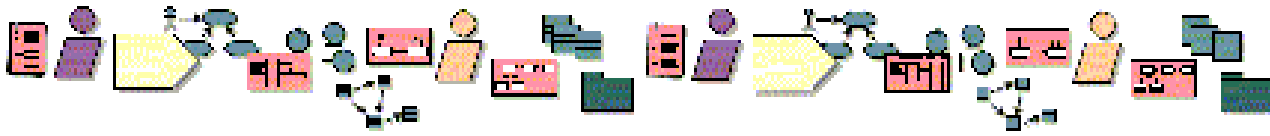
Группировка и ранжирование рисков

Когда рисков больше уже не находится, просмотрите их список с точки зрения наличия естественных группировок (наличия одинаковых рисков), и по возможности объедините риски для устранения дубликатов. Иногда идентифицированные риски могут быть симптомами другого фундаментального дополнительного риска; в этом случае сгруппируйте вместе риски, связанные с этим большим фундаментальным риском.

Предложите каждому участнику ранжировать риски по десятибалльной шкале с точки зрения их важности. Обобщите мнения участников, сравните их и пересортируйте риски.

Поскольку оценки вероятности и стоимости дороги и рискованны сами по себе, как правило, полезно оценить воздействие только первых 10 - 20 рисков. Маленькие проекты могут рассматривать меньшее количество рисков, принимая во внимание, что большие проекты представляют большую «мишень» и в результате имеют большее количество рисков. Существенно различные мнения следует обсудить подробно, чтобы увидеть, одинаково ли интерпретируется каждый риск. При групповой работе вероятность риска должна быть установлена единогласно.

В дополнение к ранжированию рисков по убыванию опасности воздействия, сгруппируйте риски в категории, основываясь на значительности их воздействия на проект (значительности рисков). В большинстве случаев достаточно пяти категорий:



1. Высокая
2. Существенная
3. Умеренная
4. Незначительная
5. Низкая

Документируйте риски и передайте документы членам проектной группы. В интерактивной версии Rational Unified Process порядок документирования указан в описании артефакта «Список рисков». Там же содержится ссылка на соответствующий шаблон MSWord.

Стратегии поведения

Ключевая идея в управлении риском состоит в том, чтобы не ждать пассивно, когда риск осуществится и станет проблемой или уничтожит проект, а решить заранее, что с ним делать. Вы решаете заранее, что Вы сделаете для каждого предполагаемого риска. Существуют три основных возможности:

- **Предотвращение риска:** реорганизовать проект так, чтобы этот риск не мог на него воздействовать.
- **Передача риска:** реорганизовать проект так, чтобы кто-то другой или что-то другое имели с ним дело (заказчик, продавец, банк, другой элемент и т.д.)
- **Принятие и локализация риска:** учесть этот риск как неизбежный. Подготовить план действий на случай проявления этого риска и контролировать его симптомы.

При принятии риска Вы должны сделать две вещи:

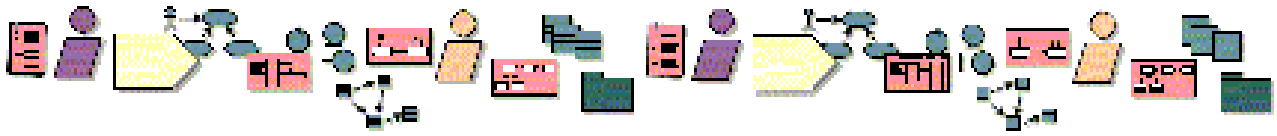
- **Снизить риск:** немедленно предпринять некоторые действенные шаги к снижению вероятности или влияния риска
- **Определить план действий:** какой способ действий выбрать, если риск станет фактической проблемой

В интерактивной версии Rational Unified Process подробно описаны методы выбора стратегии для конкретных рисков, индикаторы проявления рисков и т.п.

Концепция метрик в Rational Unified Process

Зачем нам метрики?

Благодаря метрикам мы можем управлять проектом. Благодаря метрикам мы можем оценить насколько близко или далеко мы находимся от целей, которые мы установили в нашем плане как



окончание, качество, соответствие требованиям и т.д.

Благодаря метрикам мы можем лучше оценить проект, его стоимость и качество, основываясь на прошлом опыте. Наконец, благодаря метрикам по прошествии времени мы можем оценить, как далеко мы продвинулись относительно некоторых ключевых аспектов эффективности процесса, увидеть результаты изменений.

Измерение некоторых ключевых аспектов проекта ведет к дополнительным затратам. Так что мы не пользуемся метриками только потому, что можем это делать. Мы должны установить очень точные цели этой работы, и устанавливать только те метрики, которые смогут удовлетворять этим целям.

Существуют два вида целей:

1. **Цели узнавания:** они выражаются при помощи глаголов типа вычислить, спрогнозировать, проконтролировать. Вы хотите лучше понять ваш процесс разработки. Например, Вы можете оценивать качество продукции, получать данные для прогнозирования результатов тестирования, контролировать зоны тестирования или отслеживать изменения требований.
2. **Цели управления:** они выражаются при помощи глаголов типа увеличить, уменьшить, улучшить или достичь. Обычно Вы заинтересованы наблюдать, как какие-то вещи изменяются или улучшаются через какое-то время, от одной итерации к другой, от одного проекта к другому.

Примеры

- Проконтролировать продвижение относительно плана
- Увеличить степень удовлетворения заказчика

Цели управления нельзя просто транслировать в метрику. Они должны преобразовываться в несколько меньших подцелей (в **цели воздействия**), идентифицирующих части проекта, на которые нужно воздействовать для достижения цели. И мы должны быть уверены, что участвующие в работе люди понимают полезность этого.

Пример

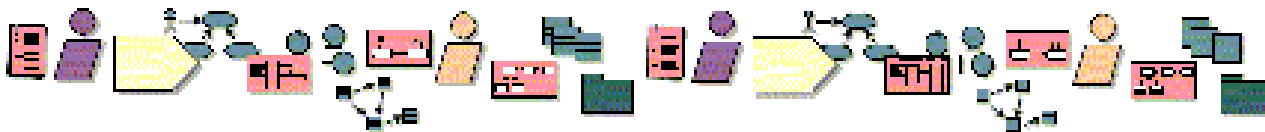
Цель "Увеличить степень удовлетворения заказчика" раскладывается на:

- Определить требования заказчика
- Измерить степень удовлетворения заказчика за несколько выпусков
- Удостовериться, что степень удовлетворения повышается

Некоторые из подцелей (но не обязательно все) требуют, чтобы для них были определены метрики.

Пример

Подцель "Измерить степень удовлетворения заказчика" может быть выполнена на основе:



- Обследования заказчика (где заказчик дает оценки различных аспектов)
- Измерения количества и серьезности вызовов для оперативной поддержки заказчика.

Что такое метрика?

Различаются два вида метрик:

- **Метрика** - измеримый атрибут сущности. Например, объем работ проекта - показатель (то есть, метрика) размера проекта. Чтобы вычислить этот показатель, Вы должны суммировать все время работы над проектом.
- **Первичная метрика** - необработанный элемент данных, который используется для вычисления метрики. В вышеупомянутом примере табель учета времени - первичная метрика. Первичная метрика – это обычно показатель, который существует в базе данных, но не интерпретируется отдельно.

Каждая метрика составляется из одной или более первичных метрик. Следовательно, каждая первичная метрика должна быть ясно идентифицирована, и определена процедура ее сбора.

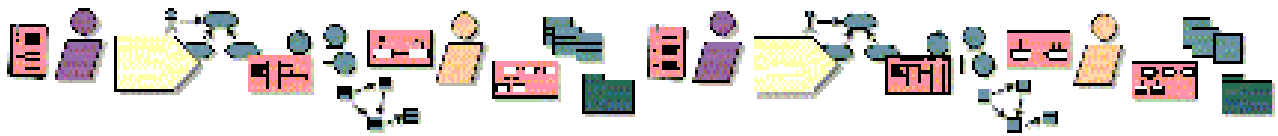
Метрики для поддержки изменений или достижения цели часто через некоторое время «обновляются» (в итерациях или проектах). Нас интересуют тенденции, а не абсолютные значения. Например, чтобы «улучшить качество» мы должны убедиться, что остаточный уровень известных дефектов через какое-то время уменьшился.

Шаблоны

Интерактивная версия Rational Unified Process предлагает шаблоны для описания метрик.

Шаблон для метрики:

Название	Название метрики и все известные синонимы
Определение	Атрибуты сущностей, которые определяются с использованием этой метрики, описание того, какова метрика и из каких первичных метрик она рассчитывается
Цели	Список задач и вопросов, связанных с этой метрикой и некоторое объяснение того, зачем нужна эта метрика
Процедура анализа	Как должна использоваться метрика. Предварительные условия для интерпретации метрики (например, как допустимый диапазон для другой метрики). Целевые значения или тенденции. Модели методов анализа и инструментальные средства, которые нужно использовать. Подразумеваемые предположения (например, среды или моделей). Градуировочные процедуры. Хранение.
Ответственность	Кто собирает и объединяет данные измерений, готовит отчеты и анализирует данные.



Шаблон для первичной метрики:

Название	Название первичной метрики
Определение	Однозначное описание метрики в терминах контекста проекта
Процедура сбора	Описание процедуры сбора. Инструмент сбора данных и форма, которую нужно использовать. Точки жизненного цикла, в которых собираются данные. Порядок проверки. Где будут сохранены данные, формат, степень точности
Ответственность	Кто отвечает за сбор данных. Кто отвечает за проверку данных.

Действия с метриками

С метриками связаны два действия:

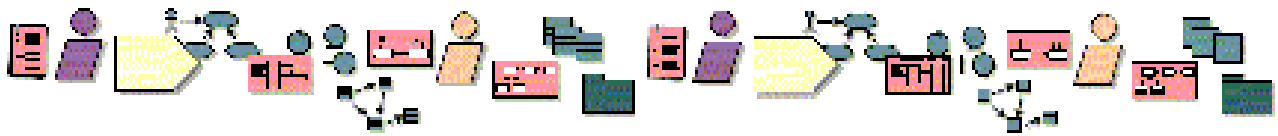
- Определение плана измерений
- Выполнение измерений

План измерений определяется один раз за цикл разработки - в стадии начала, как часть общего плана действий, или, может быть иногда, как часть процесса конфигурирования в усовершенствованном образце. План измерений может быть пересмотрен, подобно любому другому разделу плана работ, в течение проекта.

Повторяйте измерения по крайней мере один раз за итерацию, а иногда и более часто (например, еженедельно на итерации, охватывающей много месяцев).

Собранные результаты измерений - часть документа Оценка состояния, который используется для оценки продвижения и жизнеспособности проекта. Впоследствии они могут также использоваться для оценки проекта и направлений его организации.





Поток работ «Управление проектом» (продолжение)

О планирование итерационного проекта

Я думаю, если Вы дочитали до этого места, Вы уже стали убежденным сторонником итерационного подхода. Но когда подойдет время фактического планирования своего проекта, Вы будете несколько озадачены:

- Сколько итераций нужно запланировать?
- Как долго должна продолжаться каждая из них?
- Как определить содержание и цели итераций?
- Как отслеживать продвижение итераций?

Напомним о целях планирования проекта:

- Распределение задач и обязанностей членов проектной группы на определенный период
- Контроль продвижения проекта относительно плана, обнаружение отклонений и принятие корректирующих мер

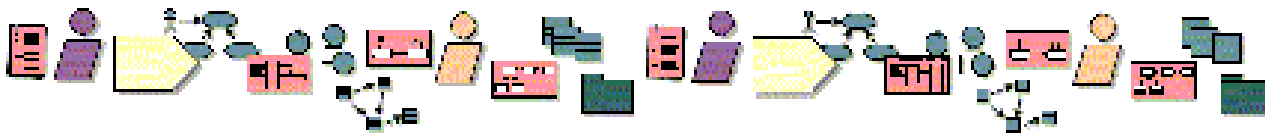
Планирование имеет дело также и с неодушевленными ресурсами, типа оборудования, инструментальных средств и бюджета, но мы не будем останавливаться на этих вопросах, потому что итерационный подход не оказывает на них существенного влияния.

Двухуровневое планирование

Изначально разработчики программных систем пытались планировать большие проекты от начала до конца в мельчайших деталях. Конечно же эти попытки были обречены на неудачу. Ведь чтобы такие планы были реальными, нужно иметь очень хорошее понимание того, что будет сформировано, нужно иметь устойчивые требования и устойчивую архитектуру, и нужно иметь опыт создания подобной системы. Этот опыт позволил бы получить детальную структуру трудозатрат.

Как можно планировать, чтобы Иванов закодировал модуль EQPT на 37 неделе, если при планировании Вы даже не знаете о существовании такого модуля?

Этот подход хорошо работает в тех отраслях промышленности, где структура трудозатрат является более или менее стабильной, а номенклатура задач детерминирована, например, основными



законами физики.

Другая крайность – утверждение о полной невозможности планирования проектов разработки больших (и маленьких) программных систем. Отсюда – полная потеря управляемости, нарушения сроков, прекращение финансирования и т.д. Может быть, именно здесь и прячутся те ужасные проценты неудачных проектов, о которых мы писали в Предисловии?

Но истина не «черная» и не «белая». Она, как всегда, где-то посередине.

Rational Unified Process рекомендует вести разработку, основываясь на двух видах планов:

- Крупномодульный план: План проекта
- Ряд мелкомодульных планов: Планы итераций

План проекта

План проекта – крупномодульный план, который разрабатывается для всего проекта только один раз. Он полностью определяет «границы» проекта для одного цикла разработки и содержит:

- Даты главных вех:
 - целей жизненного цикла (конец стадии начала, рамки проекта и его финансирование)
 - архитектуры жизненного цикла (конец стадии уточнения, законченная архитектура)
 - начальной работоспособности (конец стадии конструирования, первая «бета»-версия)
 - выпуска изделия (конец стадии перехода и полного цикла разработки)
- Потребности в укомплектовании персоналом: какие ресурсы требуются и к какому времени
- Даты малых вех: конец каждой итерации и ее главные цели, если они известны

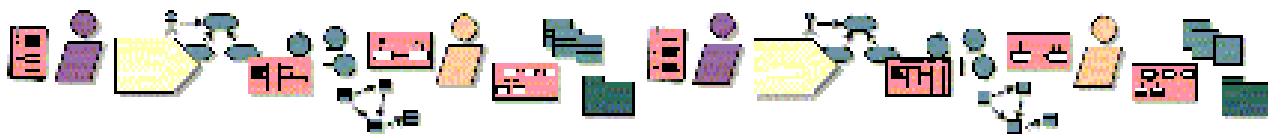
План проекта создается как можно раньше в стадии начала и модифицируется по мере необходимости.

План итерации

План итерации – мелкомодульный план, который создается один раз для каждой итерации. Обычно проект имеет два «активных» плана итерации одновременно:

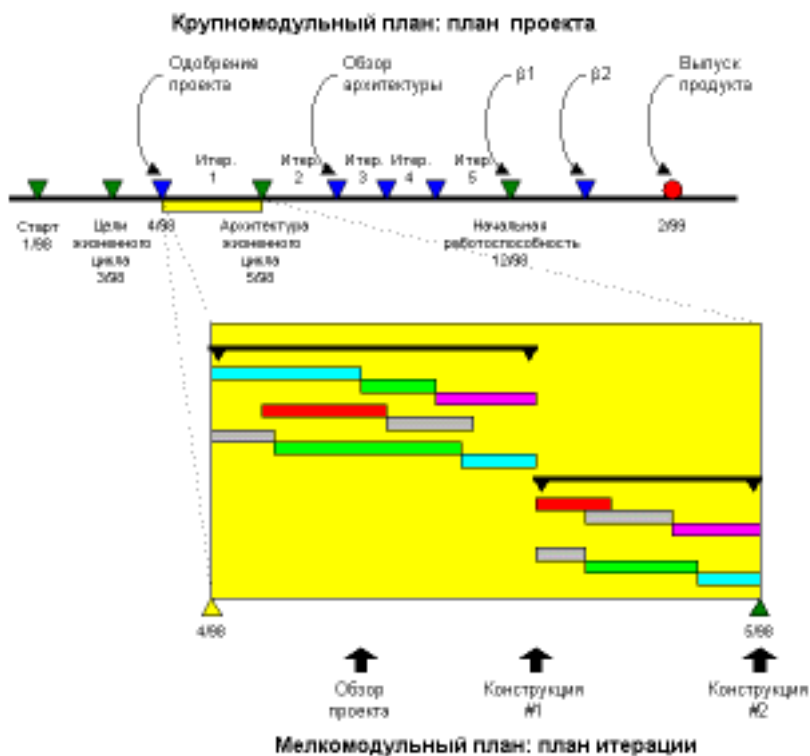
- План текущей итерации, который используется, чтобы проследить продвижение итерации
- План следующей итерации, который обычно начинают формировать во второй половине текущей итерации и который бывает готов к концу текущей итерации

План итерации формируется для определения задач и их распределения между личностями и группами с использованием традиционных методов и инструментальных средств планирования (диаграммы Ганта и т.д.). План должен содержать важные даты, типа сроков завершения



компонентов, получения компонентов от других организаций, главных обзоров и т.п.

Вы можете представить себе план итерации как «увеличительное стекло», перемещаемое вдоль плана проекта, как показано на рисунке ниже:



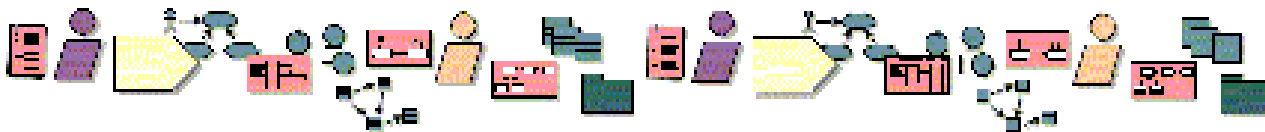
План итерации – это «увеличительное стекло», перемещаемое вдоль плана проекта

Поскольку итеративный процесс является динамичным и, как предполагается, приспособлен к изменениям в целях и в тактике, нет необходимости тратить время на детальное планирование действий, которые «скрыты за горизонтом». Эти планы все равно быстро устареют и будут игнорироваться проектной группой. План итерации охватывает реально управляемый отрезок времени и имеет нужную степень детализации для реального планирования.

Разработка плана проекта

Основные шаги разработки плана проекта уже перечислялись в разделе «Краткий обзор действий». Это:

- Определение вех проекта
- Определение целей вех
- Определение количества и длины итераций в пределах стадий



- Уточнение сроков вех и контекста
- Определение плана измерений

Определение главных вех проекта

План проекта, прежде всего, определяет сроки и характер главных вех. Эта часть плана служит подробным «атласом путей» для проекта и создается в самом начале проекта.

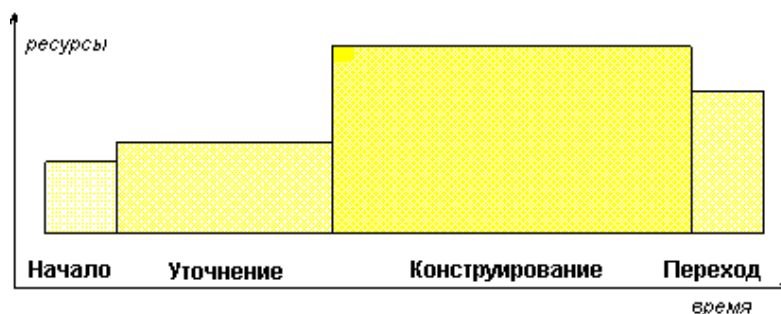
Чтобы планировать стадии проекта в начальном цикле развития, Вам, вероятно, придется сделать некоторые предположения на основе:

- Опыта работы над проектами, подобными по характеру и области применения.
- Степени новизны.
- Ограничений среды, например, быстродействия, распределенности и безопасности.
- Готовности организации.

Стадии занимают разное количество времени и имеют разные объемы работ. Хотя они значительно различаются в зависимости от проекта, типичный начальный цикл разработки для проекта среднего размера должен иметь примерно следующее соотношение между объемом работ и временем выполнения:

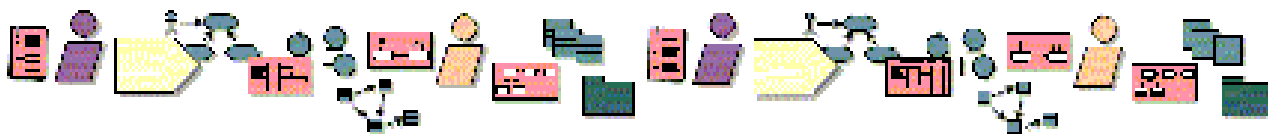
	Начало	Уточнение	Конструирование	Переход
Объем работы	~5 %	20 %	65 %	10%
Время	10 %	30 %	50 %	10%

и графически может быть изображен так:



Относительные затраты ресурсов по стадиям типичного начального цикла разработки проекта среднего размера

Для цикла развития стадии начала и уточнения будут значительно меньше. Инструментальные средства, которые могут автоматизировать некоторую часть объема работ конструирования, могут уменьшить ее долю, сделав стадию конструирования намного меньше, чем стадии начала и уточнения вместе взятые.



Определение целей вех

Каждая веха фокусируется на определенном выпуске; каждая обеспечивает четкую точку перехода в следующую стадию

Стадия	Веха	Цель
Начало	Цели жизненного цикла	Зафиксировать ресурсы проекта
Уточнение	Архитектура жизненного цикла	Стабилизировать архитектуру продукта
Конструирование	Начальная работоспособность	Завершить разработку продукта
Переход	Выпуск изделия	Успешно развернуть продукт

Каждая веха представляет собой барьер, который должен преодолеть проект; у каждой вехи ставится задача принять решение о продолжении разработки проекта.

Веха целей жизненного цикла

Оценочные критерии для стадии начала:

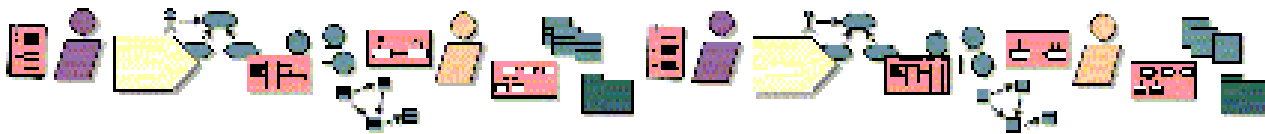
- Согласие с совладельцами при определении оценок контекста и стоимости/графика
- Понимание требований, которые подтверждаются точностью описания одного из основных прецедентов
- Степень точности оценок стоимости/графика, приоритетов, рисков и процесса развития
- Глубина и широта архитектурного прототипа
- Фактические расходы относительно запланированных расходов.

При достижении вехи целей жизненного цикла рассматривается описание деловых обстоятельств, чтобы гарантировать, что проект экономически жизнеспособен. План проекта (хотя и грубо) рассматривается с точки зрения допустимости и непротиворечивости с деловыми обстоятельствами и с принятой практикой:

- Оценки ресурсов в плане проекта должны быть совместимы с бюджетом проекта (обратите внимание на время, персонал и, особенно, затраты на развитие среды)
- План проекта не должен учитывать уровни производительности, существенно отличающиеся от среднего опыта; если проект существенно сложнее, чем те, что организация успешно выполняла прежде, оценки производительности должны масштабироваться позже, чтобы отразить большую сложность проекта.

Веха архитектуры жизненного цикла

Главные оценочные критерии для стадии уточнения включают ответы на вопросы:



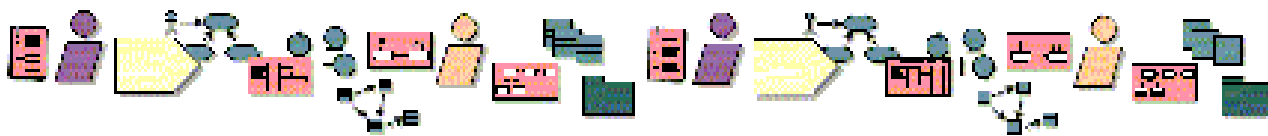
- Имеется ли устойчивое видение программы?
- Действительно ли устойчива архитектура?
- Показывает ли выполненная демонстрация, какие главные элементы риска были обнаружены и достоверно разрешены?
- Достаточно ли детализирован и точен план стадии конструирования, и основан ли он на достаточно вероятных оценках?
- Все ли совладельцы согласны, что предложенное видение может быть реализовано, если будет выполнен текущий план разработки законченной системы в контексте текущей архитектуры?
- Соответствуют ли фактические расходы ресурсов запланированным приемлемым расходам?

При достижении вехи архитектуры жизненного цикла архитектура программы должна быть устойчивой. Потребность стабильности диктуется природой стадии конструирования: при конструировании проект обычно расширяется, увеличивается число параллельно работающих разработчиков, свободно поддерживающих связь с коллегами, создающими программу. Степень независимости и параллельности, необходимая при конструировании, просто не может быть достигнута, если архитектура не устойчива.

Важность устойчивой архитектуры нельзя недооценивать. Не обманывайтесь мыслью «достаточно хорошо - закрываем» – нестабильность есть нестабильность, и лучше задержать начало конструирования и прежде получить правильную архитектуру, чем продолжать работу. Изменения архитектуры в ходе конструирования наносят ощутимые удары: они имеют тенденцию быть дорогими, разрушительными и деморализующими.

Реальная трудность оценки стабильности архитектуры состоит в том, что Вы хотите оценить «то, не знаю что»; стабильность оценивается относительно ожидаемого изменения. В результате, стабильность - по существу субъективный показатель. Но мы можем обосновывать эту субъективность больше, чем только догадками. Сама архитектура разрабатывается при рассмотрении «архитектурно существенных» сценариев - подмножества прецедентов, представляющих наиболее перспективное поведение, которое должна поддержать система. Оценка стабильности архитектуры включает гарантию того, что архитектура имеет некоторую свободную зону, что в архитектуре не возникнет никаких «сюрпризов» при дальнейшем продвижении.

Хорошим индикатором может быть также опыт работы с архитектурой: если изменения в архитектуре малы и остаются малыми, когда вводятся новые сценарии, есть все основания полагать, что архитектура стабильна. Наоборот, если каждый новый сценарий вызывает изменения архитектуры, она все еще находится в развитии, и ее устойчивость еще не гарантирована.



Веха начальной работоспособности

Оценочные критерии для стадии конструирования включают ответы на вопросы:

- Этот выпуск программы достаточно устойчив и зрел, чтобы развернуть его у пользователя?
- Действительно ли все совладельцы готовы к передаче изделия пользователю?
- Приемлемо ли еще соотношение фактических и запланированных расходов?

Если имеются сбои проекта, начало стадии перехода, вероятно, придется отложить на один выпуск, чтобы достичь этой вехи.

При достижении вехи начальной работоспособности программа должна быть готова к передаче группе развертывания. Все функциональные возможности должны быть уже разработаны. В дополнение к программному обеспечению должно быть разработано руководство пользователя и описание текущего выпуска.

После этой вехи программное обеспечение выпускается для испытания «бета»-версии. Начинается обучение пользователей и планируется промышленное развертывание.

Веха выпуска изделия

Первичные оценочные критерии для стадии перехода включают ответы на вопросы:

- Удовлетворен ли пользователь?
- Приемлемо ли соотношение фактических и запланированных расходов?

При достижении вехи выпуска изделия программа завершена, и начинается цикл эксплуатации результирующего выпуска. Сюда может входить запуск нового цикла или сопровождение некоторого дополнительного выпуска.

Определение количества и длины итераций в пределах стадий

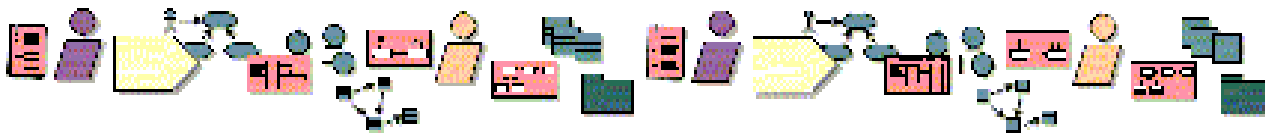
Когда будут определены длины стадий проекта, должны быть определены число итераций и их длина.

Длина итерации

Рекомендации по продолжительности выполнения итераций зависят, главным образом, от количества людей, занятых в проекте.

Например:

- Пять человек могут составить некоторый план в понедельник утром за совместным завтраком, каждый день за ланчем контролировать продвижение и перераспределять задачи, начать компоновку в четверг, и завершить итерацию к вечеру пятницы.
- Но этого будет очень сложно достичь, если работают 20 человек. Потребуется больше времени



для распределения работ, синхронизации работы подгрупп, интеграции и т.д. Итерация, скорее всего, займет 3 или 4 недели.

- При работе 40 человек уже не меньше недели потребуется для «прохождения импульса от мозга до конечностей». В этом варианте Вы имеете промежуточные уровни управления, общее понимание цели потребует более формализованной документации и вообще большей формальности. Наиболее вероятная длительность итерации – три месяца.

Играют роль и другие факторы: степень знакомства организации с итерационным подходом, включая наличие устойчивой и зрелой организации, уровень автоматизации, которую группа использует для управления кодом (например, наличие средств управления конфигурацией), обмена информацией (например, внутренняя сеть) и тестирования и т.д.

Некоторые эмпирические данные:

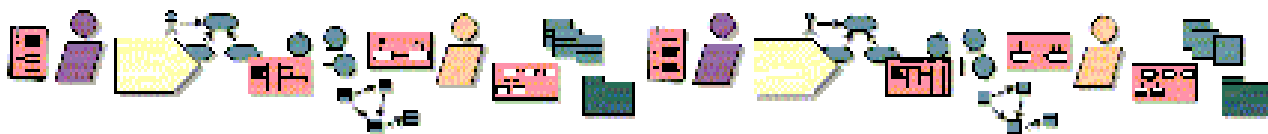
Объем программы в строках кода (SLOC)	Число разработчиков	Продолжительность итерации
10,000	5	1 неделя
50,000	15	1 месяц
500,000	45	6 месяцев
1,000,000	100	1 год

- **Итерации, длительностью более, чем 6 месяцев** вероятно должны иметь встроенные вехи, чтобы сохранить управляемость проекта. Рассмотрите возможность сокращения контекста итерации, чтобы уменьшить ее длину, и гарантировать прозрачность.
- **Итерации, длительностью более, чем 12 месяцев** создают свой собственный риск с точки зрения годового цикла финансирования. Проект, который не произвел чего-либо видимого за прошедшие 12 месяцев, имеет опасность потери финансирования.
- **Итерации, для которых необходимо меньше 1 месяца**, должны быть тщательно ограничены. Как правило, короткие итерации больше подходят для стадии конструирования, где степень вновь включаемых функциональных возможностей и степень новизны низки. Короткие итерации могут содержать маленький или не содержать никакого формального анализа или уточнения, а могут просто с приращением улучшать функциональные возможности.
- **Не все итерации должны иметь одинаковую длину**: их длина изменяется согласно их целям. Как правило, итерации уточнения будут длиннее, чем итерации конструирования. В пределах стадии итерации обычно имеют одинаковую длину - это упрощает планирование.

Количество итераций. Варианты жизненных циклов

Rational Unified Process рекомендует планировать, как правило, **шесть, плюс/минус три** итерации в проекте.

В зависимости от риска, размера и сложности проекта возможно множество вариантов. В



интерактивной версии Rational Unified Process описаны четыре различных стратегии построения жизненного цикла проекта.

■ Пошаговый жизненный цикл

Пошаговая стратегия определяет потребности пользователя, формулирует системные требования и затем исполняет остальную часть разработки в последовательных конструкциях. Первая конструкция содержит часть запланированных возможностей, следующая конструкция прибавляет новые возможности и так далее, пока система не будет закончена.

Для пошагового жизненного цикла характерны следующие итерации:

- короткая предварительная итерация, чтобы установить контекст и видение, и определить деловые прецеденты
- одна итерация уточнения, в течение которой определяются требования и устанавливается архитектура
- несколько итераций конструирования, в течение которых реализуются прецеденты и архитектура излагается в деталях
- несколько итераций перехода для передачи программы сообществу пользователей



Пошаговый жизненный цикл

Эта стратегия пригодна, если:

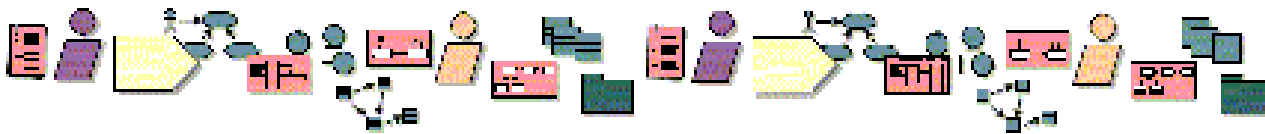
- прикладная область известна
- риски хорошо понятны
- проектная группа надежна

■ Эволюционный жизненный цикл

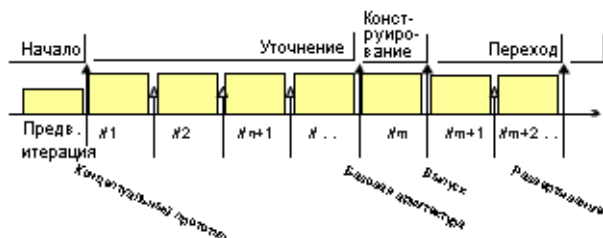
Эволюционная стратегия отличается от пошаговой тем, что потребности пользователя не поняты полностью, и все требования не могут быть определены с первого взгляда, они уточняются в каждой следующей конструкции.

Характерны следующие итерации:

- короткая предварительная итерация, чтобы установить контекст и видение, и определить деловые прецеденты



- несколько итераций уточнения, в течение которых уточняются требования
- одна итерация конструирования, в течение которой реализуются прецеденты и архитектура излагается в деталях
- несколько итераций перехода для передачи программы сообществу пользователей



Эволюционный жизненный цикл

Эта стратегия пригодна, если:

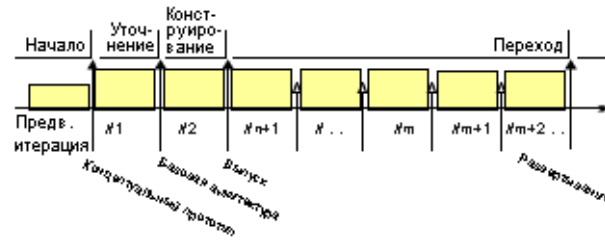
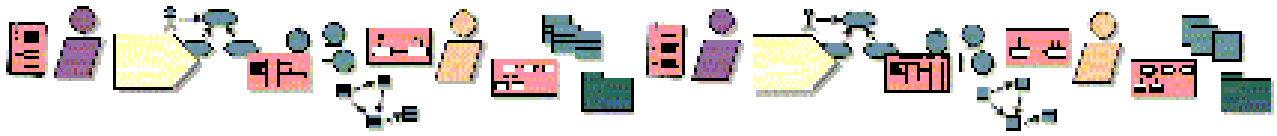
- прикладная область новая или незнакомая
 - группа неопытна
- **Жизненный цикл пошаговой поставки**

Некоторые авторы могут поэтапно поставлять заказчику постепенно возрастающие функциональные возможности. Эта стратегия может быть необходима, когда поджимают сроки поставки продукта на рынок, или когда может быть выгодной ранняя готовность некоторых главных возможностей.

При таком подходе стадия перехода начинается раньше и имеет больше итераций. Эта стратегия требует очень устойчивой архитектуры, которая трудно достижима в начальном цикле развития для «беспрецедентной» системы.

Характерны следующие итерации:

- короткая предварительная итерация, чтобы установить контекст и видение, и определить деловые прецеденты
- одна итерация уточнения, в течение которой komponуется устойчивая архитектура
- одна итерация конструирования, в течение которой реализуются прецеденты и архитектура излагается в деталях
- несколько итераций перехода для передачи программы сообществу пользователей



Жизненный цикл пошаговой поставки

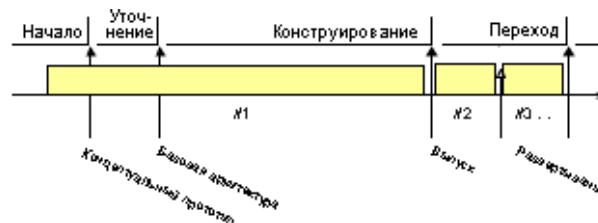
Эта стратегия пригодна, если:

- прикладная область известна, а поэтому архитектура и требования могут быть рано стабилизированы
- проектная группа надежна
- постепенные выпуски с увеличивающимися функциональными возможностями важны для заказчика
- **Жизненный цикл «главного проекта»**

Традиционный подход водопада можно представить как вырожденный случай, в котором имеется только одна длинная итерация стадии конструирования. На практике трудно избежать дополнительных итераций в стадии перехода.

Характерны следующие итерации:

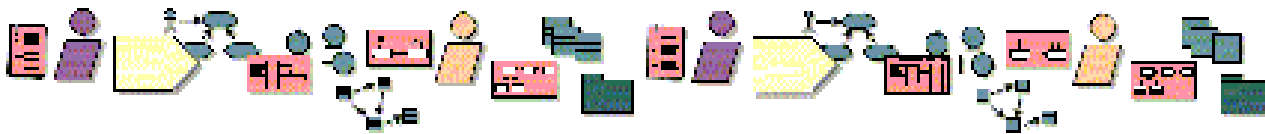
- короткая предварительная итерация, чтобы установить контекст и видение, и определить деловые прецеденты
- одна очень длинная итерация конструирования, в течение которой реализуются прецеденты и архитектура излагается в деталях
- несколько итераций перехода для передачи программы сообществу пользователей



Жизненный цикл «главного проекта»

Эта стратегия пригодна, если:

- маленькие приращения хорошо определенной функциональности добавляются к очень устойчивой программе



- новые функциональные возможности хорошо определены и поняты
- группа имеет опыт работы в прикладной области и с существующей программой

■ Гибридные стратегии

На практике немного проектов строго следуют одной стратегии. Вы часто получаете **гибрид**, какое-либо уточнение вначале, некоторое пошаговое конструирование и многочисленные поставки. Среди преимуществ стадийно-итеративной модели - возможность приспособлять гибридную технологию просто увеличивая длину и количество итераций в определенных стадиях:

- Для сложных или незнакомых прикладных областей, где нужна большая исследовательская работа: увеличьте число итераций стадии уточнения и ее длину.
- Для проектов с проблемами разработки, где возникают трудности транслирования проекта в код: увеличьте число итераций стадии конструирования и ее длину.
- Чтобы поставлять программное обеспечение в ряде постепенно возрастающих выпусков: увеличьте число итераций стадии перехода и ее длину.

Каждая итерация производит выпуск, который является выполнимой программой, используемой для оценки продвижения и качества разработки. Поскольку цели итераций различны, функциональные возможности и законченность выпусков разнообразны. Цели итераций должны быть достаточно определенны, чтобы в конце итерации можно было оценить степень их выполнения. В ранних итерациях цели обычно выражаются в оценках смягчения рисков; в более поздних итерациях цели выражаются в оценках выполнения функциональности и качества.

Уточнение сроков вех и контекста

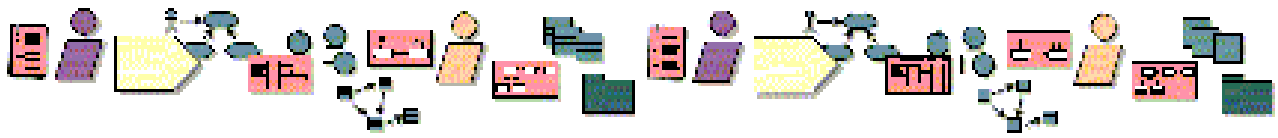
К концу стадии начала стадии могут быть запланированы более точно, принимая во внимание:

- Количество идентифицированных прецедентов
- Изученность сложности прецедентов
- Идентифицированные технические и деловые риски
- Метрики функций или прецедентов
- Результат прототипирования

Этот очень грубый план корректируется в течение стадии уточнения. Он служит основанием для формирования остальной части плана проекта.

Определение плана измерений

План измерений определяется один раз за цикл разработки, в стадии начала, как часть общего плана действий. План измерений может быть пересмотрен в ходе проекта подобно любому другому разделу плана проекта.



Эта работа состоит в следующем:

- Определение основных целей управления
- Проверка правильность целей
- Определение подцелей
- Идентификация метрик, необходимых для удовлетворения подцелей
- Идентификация первичных метрик, которые должны быть собраны для вычисления метрик
- Написание и рецензирование плана измерений
- Подключение механизмов сбора измерений

Разработка плана итерации

Как и в предыдущем разделе, напомним основные шаги, на этот раз, процесса разработки плана итерации. Это:

- Определение рамок итерации
- Определение критериев оценки итерации
- Определение действий итерации
- Распределение ответственности

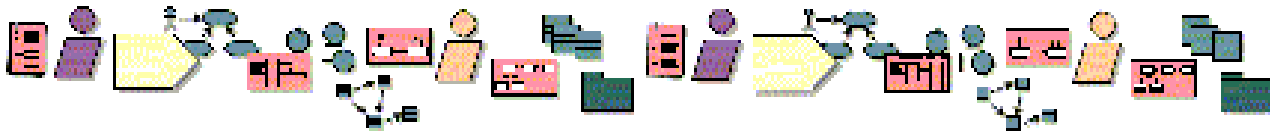
Сама по себе итерация – это набор задач, которые очень конкретно выделяются для создания выполнимой программы. Для всех итераций, кроме последней итерации перехода, это - полуфабрикат, производимый с целью смягчения рисков и доведения проекта до успешного завершения. Акцент на выполнимости программы вынуждает к почти непрерывной интеграции и позволяет рано обнаруживать технические риски и уменьшать сопутствующие риски проекта.

Выполнение итерации подразумевает некоторое количество доработки, и сопровождается изменениями в характере доработки. Короче говоря, чтобы поставить изделие высокого качества, требуется некоторая его доработка: создание полуфабрикатов и оценка пригодности архитектуры **раньше и чаще** повышают качество конечного продукта, при этом делать изменения дешевле и приспособлять их проще

Определение рамок итерации

Рамки итерации управляются четырьмя факторами:

- Главные риски проекта
- Функциональные требования к системе
- Время, отводимое на итерацию в плане проекта



- Стадия и ее цели

В стадии уточнения:

Имеются три наиболее важных фактора, влияющие на определение целей итерации в стадии уточнения:

- Риски
- Критичность
- Охват

Главный фактор для определения целей итерации - **риски**. Вы должны по возможности смягчить или удалить риски. Большинство ваших рисков должно быть смягчено в стадии уточнения, но для ключевых элементов это может быть продолжено в стадии конструирования, так как некоторые риски все же останутся высокими или обнаружатся новые риски.

Но так как целью стадии уточнения является создание базовой архитектуры, в игру вступают некоторые другие соображения, например, убедиться, что архитектура подходит для всех аспектов программного обеспечения, которое должно быть разработано (**охват**). Это важно, так как архитектура будет использоваться для дальнейшего планирования: организация группы, оценка разрабатываемого кода и т.д.

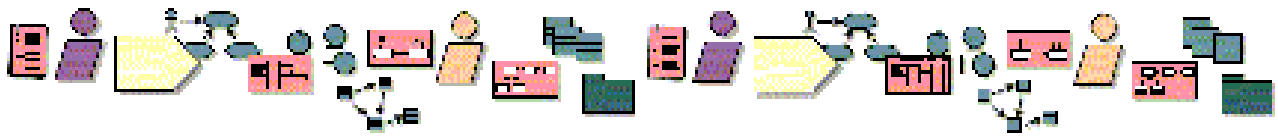
И, наконец, хотя сосредоточение на рисках важно, нужно иметь в виду главное назначение системы. Решение всех трудных проблем - это хорошо, но это не должно делаться в ущерб функциональным возможностям: удостоверьтесь, что критичные функции или услуги системы действительно охвачены (**критичность**), даже если с ними не связаны никакие предполагаемые риски.

Для наиболее разрушительных рисков из общего списка рисков нужно выбрать некоторый сценарий из некоторого прецедента, работа над которым натолкнула бы группу разработки на противодействие риску.

Для удовлетворения **критичности** удостоверьтесь, что большинство основных функций или услуг, которые должна обеспечивать система, включено в план. Выберите из некоторого прецедента некоторый сценарий, который представляет наиболее общую, наиболее частую форму обслуживания или возможность предлагаемой системы. В качестве источника информации при этом используется документ Архитектура программы, в котором архитектор расположил все прецеденты или подпотоки прецедентов по приоритетам для отражения архитектурно существенных прецедентов или сценариев.

Для удовлетворения фактора **охвата** в стадию уточнения включают сценарии, касающиеся областей, которые, как Вы знаете, будут разрабатываться в дальнейшем, хотя они не являются ни критическими, ни опасными.

Часто экономичнее создавать длинные сквозные сценарии, которые будут представлять сразу



много проблем: один сценарий может быть и критичным, и противопоставлять два различных технических риска. С другой стороны, опасно составлять слишком «жирные» сценарии, то есть пытаться охватить слишком много различных аспектов, вариантов и случаев.

Имейте также в виду, что в стадии уточнения некоторые из рисков могут иметь человеческую или программистскую природу: культура группы, обучение, выбор инструментальных средств, новые методы и т.д., и только прохождение итерации смягчает эти риски.

В стадии конструирования:

Риски остаются ключевым двигателем развития проекта в стадии конструирования, тем более что проявляются новые, непредусмотренные риски. Но запускается этот двигатель потребностью иметь законченные прецеденты. Итерации могут быть запланированы сами по себе ради попытки завершения некоторых из наиболее критичных единиц настолько рано, чтобы они могли быть полностью проверены более, чем в одной итерации. Ближе к окончанию проекта главной целью будет устойчивость всех прецедентов.

В стадии перехода:

Главная цель - окончание генерации этой итерации программы. Установленная цель итерации формулируется в терминах фиксации дефектов, уточнения эффективности или применимости. Если возможности были ограничены (или заблокированы) ради выполнения графика проекта (промежуточного этапа или «беты»), теперь они могут быть доделаны или включены, если они не подвергают опасности то, что было достигнуто ранее.

Определение критериев оценки итерации

Каждая итерация приводит к выполнимому выпуску. Выпуск, как правило, не имеет промышленного качества (кроме заключительной итерации стадии перехода), но, тем не менее, он может быть оценен.

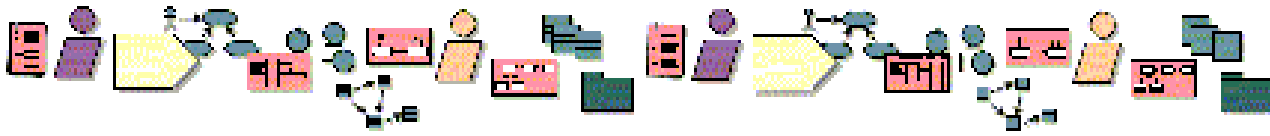
Оценка начальной итерации

Начальная итерация, как правило, сосредотачивается на обосновании концепции программы и формировании поддержки, необходимой для одобрения финансирования проекта. В результате, итеративный выпуск - общий функциональный испытательный прототип, в котором не хватает реального кода выполнения. Оценочные критерии ориентированы на запросы пользователя и качественные метрики.

Основные технические препятствия должны быть преодолены прежде, чем будет обеспечено финансирование продукта; в таком случае оценочные критерии это отразят.

Оценка итераций уточнения

Итерации уточнения фокусируются на создании устойчивой архитектуры. Поэтому критерии оценки уточнения должны сосредотачиваться на оценке стабильности архитектуры. Метрики,



которые могут использоваться:

- Стабильность (или неустойчивость) интерфейса
- Оценка изменения в архитектуре (по сравнению с базовой архитектурой)
- Эффективность ключевых функциональных возможностей

Главная цель состоит в том, чтобы гарантировать, что изменения в стадии конструирования не отразятся на всей системе, вызывая чрезмерную доработку.

Оценка итераций конструирования и перехода

Итерации конструирования и перехода оцениваются по традиционному тестированию программного обеспечения, по объему изменений организации, как например поломки, интенсивность дефектов, и по оценкам обнаруженных неисправностей. Главное в этих итерациях найти ошибки, чтобы они могли быть устранены.

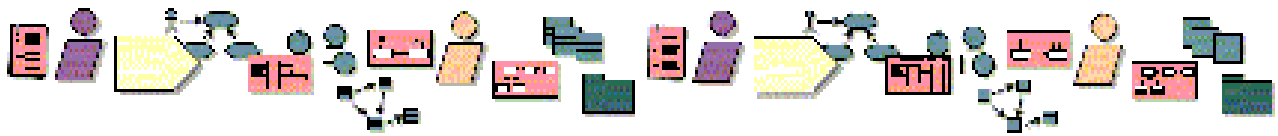
Ошибки обнаруживаются различными способами: инспектирование и обзор кода, тестирование функциональности, эксплуатационные испытания и испытания на нагрузку. Каждая методика ориентируется на обнаружение определенного набора дефектов, и каждая находит свое место. Специально эти методы обсуждаются в потоке работ Испытание.

Определение действий итерации

Необходимо выбрать соответствующий целям итерации набор действий, которые будут выполнены в течение итерации. Как правило, в каждой итерации будут выполняться все действия жизненного цикла программы:

- Выбор прецедентов и сценариев, которые реализуют необходимые функциональные возможности
- Исследование и документирование поведения прецедента (или сценария)
- Анализ и распределение поведения среди подсистем и классов, которые обеспечивают требуемое поведение
- Разработка, реализация и тестирование классов и подсистем
- Интеграция и проверка системы в целом
- Пакетирование внешних выпусков («альфа», «бета» и окончательных) продукта и передача их в среду использования.

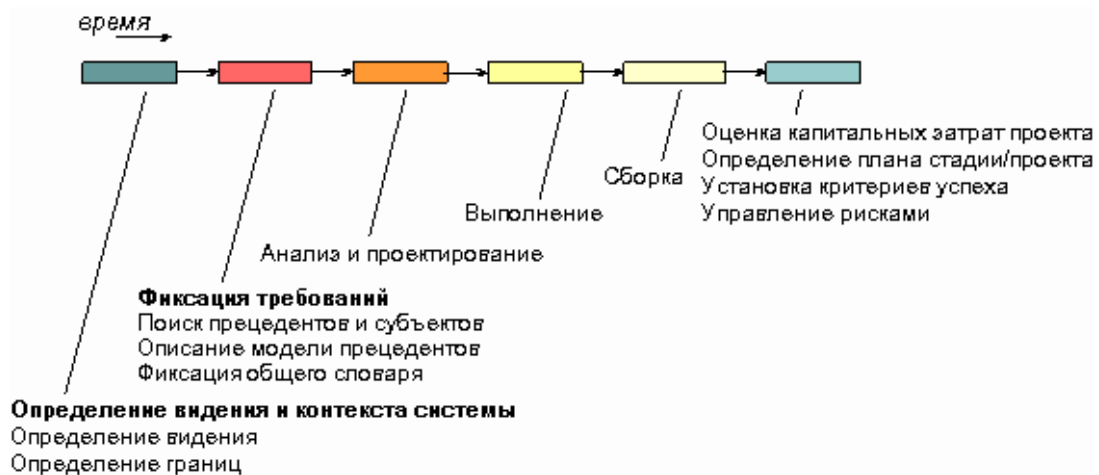
Объем, в котором выполняются эти действия, изменяется в зависимости от итерации и стадии проекта. Отдельные потоки работ (требования, анализ и проектирование, испытание и т.д.) определяют универсальные действия, которые, в свою очередь, приспосабливаются к организации в ходе процесса конфигурирования.



Различные виды итераций

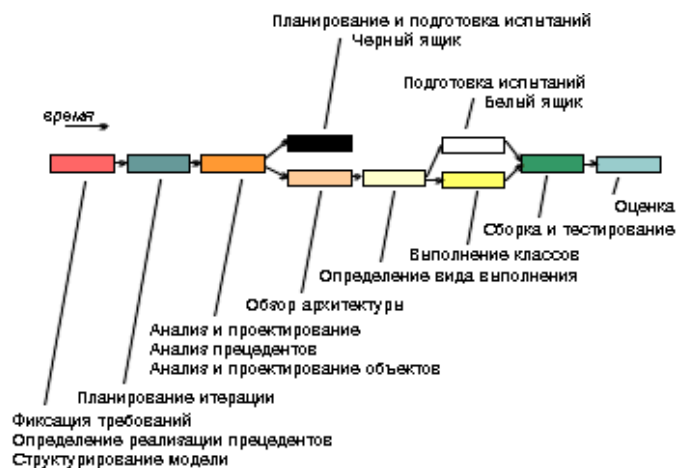
Цели стадий различны, поэтому характер итераций меняется от одной стадии к другой, даже если они проходят через одни и те же действия. Интерактивная версия Rational Unified Process предлагает брать за основу следующие типичные виды итераций:

■ В стадии начала

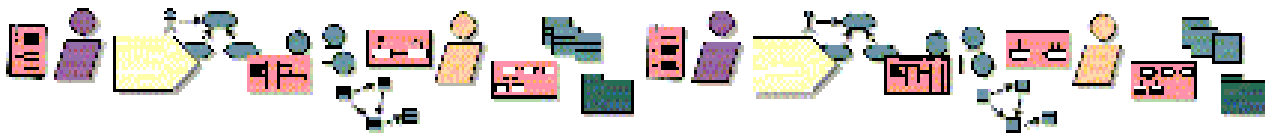


Типичная итерация стадии начала

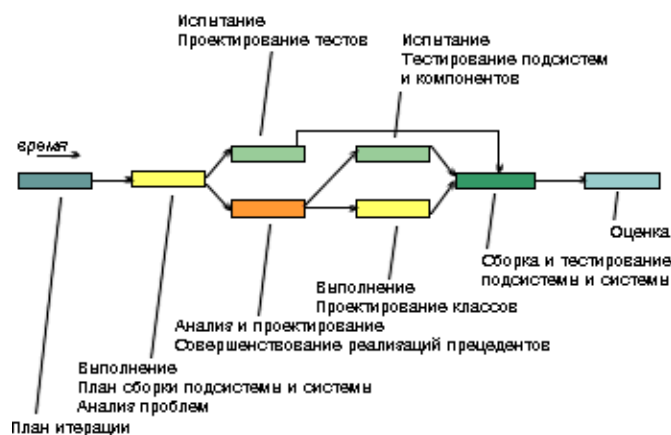
■ В стадии уточнения



Типичная итерация стадии уточнения



■ В стадии конструирования



Типичная итерация стадии конструирования

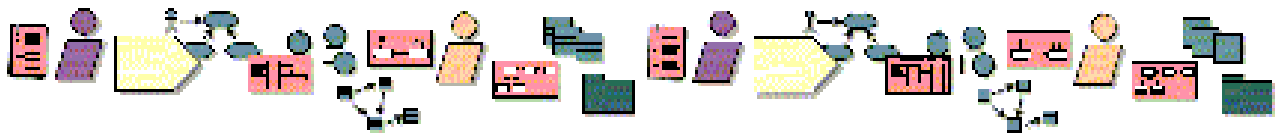
Идентификация затрагиваемых артефактов и включение действий

Когда сценарии или полные прецеденты, которые будут разработаны (плюс устранены дефекты) в итерации выбраны и кратко описаны, Вы должны определить перечень артефактов, которые будут затронуты:

- Какие классы должны быть просмотрены?
- Какие подсистемы затронуты или даже созданы?
- Какие интерфейсы, вероятно, должны быть модифицированы?
- Какие документы должны быть доработаны?

Затем из потоков работ процесса извлекаются включенные в итерацию действия и размещаются в вашем плане. Некоторые действия выполняются один раз за итерацию, некоторые должны быть произведены один раз в классе, в прецеденте, в подсистеме. Далее в план должны быть включены очевидные зависимости между действиями и определен объем работ. Большинство действий, описанных для процесса, достаточно малы, и могут быть выполнены одним человеком или очень маленькой группой людей. Это займет от нескольких часов до нескольких дней.

Вероятно, Вы обнаружите, что в итерации не хватит времени для выполнения всего намеченного. Прежде, чем увеличивать длину итерации (следовательно, отодвинуть срок поставки конечного продукта или сократить число итераций), подумайте о сокращении целей итерации. В зависимости от стадии, в которой Вы находитесь, сделайте сценарии более простыми, сократите или отключите реализацию некоторых возможностей.



Распределение ответственности

Когда набор действий для итерации будет определен, их необходимо распределить между отдельными членами проектной группы. В зависимости от индивидуальных возможностей сотрудников и контекста итерации, действия могут быть выполнены одним человеком или группой. Обзоры и инспекции, конечно, обязательно являются групповыми действиями. Другие действия, типа авторской разработки прецедента или проектирования и реализации классов, являются обычно индивидуальными (кроме случаев, когда младший член группы работает вместе со старшим наставником).

Вообще, за каждое изделие должен отвечать один человек, даже если работа над ним выполнена группой. Это

- прецеденты
- подсистемы
- классы
- тесты и планы проведения испытаний
- и т.д.

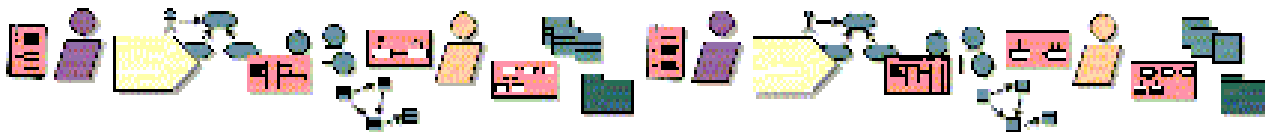
Инструментальная поддержка управления проектом

Шаблоны Microsoft Word

Большинство документов для управления проектом может быть создано с использованием файлов шаблонов, содержащихся в интерактивной версии Rational Unified Process. Эти документы перечислены ниже:

- План разработки программного обеспечения
- Деловые обстоятельства
- Список рисков, краткий обзор
- Список рисков, детальное описание
- План проекта
- План измерений
- Оценка состояния
- План итерации
- Оценка итерации

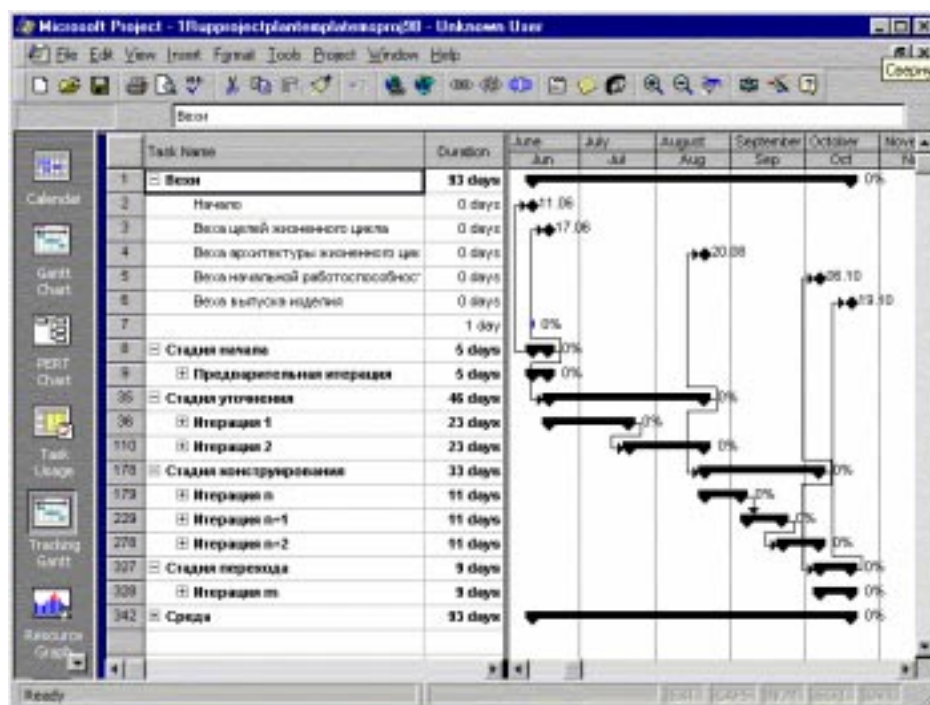
Все шаблоны должны быть установлены в среду вашего Microsoft Word, после чего могут настраиваться для нужд организации и конкретного проекта стандартными средствами.

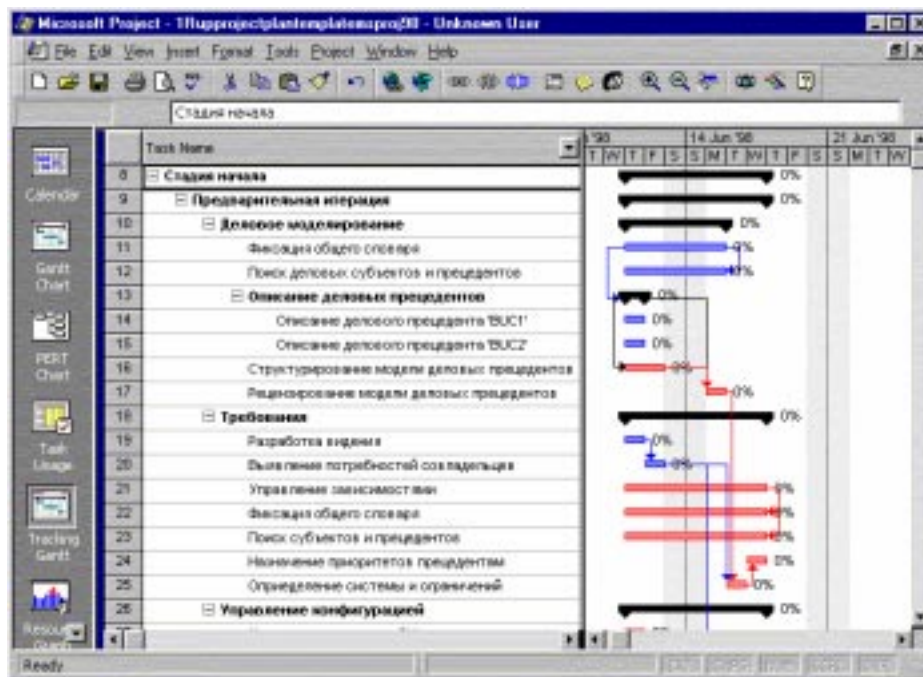
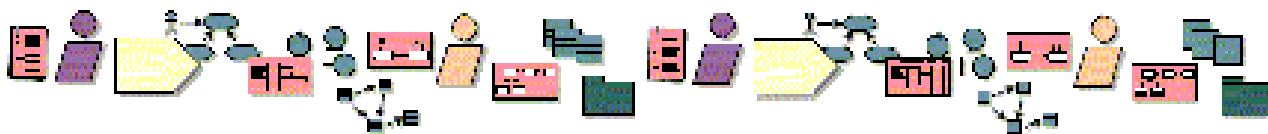


Интерактивная версия Rational Unified Process содержит шаблон плана проекта.

Связанные действия Rational Unified Process:

Шаблон охватывает один цикл разработки, который состоит из всех стадий от стадии начала до стадии перехода, и из номинального числа итераций в каждой стадии, и включает действия и работников, как это определено в основных потоках работ Rational Unified Process.





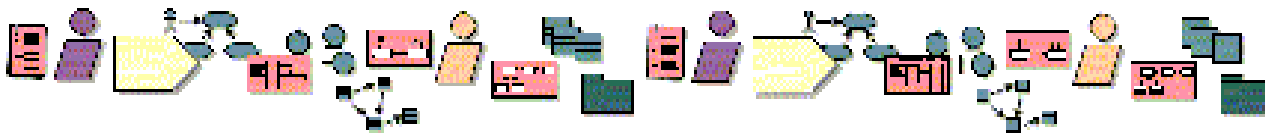
Тот же шаблон отображен со степенью детализации и в масштабе времени, удобными для работы с планом итерации (конкретно – с планом предварительной итерации стадии начала)

Подход к порождению и эксплуатации плана проекта должен быть итерационным. В начале делаются грубые оценки на основе планов стадий и планов итераций в пределах стадий. Известная информация отражается в плане, и по мере изучения план усовершенствуется на базе новой информации. Поэтому некоторые шаги планирования обязательно повторяются по мере продвижения проекта.

При работе с шаблоном плана проекта выполняются следующие действия:

1. Корректировка даты начала проекта
2. Корректировка плана стадии
3. Корректировка плана итерации
4. Корректировка назначения ресурсов
5. Расчет параметров точной настройки графика

Все эти действия выполняются стандартными средствами Microsoft Project. Интерактивная версия Rational Unified Process содержит некоторые практические рекомендации руководителю проекта по работе с Microsoft Project.



Использование Rose Planner Link

Стандартные средства Microsoft Project подразумевают достаточно большую и трудоемкую работу по адаптации предопределенного шаблона к конкретному проекту.

Так, при изменении количества итераций в стадиях, Вы должны вручную их копировать, вставлять или удалять. Действовать при этом нужно весьма осторожно, чтобы сохранить все необходимые связи между «задачами» Microsoft Project.

Перед началом каждой итерации Вы должны вручную добавить запланированные на итерацию детали для каждой из задач типа «Описание деловых прецедентов», «Описание деловых работников», «Описание деловых сущностей» и т.д.

Но все необходимые сведения для этого уже существуют в той или иной форме:

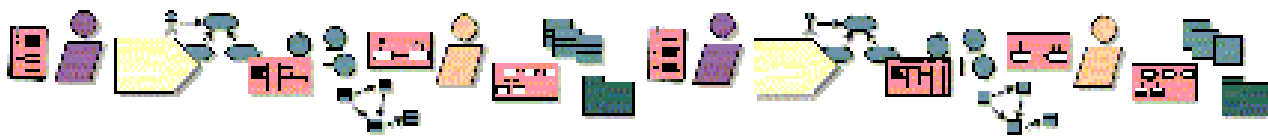
- Структура проекта отображена инженером-технологом в так называемом Усовершенствованном образце Rational Unified Process в потоке работ Среда.
- Деловые прецеденты, работники и сущности перечислены аналитиком делового процесса и проектировщиком предметной области в потоке работ Деловое моделирование.

И было бы странным не воспользоваться этой информацией для автоматизации настройки предопределенного шаблона. Программа **Rose Planner Link (RPL)** компании Ensemble Systems Inc. обеспечивает связь между группой разработки системы и руководителем проекта. RPL автоматически генерирует шаблон Microsoft Project на базе Усовершенствованного образца Rational Unified Process и обеспечивает двустороннюю интеграцию между Rational Rose 2000 и Microsoft Project 98.

Следует отметить, что Rose Planner Link не входит ни в один комплект поставки Rational Rose. Эта программа может быть приобретена отдельно.

Выводы

- Поток работ Управление проектом помогает в балансировании конкурирующих целей, управлении рисками и преодолении ограничений ради поставки изделия, отвечающего потребностям обеих категорий заказчиков – оплачивающих счета и конечных пользователей.
- Разработка в итеративном процессе выполняется под управлением плана проекта (плана стадий) и ряда планов итераций.
- Риск – «навигатор» планирования.
- Измерение - ключевая технология, поддерживающая управление проектом.
- План проекта формируется как компромисс между укомплектованностью персоналом, графиком и контекстом проекта.
- Критерии, определяющие контекст итерации, изменяются от стадии к стадии.



Поток работ «Деловое моделирование»

Большинство разрабатываемых сегодня программных систем - это повседневные инструментальные средства, используемые людьми в их производственной деятельности. Эти системы должны органично «вписываться» в организацию, в которой они используются. Разработчики должны убедиться, что их системы помогут людям в работе и что люди не воспримут их как «балласт», навязываемый начальством, который ничего не дает и только мешает спокойно жить. Чтобы лучше отвечать этим требованиям, разработчик должен попытаться понять деловую сферу прежде или параллельно с разработкой проекта программного обеспечения.

Эта глава объясняет, когда и зачем нужно выполнять деловое моделирование и обсуждает некоторые пути его реализации. Кроме того, она кратко описывает, как из деловых моделей получаются требования к программному обеспечению.

Введение

Цели делового моделирования

Rational Unified Process следующим образом определяет цели делового моделирования:

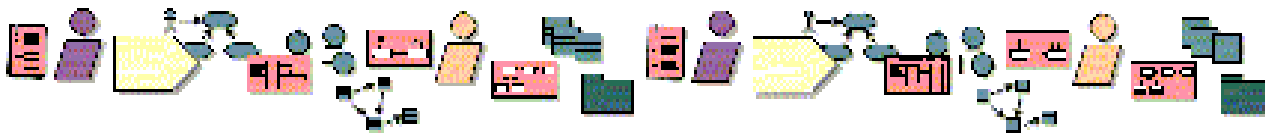
- Понять структуру и динамику организации.
- Гарантировать, что заказчики, конечные пользователи и разработчики имеют одинаковое понимание организации.
- Получить требования к системе для поддержки организации.

Для достижения этих целей разрабатываются модель деловых прецедентов и модель деловых объектов.

В дополнение к этим моделям разрабатываются следующие артефакты:

- Дополнительная деловая спецификация
- Глоссарий

Деловое моделирование - не что-то такое, что нужно делать в каждом случае разработки программного обеспечения. На самом деле, деловые модели имеют существенное значение, когда в непосредственное использование системы включено большое количество людей, совместно обрабатывающих деловую информацию. Например, для добавления новой возможности программного обеспечения существующего телекоммуникационного переключателя Вам не потребуется делового моделирования. С другой стороны, если бы Вы разрабатывали систему



управления заказами для поддержки поставок телекоммуникационных решений, деловое моделирование имело бы большую ценность. Процессы заказа и поставки в этой области сложны, потому что они связаны с продажей решений, изготовленных на заказ, и изделий, которых нет в наличии.

Сферы применения делового моделирования

Несколько лет назад был очень популярен и воспринимался как синоним «революционного прорыва» в управлении организациями термин «реорганизация делового процесса» (Business-Process Reengineering - BRP). По сравнению с деловым моделированием, которое мы будем обсуждать ниже, BRP - намного более амбициозное понятие, включающее помимо составления диаграмм существующей организации также и формирование предлагаемой организации, и осуществление нового пути выполнения вещей. Описываемая методика моделирования также может использоваться как часть этого процесса. Но это далеко не главное ее назначение.

Возможные сферы применения делового моделирования проще всего продемонстрировать на примерах:

- Вы хотите отобразить схему организации и ее процессов, чтобы лучше понять требования к создаваемому приложению.

В этом случае деловое моделирование - часть проекта разработки программного обеспечения, выполняемая прежде всего другого в начальной стадии.

- Вы строите приложение, основной целью которого является предоставление информации.

В этом случае вам может оказаться достаточной статическая модель информации, без рассмотрения потоков работ. Такую модель называют моделью предметной области. Моделирование предметной области – это часть проекта разработки программного обеспечения, выполняемая в начальной стадии и в стадии уточнения.

- Вы строите «большую» систему или семейство приложений.

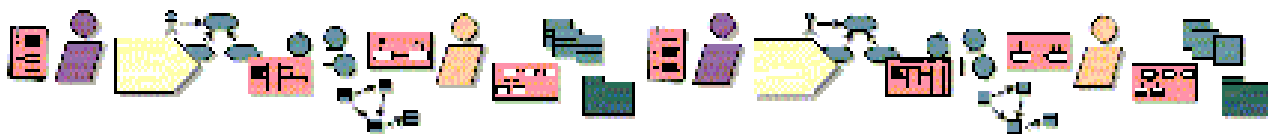
В этом случае деловое моделирование может выполняться как самостоятельный проект. Разработанная модель будет снабжать исходной информацией несколько проектов разработки программного обеспечения. Деловые модели помогут Вам определить функциональные требования и сформировать архитектуру семейства приложений.

- Вы разрабатываете приложение, которое будет использоваться несколькими организациями.

В этом случае деловое моделирование может быть полезным для унификации процессов в организациях, чтобы избежать, насколько это возможно, неоправданной сложности системы.

Если же унификация процессов невозможна, деловое моделирование поможет Вам лучше понять и управлять различиями в том, как организации будут использовать приложение, и расставить функциональные возможности приложения по приоритетам.

- Организация решила специализироваться в новой деловой сфере и внедрить



информационную систему для поддержки этой своей деятельности.

В этом случае целью делового моделирования может быть не только определение требований к системе, но и определение выполнимости новой специализации. Деловое моделирование может выполняться как самостоятельный проект.

- Если предприятие решило полностью обновить порядок своей деятельности (реорганизовать деловые процессы), деловое моделирование выполняется как один или несколько самостоятельных проектов. Реорганизация деловых процессов обычно производится в несколько этапов: формирование представления о новом порядке действий, изучение существующих деловых процессов, проектирование новых деловых процессов и внедрение новых деловых процессов.

Поток работ делового моделирования, описанный в Rational Unified Process, фокусируется на трех первых случаях. Однако описанные методы могут использоваться во всех случаях.

Язык моделирования

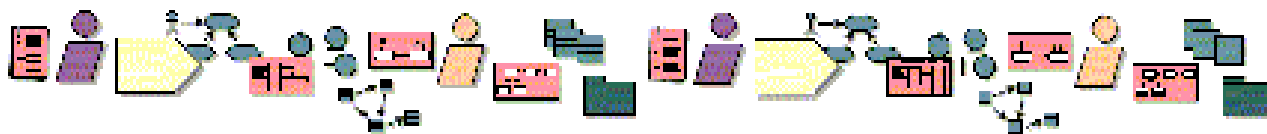
Как и во всех других моделях Rational Unified Process, изобразительным средством делового моделирования является UML.

Одно из главных преимуществ использования методов моделирования программного обеспечения для делового моделирования – это то, что Вы говорите на одном языке. Будет намного проще понять, как что-то, описанное в деловой сфере, касается чего-то, принадлежащего сфере программирования. То же относится и к описанию связей между артефактами в деловых моделях и соответствующими артефактами в системных моделях.

Кроме того, и исторически, методы моделирования, которые родились и созрели в сфере программирования, открывали новые пути визуализации организации. А поскольку объектно-ориентированные методы визуального моделирования обычны для новых программных проектов, использование подобных методов в деловой сфере выглядит совершенно естественным.

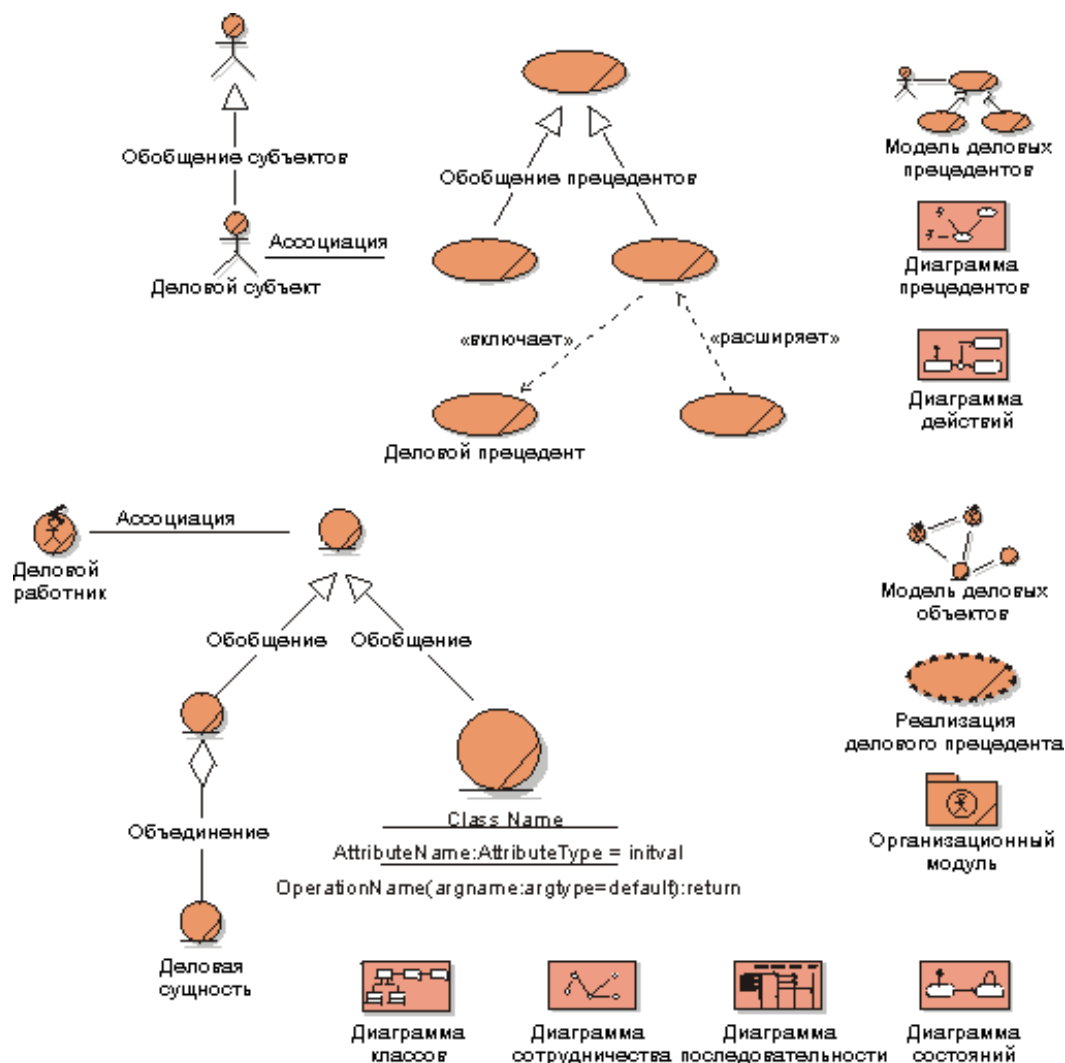
Для обозначения элементов деловых моделей Rational Unified Process использует стереотипы стандартных элементов UML.

Rational Unified Process предполагает, что участники проекта знакомы с UML. Поэтому интерактивная версия RUP не содержит его описания. Вместо этого RUP содержит подробные методические рекомендации по способам обследования объектов моделирования и по особенностям использования языка (UML), инструментальных средств поддержки моделирования (Rational Rose и Rational Requisite Pro) и документирования (Rational SoDA).



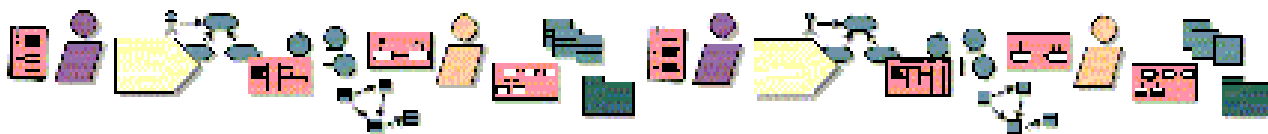
Основные понятия

Номенклатура основных понятий делового моделирования (модели, диаграммы, объекты моделей и их связи) представлена на схеме внизу.



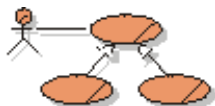
Основные понятия делового моделирования. Каждое понятие представлено своим стандартным обозначением в Rational Unified Process

Деловое моделирование в Rational Unified Process рассматривает деловую сферу с двух точек зрения: **внешней** (что полезного делает деловая сфера для своих контрагентов – **модель деловых прецедентов**) и **внутренней** (как деловая сфера это делает с позиций внутреннего устройства и функционирования – **модель деловых объектов**). На иллюстрации этим моделям отведены, соответственно, верхняя и нижняя части.



Рассмотрим некоторые из понятий делового моделирования более подробно.

Модель деловых прецедентов

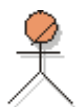


Модель деловых прецедентов - модель, которая описывает процессы деловой сферы и их взаимодействие с внешней средой, например, с заказчиками и партнерами.

Основная цель создания модели деловых прецедентов состоит в том, чтобы описать, как используется деловая сфера заказчиками и партнерами. При этом могут быть показаны действия, которые непосредственно касаются заказчика или партнера, а также действия по поддержке или управлению задачами, которые косвенно касаются внешней стороны дела.

Модель описывает деловую сферу в терминах деловых субъектов и деловых прецедентов.

Деловой субъект



Деловой субъект представляет роль, которую кто-то или что-то играет в деловой среде.

Деловой субъект имеет **имя**.

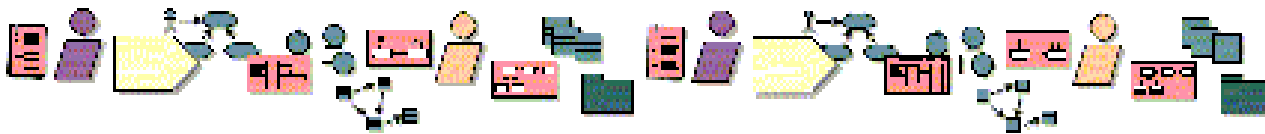
Чтобы понять цель деятельности, Вы должны знать **кто** взаимодействует с деловой сферой; то есть, кто размещает запросы или заинтересован в получении ответа. Различные типы «взаимодействующих субъектов» представляются как **деловые субъекты**.

Термин «субъект» означает роль, которую играет кто-то или что-то при взаимодействии в деловой среде. Ниже перечислены типы пользователей, которые потенциально могут быть деловыми субъектами:

- Заказчики
- Поставщики
- Партнеры
- Потенциальные заказчики («рынок»)
- Властные органы
- Коллеги из не моделированных частей деловой сферы.

Чаще всего субъект соответствует пользователю-человеку. Однако, бывают ситуации, когда роль субъекта играет, например, информационная система. Если сетевые услуги вашего банка настолько хороши, что Вы можете управлять большинством операций банка с персонального компьютера из своего помещения, то ваш прецедент, взаимодействующий с субъектом «поставщик денег» (банк), фактически взаимодействует с информационной системой.

Субъект представляет специфический тип делового пользователя, а не реального физического



пользователя. Несколько физических пользователей деловой сферы могут играть относительно нее одну и ту же роль; то есть они действуют как экземпляры одного и того же субъекта. В то же время один и тот же субъект может действовать как несколько различных субъектов. Это означает, что один и тот же человек может воплощаться в экземплярах различных субъектов.

Деловой прецедент



Экземпляр делового прецедента - последовательность действий, выполняемых в деловой сфере, которая производит видимый результат, имеющий значение для индивидуального субъекта деловой сферы.

Деловой прецедент определяет набор экземпляров деловых прецедентов. Деловой прецедент имеет **имя**.

Процессы деловой сферы определяются как ряд различных деловых прецедентов, каждый из которых представляет определенный поток работ в деловой сфере. Деловой прецедент определяет то, что должно случиться в деловой сфере, когда он выполнен; он описывает эффект последовательности действий, которые производят результат, ценный для конкретного делового субъекта.

С точки зрения индивидуального субъекта деловой прецедент определяет законченный поток работ, который произвел нужные результаты. Это подобно тому, что называют «деловым процессом», но «деловой прецедент» имеет намного более строгое определение.

Совокупность деловых прецедентов представляет все возможные способы использования деловой сферы.

Связь ассоциации в модели деловых прецедентов

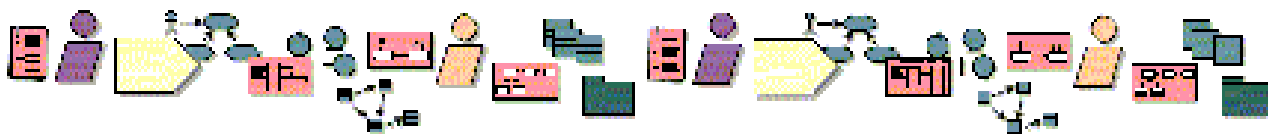


Связь ассоциации между прецедентом и субъектом указывает, что экземпляр прецедента и экземпляр субъекта взаимодействуют.

Деловые субъекты взаимодействуют с деловой сферой, посылая и принимая сообщения. Обе стороны могут брать инициативу взаимодействия на себя.

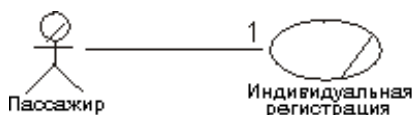
Чтобы полностью понять роль делового субъекта, Вы должны знать, в какие процессы включен этот субъект. Это обозначают связью ассоциации между деловым субъектом и деловым прецедентом, представляющим процесс. Связь ассоциации указывает на существование взаимодействия.

Множественность ассоциации показывает, сколько экземпляров делового прецедента могут одновременно взаимодействовать с одним экземпляром делового субъекта; множественность в обратном направлении показывает, сколько экземпляров делового субъекта могут взаимодействовать с одним экземпляром делового прецедента.



Пример:

Когда экземпляр делового субъекта Пассажир подходит к стойке регистрации пассажиров и предъявляет свои билет и багаж, он посылает сообщение экземпляру прецедента Индивидуальная регистрация. В конце процедуры регистрации деловой прецедент распечатает и выдаст пассажиру посадочный талон и одну или более багажных квитанций. Пассажир может связываться только с одним экземпляром прецедента Индивидуальная регистрация. Таким образом, множественность связи равна [1].



При регистрации в аэропорту Пассажир взаимодействует с прецедентом Индивидуальная регистрация.

Когда субъект и прецедент взаимодействуют, их связь может быть реализована различными средствами. Например, телефон, факс, почта и электронная почта. При этом может быть послано одно или несколько сообщений, но между ними имеется только одна связь ассоциации.

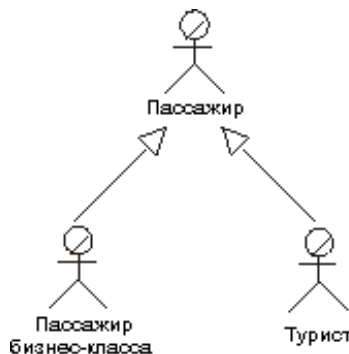
Обобщение субъектов в модели деловых прецедентов

Связь обобщения класса деловых субъектов (потомка) с другим классом деловых субъектов (предком) указывает, что потомок наследует роль, которую предок может играть в деловом прецеденте.

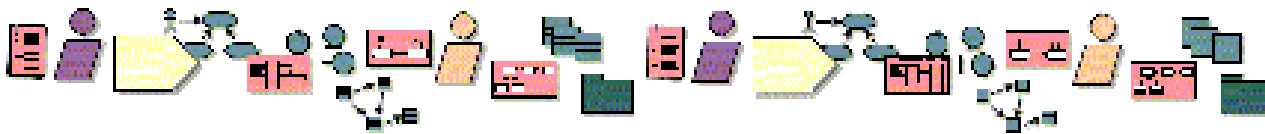
Несколько деловых субъектов могут играть одну и ту же роль в конкретном деловом прецеденте.

Пример:

Субъекты Пассажир бизнес-класса и Турист одинаково являются внешними объектами делового прецедента, который выполняет регистрацию. Их общая роль, смоделированная деловым субъектом Пассажир, унаследована двумя первоначальными деловыми субъектами. Мы отображаем эту ситуацию с использованием связей обобщения.



Субъекты Пассажир бизнес-класса и Турист наследуют все атрибуты Пассажира. Таким образом, оба эти субъекта могут действовать как пассажиры.



С другой стороны, один деловой пользователь может играть относительно деловой сферы несколько различных ролей, что означает, что пользователь может корреспондироваться с несколькими деловыми субъектами. В случае, когда имеется общее поведение некоторого вида пользователей, модель может стать более ясной, если пользователь представлен одним деловым субъектом, который наследует свойства нескольких деловых субъектов. Каждый унаследованный деловой субъект представляет одну из пользовательских ролей относительно деловой сферы.

Связь включения в модели деловых прецедентов

«включает»
----->

Связь включения – это связь от основного прецедента к включаемому прецеденту. Она определяет, что поведение, определенное для включаемого прецедента, явно вставлено в поведение, определенное для основного прецедента.

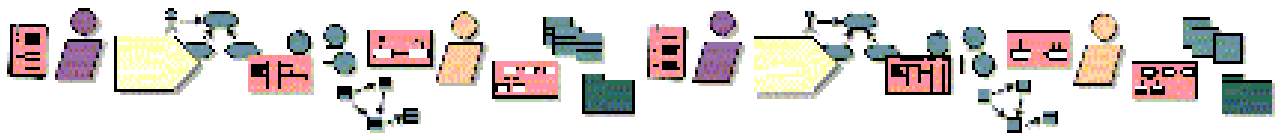
Связи включения используются, чтобы выделить из основного прецедента те части потока работ, которые влияют на основной прецедент только своим результатом, но не методом достижения этого результата. Вы можете выделить эти части работ в отдельный раздел, если это упростит понимание основного прецедента (детальное поведение «скрыто»), или, если выделенное в раздел поведение может многократно использоваться в других основных прецедентах.

Пример:

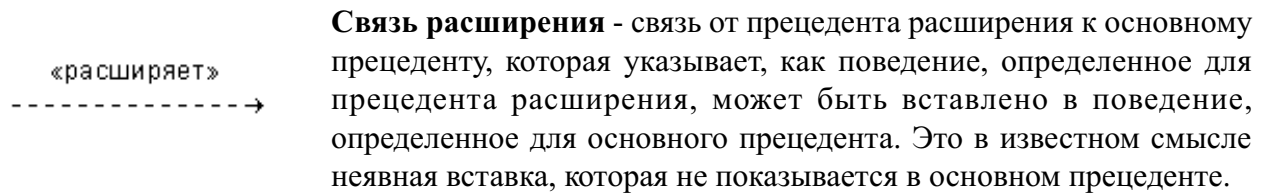


Оба деловых прецедента: Индивидуальная регистрация и Групповая регистрация, включают деловой прецедент Оформление и обработка багажа.

Экземпляр делового прецедента, который выполняет описание основного прецедента, будет выполнять также и описание включенного прецедента. Весь поток работ, описанный во включенном деловом прецеденте, включается в основной. Включаемый деловой прецедент такого вида - всегда абстрактный, и для него нет потребности в связи с деловым субъектом.



Связь расширения в модели деловых прецедентов



Когда Вы выделите поток работ делового прецедента, Вы можете увидеть в нем поведение, которое является условным или необязательным. Если эта часть поведения является существенной, Вы можете описать ее отдельно. Это можно сделать, выделив его в отдельный подраздел описания потока работ, или описать его в отдельном деловом прецеденте, который является расширением к первоначальному деловому прецеденту.

Второй подход особенно интересен, если извлеченная часть существенна, логически связана и естественно ограничена, и если Вы хотите сохранить первоначальный деловой прецедент простым. Или если одно и то же необязательное расширение применимо к нескольким деловым прецедентам.

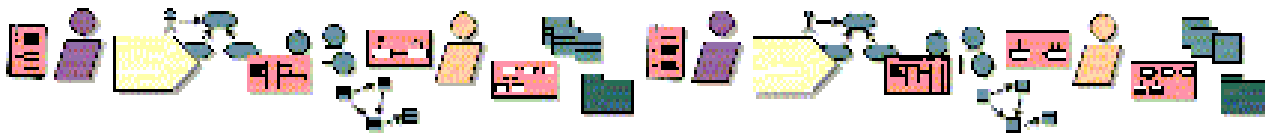
Экземпляр делового прецедента, который имеет необязательное расширение, сначала следует в соответствии с описанием основного прецедента и затем, если выполняется некоторое условие, изменяет направление, чтобы следовать описанию прецедента расширения. Когда прецедент расширения заканчивается, происходит возврат к описанию основного прецедента.

Пример:

Оформление и обработка специального багажа включается в индивидуальную регистрацию только в тех случаях, когда пассажир имеет такой багаж и должен перейти к специальной стойке оформления специального багажа.



Поток работ прецедента Оформление и обработка специального багажа вставлен в прецедент Индивидуальная регистрация с использованием связи расширения.

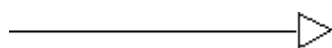


Деловые прецеденты расширения **должны быть** значимыми и самозавершенными, даже если поток работ добавляемого делового прецедента не выполняется. Большинство деловых прецедентов расширения не могут выполняться сами по себе.

Например, используйте связь расширения для дополнения делового прецедента в следующих случаях:

- Для моделирования условного выражения или необязательного поведения в деловом прецеденте, описывая потоки работ, где условное или необязательное поведение отличается от обязательного поведения, в других прецедентах.
- Для моделирования сложного потока работ, который происходит достаточно редко.
- Для моделирования отдельного подпотока, который выполняется только при известных условиях.
- Для моделирования нескольких различных деловых прецедентов, которые могут быть выполнены в некоторой точке в порядке, устанавливаемом деловым субъектом.

Обобщение прецедентов в модели деловых прецедентов



Обобщение прецедентов – это связь прецедента потомка с прецедентом предком, определяющая, что потомок может наследовать всё поведение и все характеристики, описанные для предка.

Обобщения прецедентов используются, чтобы показать общую часть структуры, целей и поведения потоков работ. Прецедент предок может быть специализирован в один или большее количество прецедентов потомков, которые представляют более частные формы предка. Это и есть обобщение в применении к прецедентам.

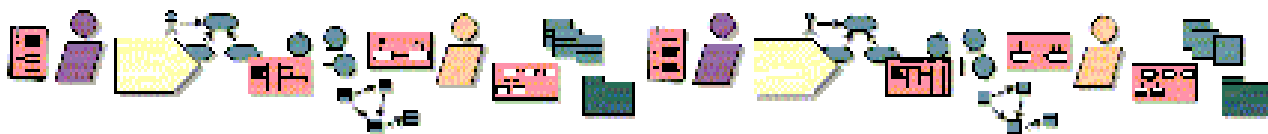
Когда Вы выделили поток работ каждого делового прецедента, Вы можете найти структуры и поведение, которые являются общими для нескольких деловых прецедентов. Чтобы избежать многократного описания одного и того же потока работ, Вы можете помещать общее поведение в отдельный деловой прецедент.

Модель деловых объектов



Модель деловых объектов – это объектная модель, описывающая реализацию деловых прецедентов. Она служит абстракцией того, как должны быть связаны деловые работники и деловые сущности, и как они должны сотрудничать в деловой сфере.

Модель деловых объектов описывает прецедент деловой сферы с внутренней точки зрения деловых работников. Модель определяет, **как** люди, которые работают в деловой сфере, и вещи, которые они обрабатывают и используют (классы и объекты деловой сферы), должны быть



связаны друг с другом, статически и динамически, чтобы произвести ожидаемые результаты. Все вместе объекты классов модели должны быть способны выполнить все прецеденты деловой сферы.

Деловой работник



Деловой работник представляет роль или набор ролей в деловой сфере. Деловой работник взаимодействует с другими работниками и управляет деловыми сущностями, участвуя в реализации делового прецедента.

Деловой работник представляет абстракцию человека, который действует в пределах деловой сферы. Объект - деловой работник взаимодействует с другими объектами - деловыми работниками и управляет деловыми объектами - сущностями для реализации делового прецедента.

Работник подвергается обработке («пилотируется»), когда начинается поток работ соответствующего экземпляра прецедента или, самое позднее, с того времени, когда человек дал задание начать выполнение его роли в экземпляре прецедента. Объект - работник «живет» (человек занят) пока выполняется прецедент.

Объекту обычно соответствует некоторая информация, которая описывает его характеристики. Эта информация может быть смоделирована как **атрибуты** класса.

Операция класса делового работника представляет определенное действие, которое будет выполнено индивидуумом этого класса. Операция делового работника инициализируется **сообщением** от другого индивидуума или от субъекта. Операция имеет **название** и, может быть, **параметры**.

Деловая сущность

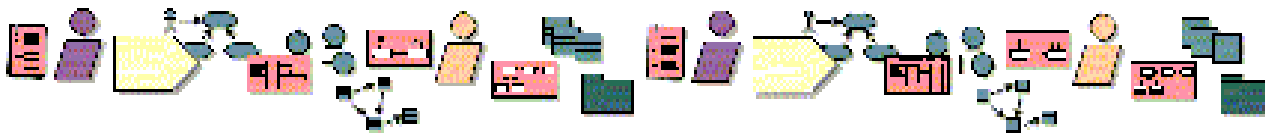


Деловая сущность представляет «вещь», которую обрабатывает или использует деловой работник.

Деловые сущности представляют «вещи», обрабатываемые или используемые деловыми работниками при выполнении делового прецедента. Деловая сущность часто представляет что-то ценное для нескольких деловых прецедентов или экземпляров прецедентов, так что объекты деловой сущности могут быть довольно долгоживущими. Вообще, считается хорошей практикой, если деловая сущность не содержит никакой информации о том, как и для кого она используется.

Как правило, деловая сущность представляет документ или существенную часть изделия. Иногда она представляет что-то менее материальное, вроде важной информации относительно рынка или заказчика. Примеры деловых сущностей в ресторане - Меню и Напиток; в аэропорту – Билет и Посадочный талон.

Вы должны моделировать как деловые сущности только те явления, к которым должны



обращаться другие классы в модели деловых объектов. Другие «вещи» могут быть смоделированы как **атрибуты** соответствующих классов или только описаны в этих классах в виде текста.

Теоретически любое явление может быть смоделировано как класс. Однако, использование атрибутов может сократить число поддерживаемых классов и сделать объектную модель более простой для понимания.

Операция определяет инструмент, которым управляется деловая сущность. **Операция** имеет **название** и, может быть, **параметры**. Доступ к объекту деловой сущности показывается как **сообщение**, посылаемое объекту деловой сущности.

Связь ассоциации в модели деловых объектов

_____ **Ассоциация** от класса А к классу В в модели деловых объектов указывает, что А или объекты А содержат ссылку на В или объекты В. Ассоциация имеет **название** и **множественность**. Множественность указывает, сколько объектов связанного класса могут быть связаны с объектом исходного класса.

Ассоциация представляет структурные связи между экземплярами деловых работников и деловых сущностей в модели деловых объектов. Это информация, которая должна сохраняться некоторое время, и она не отражает связей процедурной зависимости. Каждая ассоциация имеет название и множественность. Множественность указывает, сколько объектов связанного класса могут быть связаны с объектом исходного класса. Она указывается константой или диапазоном (например, 0 .. 5), который показывает число объектов, которые могут быть связаны.

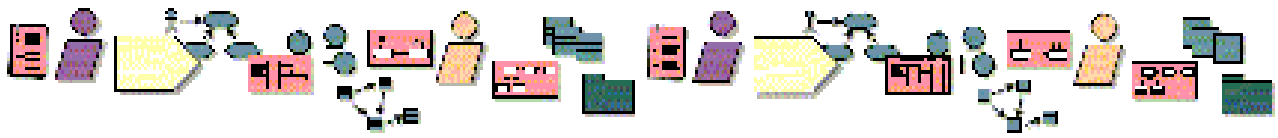
Пример:

Агент, который регистрирует пассажиров авиакомпании, выполняет набор инструкций, которые описывают его действия в деловом прецеденте регистрации. Каждый служащий, действующий как агент по регистрации, должен знать эти процедуры, чтобы прецедент регистрации работал гладко. Класс делового работника Агент регистрации должен иметь связь ассоциации с классом деловой сущности, представляющей набор инструкций.

Некоторые типы ассоциаций могут иметь довольно широкое толкование. Для этих типов важно, чтобы Вы определили специфическую интерпретацию в каждом случае. Для этой цели может быть определена роль, которую играет сторона ассоциации. Если этого не достаточно для описания ассоциации, ей можно также давать название.

Связь объединения в модели деловых объектов

◊ _____ Связь **объединения** от класса А к классу В указывает, что объект класса А составлен из объектов класса В. Объединение – специфическая форма ассоциации.

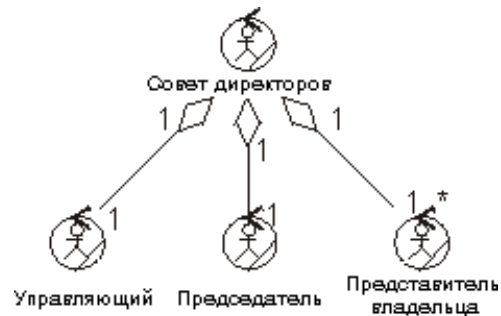


Иногда группа людей действует в прецеденте как единое целое, или, в более общем виде, явление составляется из других независимых явлений. Например, школьный класс состоит из учащихся. Такое составное явление называется агрегатом (объединением).

Объединение моделируется с использованием отдельного класса для составного явления. Этот класс имеет связи объединения с классами, которые представляют его составляющие. Такая конструкция дает возможность как обращаться к составляющим как частям целого, так и обрабатывать их как самостоятельные вещи. Класс-объединение не обязательно должен иметь много своих собственных свойств. Его характеристикой может быть то, что он объединяет различные компоненты.

Пример:

Совет директоров компании состоит из председателя, управляющего высшего ранга и нескольких представителей владельца.



Класс-объединение включает другие классы.

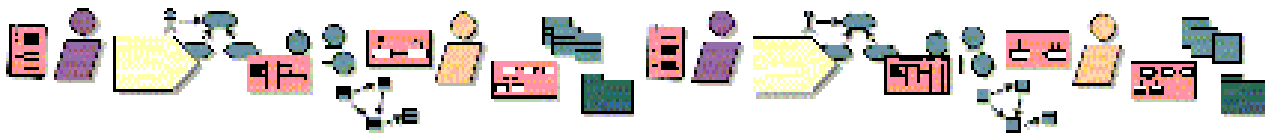
Вы можете использовать объединения только в случаях необходимости; то есть тогда, когда предполагается, что и объединение, и любая из его непосредственных составляющих действуют или будут полезны сами по себе. Хорошее объединение - естественная, ясная часть модели деловых объектов - ее назначение должно быть легко понятно из контекста.

Связь объединения можно применять только к классам, представляющим один и тот же вид явления. Например, не имеет смысла для деловой сущности быть агрегатом деловых работников.

Связь обобщения в модели деловых объектов

Связь обобщения от одного класса (потомка) к другому классу (предку) указывает, что объект класса-потомка наследует все атрибуты, операции и связи, которые принадлежат классу предку.

Многие вещи в реальной жизни имеют общие свойства. Например, и собаки и кошки - животные. Классы также могут иметь общие свойства. Отношения этого типа между классами могут быть представлены посредством связи **обобщения**. Извлекая общие свойства в отдельный класс можно существенно упростить деловую модель, чтобы легче изменять ее в будущем.



Класс, который наследует общие характеристики от другого класса, называется потомком. Класс, свойства которого унаследовал потомок, называется предком. Связь обобщения показывает, как классы наследуются один от другого. Она означает, что определение предка, включая все атрибуты и операции, является также правильным и для потомка. Наследуются также и связи предка.

Обобщение может иметь место в нескольких этапах, что дает возможность моделировать сложную, многоуровневую иерархию наследований, хотя для простоты понимания число уровней желательно ограничить. Общие свойства помещаются в верхней части иерархии наследований, а специализация усиливается вниз по иерархии. Другими словами, связи обобщения могут использоваться для моделирования большого количества специализаций обобщённого понятия.

Пример:

Пассажиры, проходящие регистрацию в аэропорту, имеют различные виды багажа: обычный багаж, ручную кладь и специальный багаж. С точки зрения авиакомпании, все эти виды багажа имеют несколько общих свойств. Кроме того, что все они являются багажом, каждый из них имеет, например, владельца и вес. Эти общие свойства могут быть смоделированы атрибутами и операциями в отдельном классе под названием Багаж. Обычный багаж, Ручная кладь и Специальный багаж наследуют свойства этого класса.

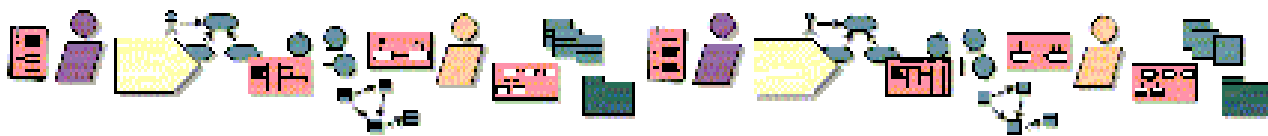


Классы Обычный багаж, Ручная кладь и Специальный багаж имеют общие свойства. Все они являются специализациями обобщённого понятия Багаж.

Класс может наследовать свойства от нескольких других классов. Это называется «множественным наследованием», хотя обычно встречается наследование только от одного класса. Если класс наследует от нескольких классов, важно проверить, как называются в предках ассоциации, атрибуты и операции. Если то же самое название появляется в нескольких предках, Вы должны описать, что оно означает с определенным классом наследования.

Класс, который существует только как предок других классов, называется абстрактным классом. Абстрактный класс никогда не обрабатывается. Однако, объект класса-наследника абстрактного класса соответствует собственному описанию и описанию класса-предка. Классы, которые подвергаются обработке в деловой сфере – конкретные классы.

Главная цель использования наследования состоит в том, чтобы достичь объектной модели, которая приспособлена к изменению. Однако, наследование должно использоваться очень



тщательно:

- Наследование – это только способ структурирования описания. Вы показываете, какие из явлений имеют некоторые общие свойства.
- Когда дело доходит до реализации, Вы должны найти способ выполнения как обработки предка, так и его потомка всякий раз, когда должен быть обработан класс-потомок.
- Обобщения используются только между классами одного и того же стереотипа.

Поскольку различные стереотипы классов имеют различные цели, связь обобщения от класса одного стереотипа к классу другого стереотипа не имеет смысла. Например, если бы Вы позволили классу делового работника наследовать свойства деловой сущности, деловой работник стал бы удивительным гибридом.

Организационный модуль



Организационный модуль включает деловых работников, деловые сущности и другие организационные модули, которые объединяются в соответствии с некоторым критерием. Организационный модуль имеет **название**.

Как правило, деловую сферу структурируют, организуя служащих в департаменты, отделы, группы и так далее, согласно различным критериям. Эта ситуация моделируется распределением деловых работников и деловых сущностей по организационным модулям.

Организационный модуль - конструкция для структурирования деловой сферы путем деления ее на меньшие части. Организационные модули могут содержать не только деловых работников и деловые сущности, но также и другие организационные модули, что позволяет Вам формировать иерархию организационных модулей. Деловой работник или деловая сущность могут быть непосредственными элементами только одного организационного модуля.

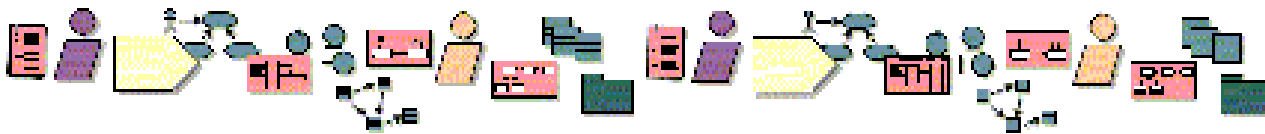
Реализация делового прецедента



Реализация делового прецедента описывает, как реализован конкретный деловой прецедент в пределах модели деловых объектов, в терминах сотрудничества объектов.

Модель деловых прецедентов описывает деловую сферу в терминах деловых субъектов и деловых прецедентов, соответствующих заказчикам и деловым процессам. Модель деловых прецедентов содержит описания потока работ, которые идентифицируют то, **что** делается. То, **как** выполняется работа в каждом деловом прецеденте, описывается в модели деловых объектов.

Набор работников, которые выполняют работу делового прецедента, и деловых объектов, к которым они обращаются и которыми управляют в процессе работы, называется реализацией делового прецедента. Объекты одного и того же класса могут участвовать в нескольких различных



реализациях деловых прецедентов, отражая тот факт, что один и тот же вид ресурса одновременно участвует в различных процессах.

Диаграмма прецедентов



Диаграмма прецедентов показывает деловые субъекты, деловые прецеденты, пакеты деловых прецедентов и связи между ними.

Никаких строгих правил относительно того, что нужно показывать в диаграммах прецедентов, не существует. Вы показываете те связи в модели, которые считаете интересными. Могут быть полезны следующие диаграммы:

- Деловые субъекты, принадлежащие к одному и тому же пакету прецедентов.
- Деловой субъект и все деловые прецеденты, с которыми он взаимодействует. Диаграмма этого типа может функционировать как локальная диаграмма делового субъекта и, вероятно, должна быть с ним связана.
- Деловые прецеденты, обрабатывающие одну и ту же информацию.
- Деловые прецеденты, используемые одной и той же группой субъектов.
- Деловые прецеденты, которые часто выполняются в одной последовательности.
- Деловые прецеденты, принадлежащие одному пакету прецедентов.
- Наиболее важные деловые прецеденты. Диаграмма этого типа может функционировать как резюме модели.
- Определенный деловой прецедент и его связи с деловыми субъектами и другими деловыми прецедентами. Диаграмма этого типа может функционировать как локальная диаграмма делового прецедента и, вероятно, должна быть с ним связана.

Рекомендуется, чтобы каждый деловой субъект, деловой прецедент и связь были включены по крайней мере в одну из диаграмм. Если это делает модель деловых прецедентов более ясной, они могут включаться в несколько диаграмм, или Вы можете показывать их несколько раз на одной и той же диаграмме.

Диаграмма действий

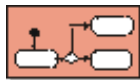
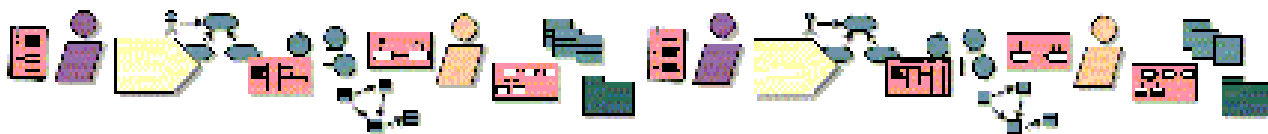


Диаграмма действий в модели деловых прецедентов иллюстрирует поток работ делового прецедента.

Поток работ делового прецедента описывает последовательность действий, которые вместе что-то производят для делового субъекта. Поток работ часто состоит из основного потока и одного или нескольких альтернативных потоков.



Поток работ делового прецедента может быть описан графически с помощью диаграммы действий. Такие диаграммы показывают:

- **Состояния действия**, которые представляют выполнение действия или шага в пределах потока работ.
- **Переходы** между этими состояниями действия.
- **Решения**, для которых определен набор **условий защиты**. Эти условия защиты управляют тем, какой переход должен быть выполнен, когда состояние действия завершено.

Пример диаграммы действий представлен на иллюстрации ниже.

Примечание: Rational Rose 98i не поддерживает работу с диаграммами действий.

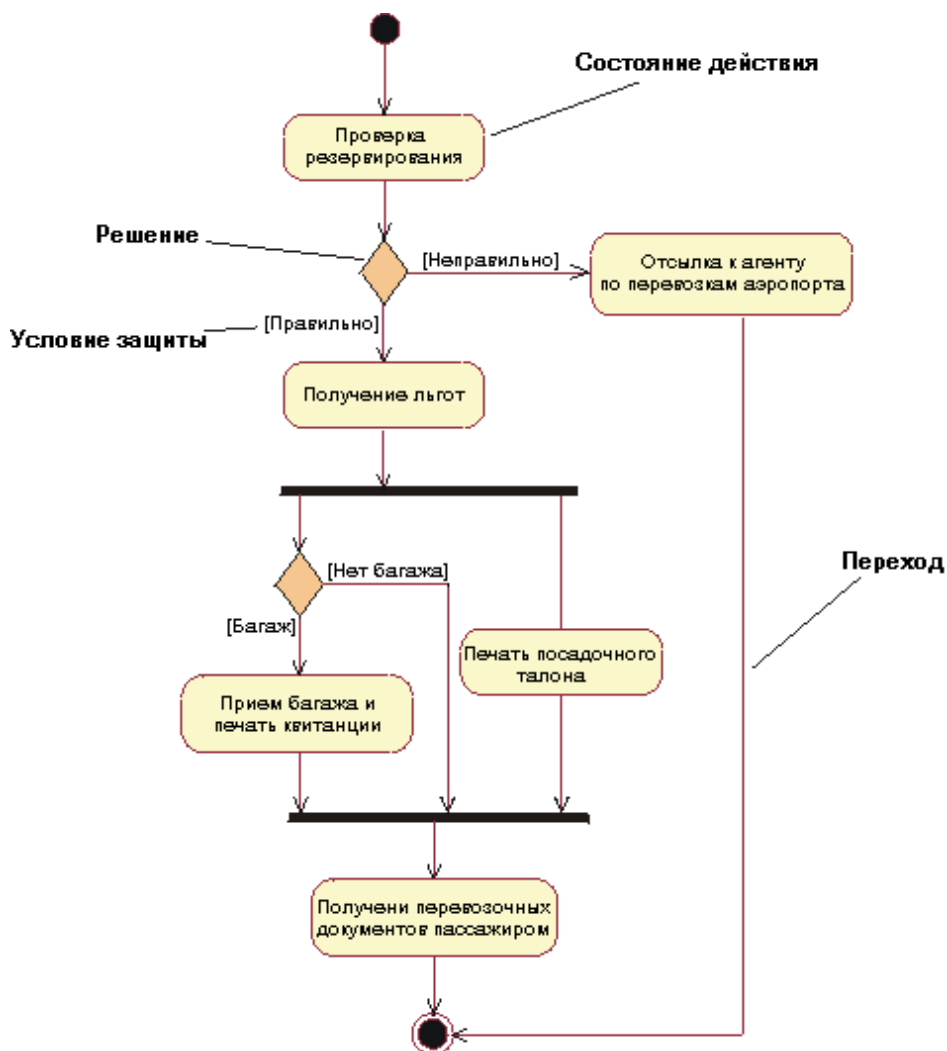


Диаграмма действий для делового прецедента Индивидуальная регистрация в модели деловых прецедентов регистрации в аэропорту.

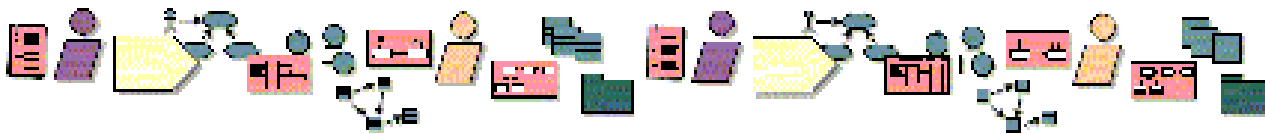


Диаграмма классов

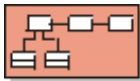


Диаграмма классов показывает совокупность декларативных (статических) элементов модели, таких как классы и пакеты, а также их содержание и связи.

Диаграммы классов показывают связи ассоциации, объединения и обобщения между деловыми работниками и деловыми сущностями. Могут представлять интерес следующие виды диаграмм классов:

- Иерархии наследований.
- Объединения деловых работников и деловых сущностей.
- Как деловые работники и сущности связаны посредством ассоциаций.

Диаграмма сотрудничества



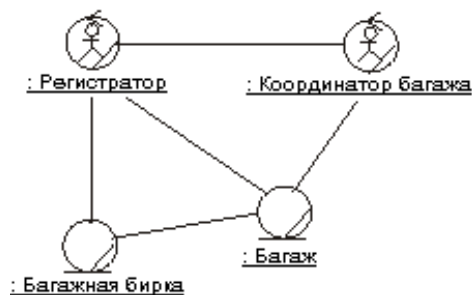
Диаграмма сотрудничества описывает экземпляр взаимодействия между объектами; она показывает объекты, участвующие во взаимодействии, с их связями друг с другом и с сообщениями, которые они посылают друг другу.

Различают групповые и рабочие диаграммы сотрудничества.

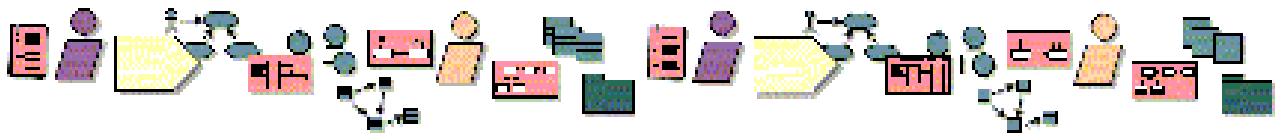
Групповая диаграмма сотрудничества содержит деловых работников, деловые сущности и связи между ними. Если это сделает диаграмму более информативной, Вы можете также включать в диаграмму деловых субъектов, которые связаны с деловыми работниками.

В групповую диаграмму сотрудничества не включают сообщения и порядковые номера сообщений. Вы не можете показать, как передается между работниками управление поведением. Часто, но не всегда, это достаточный уровень детализации для определения внутреннего поведения делового прецедента. Если уровень детализации не достаточен, разработайте диаграмму последовательности или рабочую диаграмму сотрудничества критической части.

Пример:



Групповая диаграмма сотрудничества для прецедента Индивидуальная регистрация.



Рабочая диаграмма сотрудничества семантически идентична диаграмме последовательности, но сосредоточена на объектах, в то время как последняя фокусируется на взаимодействиях. В отличие от групповой диаграммы сотрудничества, рабочая диаграмма сотрудничества должна представлять подмножество объектов в соответствии с рабочей последовательностью взаимодействий, включая их связи, сообщения и последовательности сообщений.

Диаграмма последовательности



Диаграмма последовательности описывает экземпляр взаимодействий между объектами, размещая их в хронологическом порядке; она показывает «линии жизни» объектов, участвующих во взаимодействии, и сообщения, которые они посылают друг другу.

Диаграмма последовательности графически изображает подробности взаимодействия деловых работников и деловых субъектов и то, как обращаются к деловым сущностям в ходе выполнения делового прецедента. Диаграмма последовательности кратко описывает то, что делают участвующие деловые работники, как управляются деловые сущности (в терминах запусков), и как они связываются, посылая сообщения друг другу.

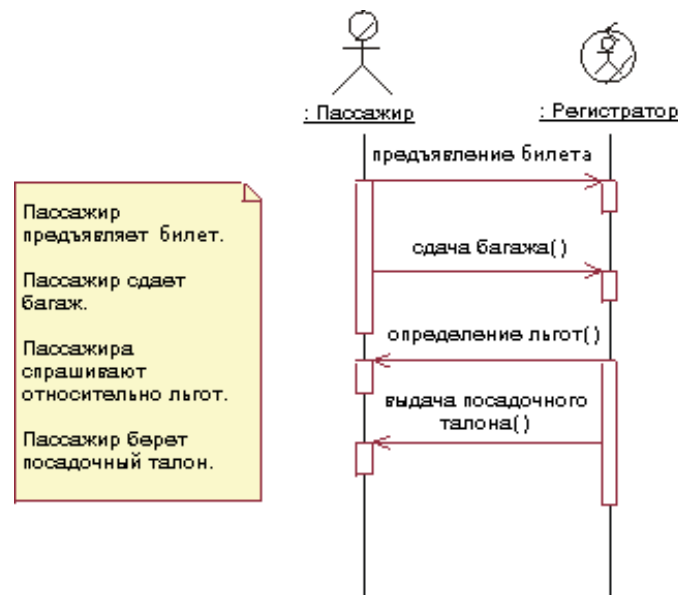
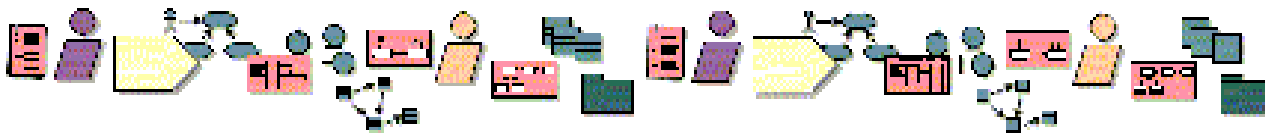


Диаграмма последовательности части делового прецедента Индивидуальная регистрация.

Та же самая информация, которая представлена в диаграмме последовательности, может быть представлена вместо этого в рабочей диаграмме сотрудничества.



Диаграммы состояний

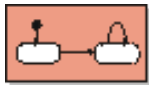


Диаграмма состояний представляет машину состояний, поведение которой определяется последовательностью состояний, проходимых объектом в течение его жизни в ответ на события, вместе с его ответами и действиями.

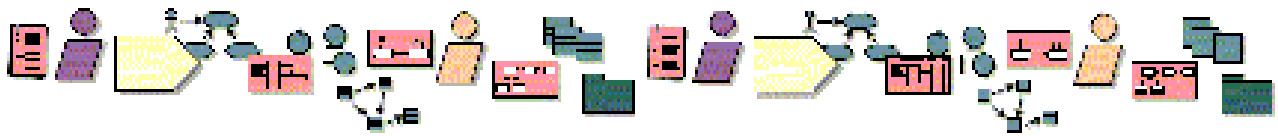
Диаграмму состояний можно использовать для иллюстрации состояний, которые может иметь деловой работник или деловая сущность, событий, которые вызывают переход из одного состояния в другое, и действий, которые вытекают из изменения состояния. Диаграмма состояний часто упрощает проверку правильности конструкции класса.

Для каждого состояния, которое может иметь объект класса, диаграмма показывает сообщения, которые он может принимать, действия, которые будут выполнены и состояние, в которое перейдет объект класса после этого.

Пример:



Диаграмма состояний деловой сущности Багаж.



Краткий обзор действий

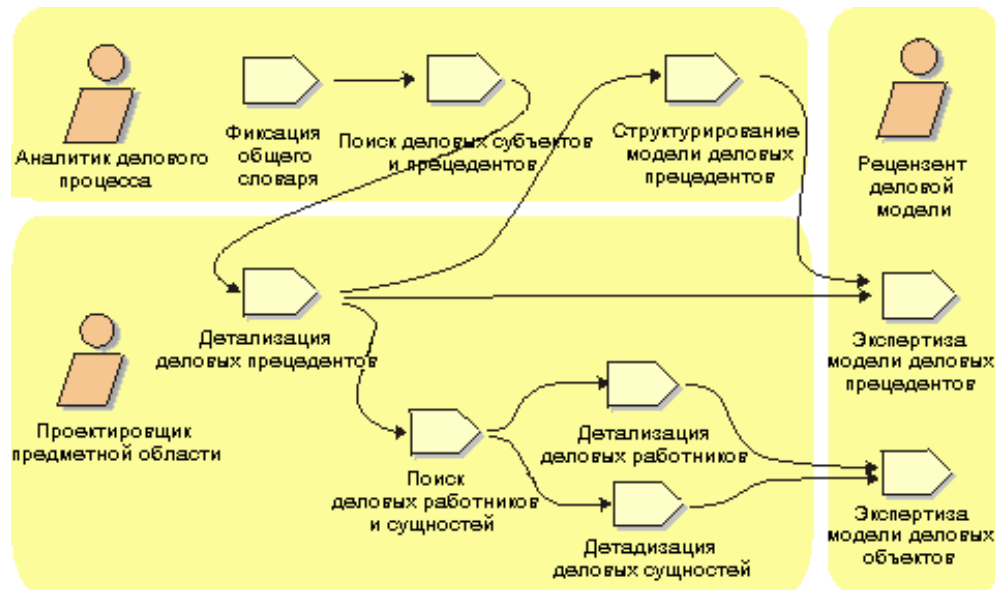


Диаграмма краткого обзора действий потока работ показывает все действия делового моделирования

Работник: Аналитик делового процесса

Аналитик делового процесса ведет и координирует моделирование деловых прецедентов, выделяя и определяя границы моделирования организации. Например, он устанавливает, какие деловые субъекты и деловые прецеденты существуют и как они взаимодействуют.

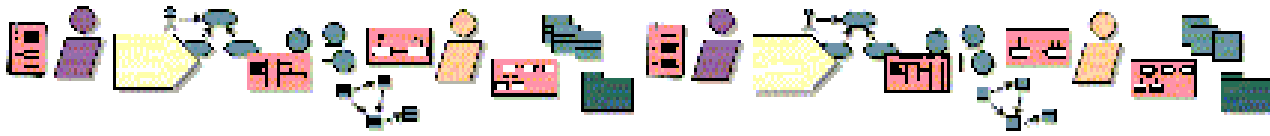
Действие: Фиксация общего словаря

В процессе делового моделирования Вы должны определить общий словарь, используя наиболее общие термины и выражения из прикладной области. Впоследствии Вы должны последовательно использовать общий словарь во всех текстовых описаниях предметной области. Таким образом, Вы сохраните текстовые описания непротиворечивыми и избежите недоразумений среди участников проекта относительно использования и значения терминов. Вы должны документировать словарь в глоссарии.

Действие «Фиксация общего словаря» включает следующие шаги:

- Поиск общих терминов
- Оценка результатов

Инструментальная поддержка в Rational Unified Process - Rational RequisitePro.



Действие: Поиск деловых субъектов и прецедентов

Целью этого действия является определение границ моделируемой предметной области.

Действие «Поиск деловых субъектов и прецедентов» включает следующие шаги:

- Поиск деловых субъектов
- Поиск деловых прецедентов
- Расстановка деловых прецедентов по приоритетам
- Разработка схемы потока работ делового прецедента
- Описание взаимодействия деловых субъектов с прецедентами
- Оформление деловых прецедентов и субъектов
- Представление модели деловых прецедентов в диаграммах прецедентов
- Разработка отчета об исследовании модели деловых прецедентов
- Оценка результатов

Инструментальная поддержка в Rational Unified Process - Rational Rose и Rational RequisitePro.

Действие: Структурирование модели деловых прецедентов

Целью этого действия являются: извлечение из деловых прецедентов поведения, которое нужно рассматривать как абстрактные деловые прецеденты; нахождение новых абстрактных деловых субъектов, которые определяют роли, являющиеся общими для нескольких деловых субъектов.

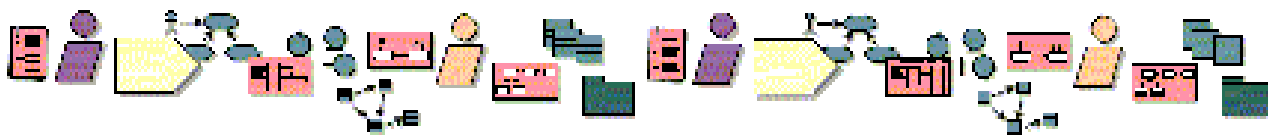
Действие «Структурирование модели деловых прецедентов» включает следующие шаги:

- Поиск связей включения между деловыми прецедентами
- Поиск связей расширения между деловыми прецедентами
- Поиск связей обобщения между деловыми прецедентами
- Поиск связей обобщения между деловыми субъектами
- Оценка результатов

Инструментальная поддержка в Rational Unified Process - Rational Rose.

Работник: Проектировщик предметной области

Проектировщик предметной области детализирует часть технических требований организации, описывая поток работ одного или нескольких деловых прецедентов. Он определяет деловых работников и деловые сущности, необходимые для реализации делового прецедента и распределяет между ними его выполнение. Проектировщик предметной области определяет



ответственности, действия, атрибуты и связи одного или нескольких деловых работников и деловых сущностей.

Действие: Детализация делового прецедента

Целью этого действия является подробное описание потока работ делового прецедента. Сделать это нужно так, чтобы его могли понять заказчики, пользователи и совладельцы.

Действие «Детализация делового прецедента» включает следующие шаги:

- Сбор информации о деловом прецеденте
- Детализация потока работ делового прецедента
- Структурирование потока работ делового прецедента
- Иллюстрирование связей с деловыми субъектами и другими деловыми прецедентами
- Описание специальных требований делового прецедента
- Описание целей выполнения делового прецедента (необязательно)
- Оценка результатов

Инструментальная поддержка в Rational Unified Process - Rational Rose.

Действие: Поиск деловых работников и сущностей

Целями этого действия являются: идентификация всех «ролей» и «вещей» в деловой сфере; описание того, как выполняются деловые прецеденты деловыми работниками и деловыми сущностями.

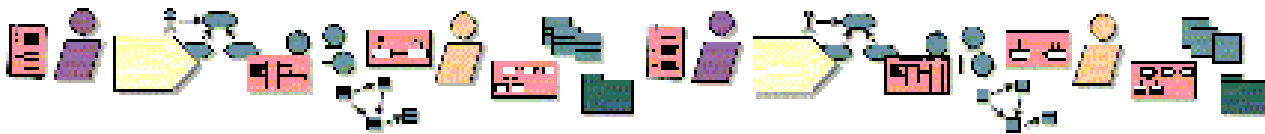
Действие «Поиск деловых работников и сущностей» включает следующие шаги:

- Идентификация организационных модулей
- Идентификация деловых работников
- Идентификация деловых сущностей
- Определение реализаций делового прецедента
- Структурирование модели деловых объектов
- Оценка результатов

Инструментальная поддержка в Rational Unified Process - Rational Rose.

Действие: Детализация делового работника

Целью этого действия является подробное описание обязанностей делового работника.



Действие «Детализация делового работника» включает следующие шаги:

- Выделение области ответственности
- Определение операций
- Определение атрибутов
- Описание требований к компетентности
- Оценка результатов

Инструментальная поддержка в Rational Unified Process - Rational Rose.

Действие: Детализация деловой сущности

Целью этого действия является детализация определения деловой сущности.

Действие «Детализация деловой сущности» включает следующие шаги:

- Выделение области ответственности
- Определение операций
- Определение атрибутов
- Оценка результатов

Инструментальная поддержка в Rational Unified Process - Rational Rose.

Работник: Рецензент деловой модели

Рецензент деловой модели участвует в формальных обзорах модели деловых прецедентов и модели деловых объектов.

Человек, действующий как рецензент деловой модели, должен иметь некоторый объем знаний деловой области или технологии, которую предполагается использовать для автоматизации деловой сферы. Другой навык, которым должен обладать рецензент деловой модели - детальное знание методики делового моделирования приложений.

Действия: Экспертиза моделей деловых прецедентов и деловых объектов

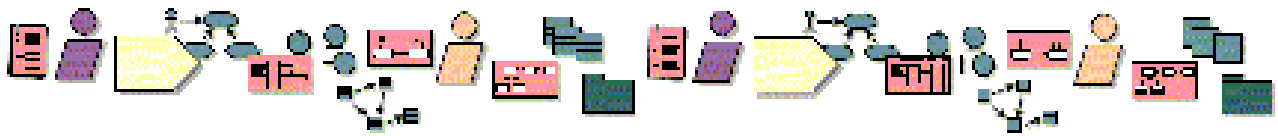
Целью этих действия является формальная проверка того, что результаты моделирования деловых прецедентов и деловых объектов соответствуют взглядам совладельцев деловой сферы.

Описания этих действий в интерактивной версии Rational Unified Process содержат следующую информацию:

- Рекомендации по порядку проведения экспертизы
- Контрольные точки для модели деловых прецедентов

- Контрольные точки для деловых субъектов
- Контрольные точки для деловых прецедентов
- Контрольные точки для отчетов о деловых прецедентах
- Контрольные точки для модели деловых объектов
- Контрольные точки для организационных модулей
- Контрольные точки для деловых работников
- Контрольные точки для деловых сущностей
- Контрольные точки для реализаций деловых прецедентов
- Контрольные точки для дополнительных деловых спецификаций
- Контрольные точки для глоссария





Поток работ «Деловое моделирование» (продолжение)

В предыдущей главе мы начали обсуждать поток работ «Деловое моделирование».

Вы познакомились с целями и основными понятиями делового моделирования. Затем, в весьма формальном изложении, Вы познакомились с кратким обзором действий моделирования.

Но где же тот замечательный практический опыт, который пообещал нам Rational Unified Process?

Этот опыт изложен в интерактивной версии Rational Unified Process, в разделах «*Руководящие указания*», «*Практические рекомендации*», «*Инструментальная поддержка*» и др. Я не беру на себя смелость заменить своим «Введением» интерактивную версию Rational Unified Process. Поэтому я решил попробовать изложить крупными буквами этот опыт на примерах.

Примечание: В дальнейшем материалы, относящиеся к примеру моделирования, выделены серым фоном.

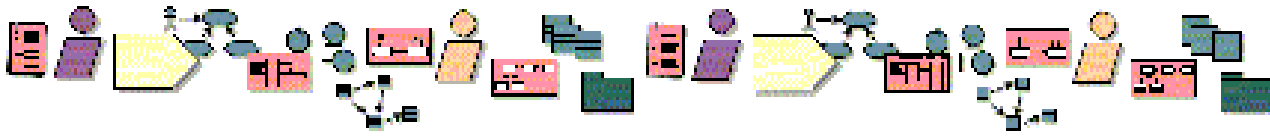
Пример моделирования: начало

Вымышленная компания Classics, Ink. выросла из розничного магазина, торговавшего компакт-дисками классической музыки. Компания внедрила размещение заказов по телефону и расширила ассортимент товаров. В планах руководства компании – электронная торговля (прием заказов и продажи с использованием Интернет).

Classics, Ink. переросла имеющиеся в компании средства автоматизации: контрольно-кассовый аппарат, системы финансового учета и управления запасами «из коробок». Новые потребности возникли и в связи с внедрением электронной торговли. Руководство компании хочет создать (или купить) новую систему обработки и выполнения заказов (Order Processing and Fulfillment System - OPFS).

Руководство Classics, Inc. своевременно поняло, что при разработке системы нужно рассмотреть всю деятельность компании, а не только обработку заказов, т.е. нужно формировать систему, которая будет поддерживать весь бизнес. Следующим естественным шагом явилось создание деловой модели предприятия.

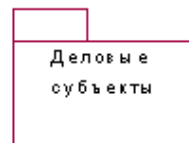
Нам очень повезло! Специалисты компании решили использовать для



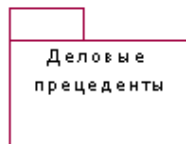
моделирования методы и инструментальные средства Rational: Rational Unified Process и Rational Rose. Теперь мы можем ознакомиться с некоторыми результатами их работы. Мы не будем полностью повторять проделанный ими путь. Но мы рассмотрим некоторые их решения, и постараемся понять, почему они сделали так, а не иначе.

Вот как выглядит деловая модель Classics, Inc. на самом верхнем уровне абстракции.

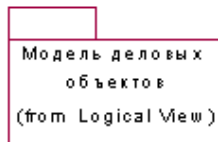
Деловое моделирование



Деловой субъект представляет роль, которую кто-то или что-то играет по отношению к предприятию в деловой среде.



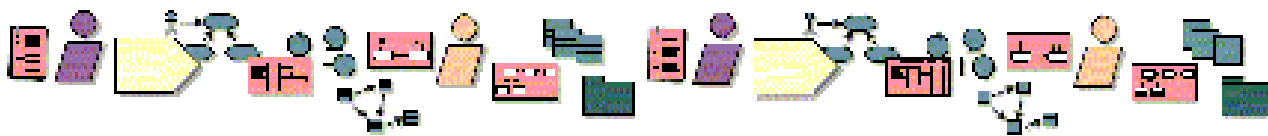
Модель деловых прецедентов - это модель назначенных предприятию функций. Модель деловых прецедентов используется как существенная исходная информация для идентификации ролей и объектов в организации.



Модель деловых объектов - это объектная модель, описывающая реализацию деловых прецедентов. Она служит абстракцией того, как должны быть связаны деловые работники и деловые сущности и как они должны сотрудничать при выполнении прецедента.

Деловая модель Classics, Inc.

Деловая модель Classics, Inc. содержит два представления компании: внешнее – модель деловых прецедентов (Use Case View) и внутреннее – модель деловых объектов (Logical View). Модель деловых прецедентов они представили в виде двух пакетов: пакет Деловые субъекты и пакет Деловые прецеденты. Постараемся понять, зачем им это понадобилось.



Модель деловых прецедентов

Поиск деловых субъектов

*Послушайте!
Ведь, если звезды зажигают –
Значит – это кому-нибудь нужно?*

В. Маяковский, 1914

Чтобы полностью понять цель любого предприятия, прежде всего нужно определить «кому это нужно», т.е. **кто** взаимодействует с предприятием, кто размещает запросы и/или заинтересован в получении ответов. Различные типы «взаимодействующих субъектов» представляются как **деловые субъекты**.

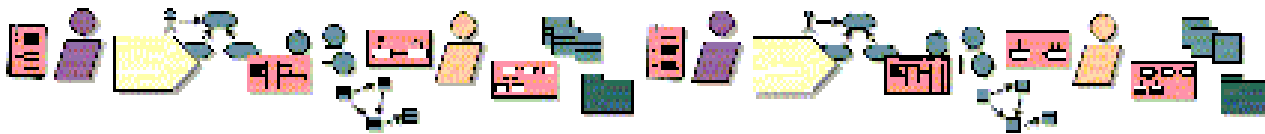
Термин «субъект» означает роль, которую кто-то или что-то играет при взаимодействии с предприятием. Ниже перечислены потенциальные деловые субъекты:

- Акционеры
- Заказчики
- Потенциальные заказчики («рынок»)
- Поставщики
- Партнеры
- Налоговые органы
- Коллеги из не моделированных частей предприятия.

Деловые субъекты определяют внешнюю границу делового моделирования.

Обратите внимание на последнюю строчку в списке потенциальных субъектов. Не всегда требуется моделировать деятельность всего предприятия. Тогда на внешней границе окажется работник из не моделируемой части предприятия.

Мы уже обсуждали, что чаще всего субъект соответствует человеку. Однако, бывают ситуации, когда роль субъекта играет, например, информационная система. Если Вы вполне довольны функционирующей на предприятии системой финансового учета, Вы можете ограничиться моделированием взаимодействия с этой системой, т.е. рассматривать ее как внешнюю границу моделирования. Если Вы можете управлять большинством операций банка с персонального компьютера в вашем помещении, ваш прецедент, взаимодействующий с субъектом «поставщик денег» (банк), фактически взаимодействует с информационной системой.



Семинар по деловым субъектам

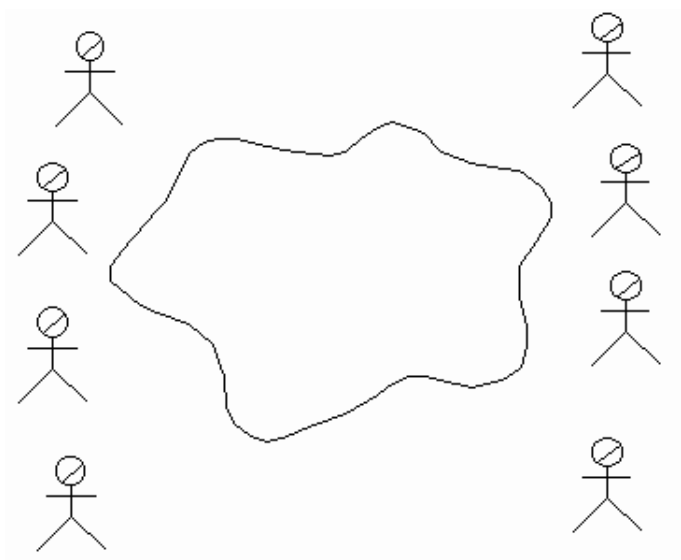
Rational Unified Process рекомендует использовать для поиска деловых субъектов и прецедентов метод мозговой атаки.

В семинаре должны принять участие представители пользователей и представители группы разработки. Желательно, чтобы их суммарный опыт охватывал следующие области:

- Потребности совладельцев
- Проект системы
- Rational Unified Process

В центре группы – ведущий. Ведущий является катализатором для всех идей и пожеланий.

На середине грифельной доски нарисуйте «облако» (моделируемое предприятие), а затем задайте вопрос: «Какие роли играют люди, организации или системы окружающей среды, которые взаимодействуют с предприятием?» Нарисуйте справа или слева от облака-предприятия изображение человечка для каждой предложенной роли и запишите под человечком название роли. Теперь на доске два столбика субъектов – по одному по каждую сторону облака.

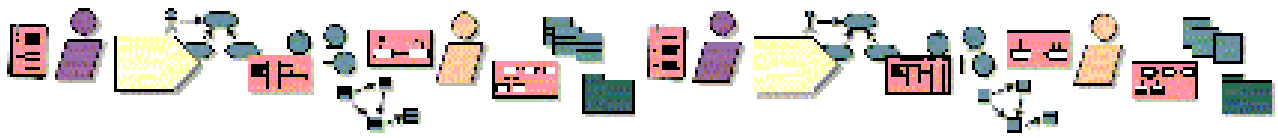


Вот что, приблизительно, нарисовано на доске. Каждый человечек имеет имя - название субъекта

Будьте готовы к обнаружению множества субъектов. Для начала это хорошо; позже Вы удалите лишние.

Вопросы, требующие ответа:

- Кто извлекает пользу от существования предприятия?



Иногда полезно использовать слова «роль» или «пользователь» вместо слова «субъект».

Вам могут задать вопрос, подобный следующему: «Почему газета «Экстра Балт» не субъект? Ведь она всегда публикует нашу рекламу!» Вы должны будете задать много вопросов, чтобы понять роль этой газеты.

- Какова роль газеты «Экстра Балт»?
- Кто еще, кроме газеты «Экстра Балт», может выполнять эту роль?
- Почему вы размещаете рекламу именно в этой газете?

В результате обсуждения, скорее всего, окажется, что реклама может размещаться и в других газетах, на радио, в вагонах метро. Взаимодействия со всеми этими «рекламными агентами» выполняются примерно по одинаковым правилам. И, что газета «Экстра Балт» – это экземпляр делового субъекта Рекламный агент.

Название субъекта должно отражать роль.

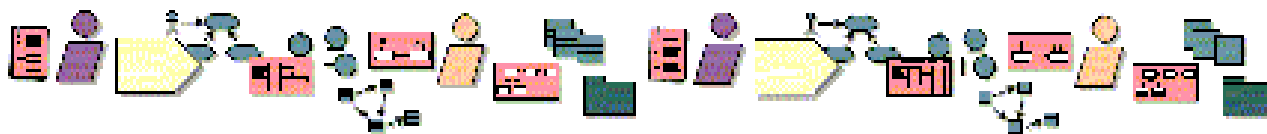
Используйте маленькие хитрости ремесла:

- Спросите каждого, все ли субъекты учтены.
- Предложите некоторые заведомо неправильные варианты. Это побудит участников исправить Вашу ошибку и уточнить роли.
- Всегда принимайте все предложения - Вы сможете удалить дубликаты и не-субъекты позже. Критика чьих-то предложений только ухудшит обстановку семинара.

Определение субъектов обычно занимает от 1 до 4 часов. На доске теперь должно быть нарисовано много субъектов, но удостоверьтесь, что все еще есть место для рисования прецедентов. Этим Вы займетесь на следующем семинаре. Не льстите себя надеждой, что Вы все сделали правильно. Вы не можете быть убеждены в обнаружении всех субъектов, пока не нашли и не описали все прецеденты.

Пример моделирования: деловые субъекты

Мне представляется, что специалисты Classics, Inc. поступили вполне разумно, когда выделили деловые субъекты в отдельный пакет модели прецедентов. Таким путем они сразу расставили все точки над *i* – определили границы моделирования.

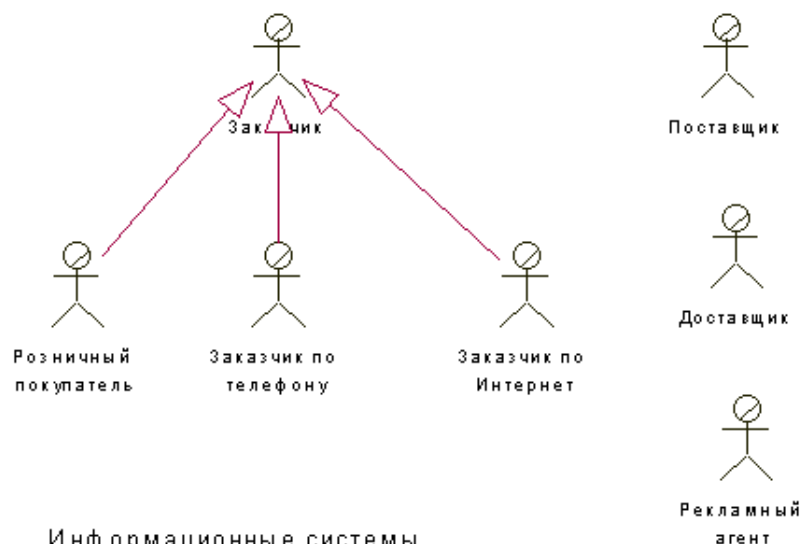


Вот что у них получилось:

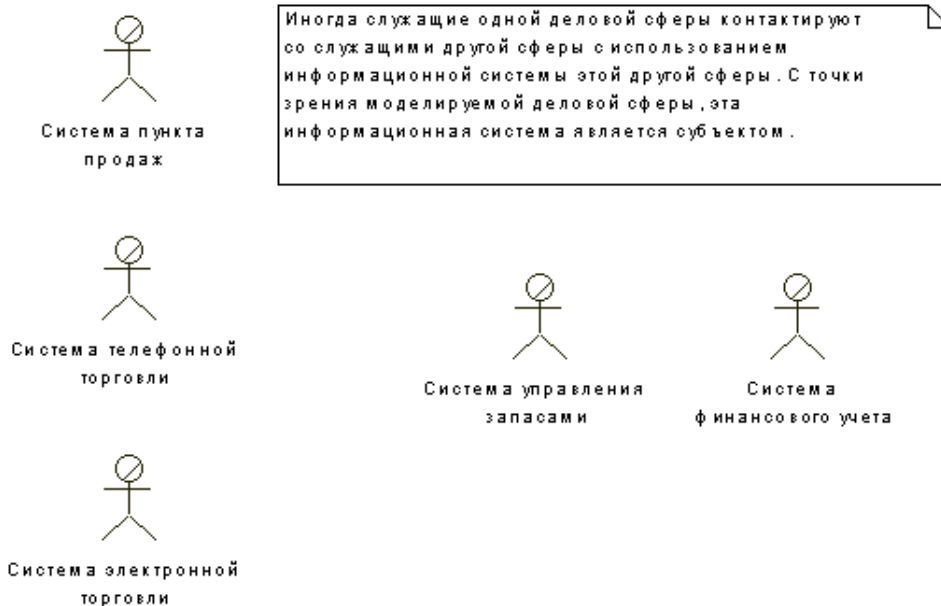
Деловые субъекты

Деловой субъект представляет роль, которую кто-то или что-то играет по отношению к предприятию в деловой сфере.

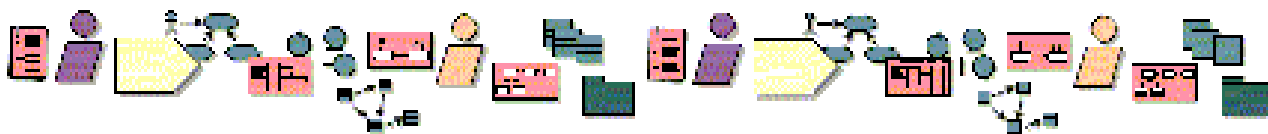
Заказчики



Информационные системы



Пакет Деловые субъекты модели деловых прецедентов Classics, Inc.



Наиболее важным деловым субъектом для компании Classics, Inc. является Заказчик.

Диаграмма показывает, что имеются три типа заказчиков, по разному взаимодействующих с предприятием: Розничный покупатель, Заказчик по телефону и Заказчик по Интернет. Связи обобщения отражают тот факт, что эти три типа субъектов наследуют свойства Заказчика.

На диаграмме показаны пять деловых субъектов, представляющих информационные системы. Два из них – это уже упоминавшиеся выше системы финансового учета и управления запасами «из коробок». Возможно, руководство компании решило продолжить их использование, и поэтому аналитики рассматривают эти системы как внешние субъекты.

Несколько сложнее для понимания определение в качестве деловых субъектов систем торговли (пункта продаж, телефонной и электронной торговли), из-за которых и заваривалась вся каша с созданием новой системы. Думаю, мы поймем это в дальнейшем.

Поиск деловых прецедентов

Для нахождения первичных деловых прецедентов рассмотрите, что получает каждый из деловых субъектов от предприятия. Начните с первоочередных и наиболее важных деловых субъектов - заказчиков - и задайте себе вопрос:

- Какие основные услуги оказывает предприятие заказчику?

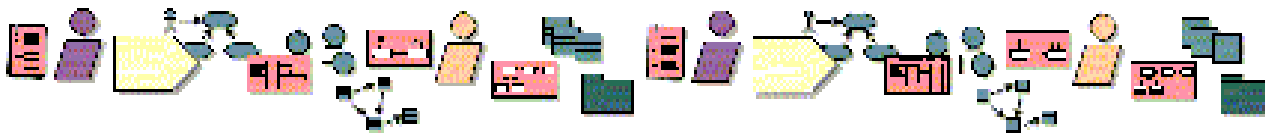
Хорошую помощь может оказать изучение жизненного цикла заказчика. Как происходит первый контакт заказчика с предприятием? Какие стадии или состояния проходит заказчик при взаимодействии с предприятием?

По той же схеме просмотрите все деловые субъекты и задайте себе вопросы:

- Что делает предприятие для субъекта?
- Что делает субъект для предприятия?

Процессы поддержки деловой деятельности (они содержат действия, из которых заказчик не извлекает непосредственную выгоду) также могут быть представлены как деловые прецеденты. Ищите следующие виды действий:

- Развитие и поддержка персонала.
- Развитие и поддержка информационной технологии.
- Развитие и поддержка офиса.
- Защита.



■ Правовые действия.

Процессы управленческого характера могут быть представлены как деловые прецеденты, даже при том условии, что они реже представляют интерес с точки зрения информационной системы. Эти процессы обнаруживаются при анализе действий, которые имеют отношение к управлению деятельностью в целом. Процесс этого вида обычно взаимодействует с субъектом (субъектами) владельца. Рассмотрите, как субъект владеет взаимодействует с деловой сферой. Ищите следующие виды действий:

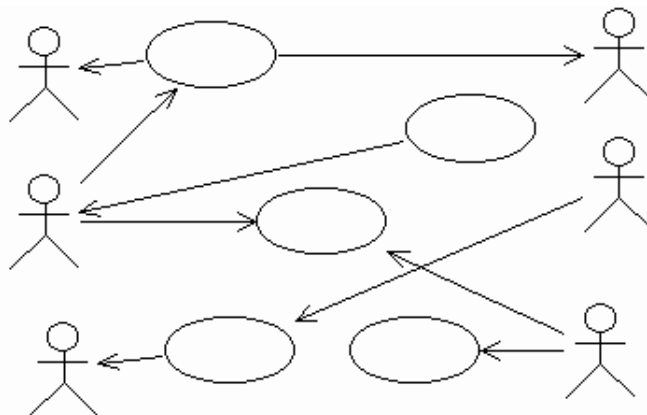
- Подготовка и представление информации владельцам и инвесторам.
- Выдвижение целей долгосрочного бюджета.
- Координация и расположение по приоритетам других прецедентов деятельности.
- Создание новых процессов в деловой сфере.
- Контроль процессов деятельности.

Жизненные циклы процессов этого вида часто составляют один финансовый год.

Другой способ нахождения деловых прецедентов состоит в том, чтобы предложить специалистам по проблемной области описать каждое действие в деловой сфере, и затем сгруппировать эти действия в деловые прецеденты.

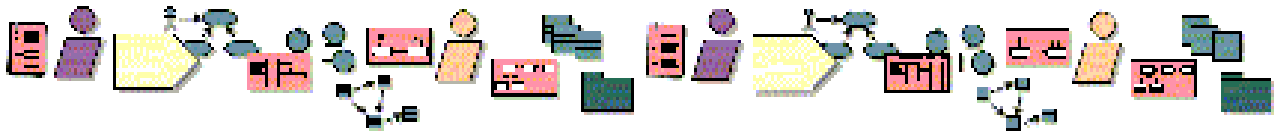
Семинар по деловым прецедентам

Сотрите с доски облако-предприятие и начните идентифицировать прецеденты. Сосредоточьтесь на конкретных прецедентах – избегайте дискуссии относительно их связей. Рисуите эллипсы и записывайте названия для каждого предложенного прецедента. Рисуите стрелки к субъектам.



Вот что получилось после рисования эллипсов-прецедентов. Каждый прецедент имеет имя.
Стрелки обозначают связи между субъектами и прецедентами

Используйте как преимущество то, что Вы ничего не знаете о деятельности предприятия (в крайнем случае, притворитесь, что это так). Пусть об этом расскажут участники семинара.



Задавайте уточняющие вопросы. Когда участники обеспечат Вас объяснениями, появится новый прецедент.

Некоторые люди сразу понимают концепцию прецедента, а для других она оказывается сложной. Чтобы поместить эту концепцию в более понятную среду, попросите кого-нибудь нарисовать свое представление работы предприятия. Например, это может быть структурная схема, на которой обозначены задачи предприятия. Такое представление обычно легко иллюстрируется: один из участников может взять мел и показать на этой схеме, как работает предприятие. Спросите, что случится, если изменятся связи - это поможет Вам идентифицировать альтернативные потоки в прецедентах.

Если Вы работаете с административной системой, представлением может быть диаграмма, описывающая перемещение деловой сущности (например, документа) от одного человека к другому и что они, как предполагается, с ней делают. Например, чтобы показать процесс заказа и поставки, диаграмма может отображать офис заказчика, наши офис и склад, и склад заказчика.

Позвольте прецедентам иметь длинные названия. Недавно идентифицированный прецедент может иметь названием целое предложение - это будет хорошим заделом для краткого описания прецедента. Впоследствии название будет сокращено.

Всегда будет присутствовать ряд прецедентов, которые, как кажется, содержат части общего. Убедитесь, что все понимают, что на данном этапе это приемлемо. Нет никакого смысла в структурировании, так как мы пока мало знаем о содержании прецедентов. Вы должны подождать, пока будут определены потоки событий, и только потом начинать обсуждения связей прецедентов.

Когда участники семинара придут к выводу, что прецеденты на грифельной доске отображают деятельность предприятия, сделайте перерыв.

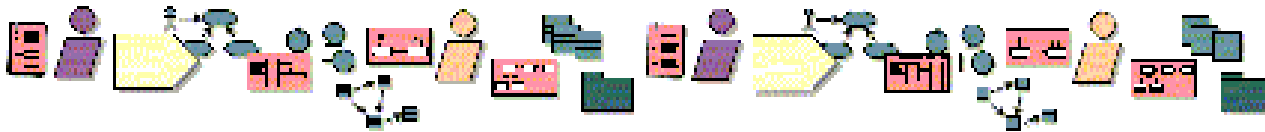
Приступая к работе после перерыва, сделайте обзор результатов предыдущего сеанса:

- Рассмотрите каждый субъект: каковы его задачи в этой системе? Слово «задача» может оказаться понятнее слова «прецедент» для людей, не знакомых с методикой моделирования прецедентов.
- Рассмотрите каждый предложенный прецедент: вы понимаете значение выполнения этого прецедента для пользователя? Если прецедент имеет положительный результат, то пользователь захочет его выполнить.
- Рассмотрите каждый предложенный прецедент: является ли он полным прецедентом? Или это только маленькая часть чего-то большего?

Работайте с прецедентами поочередно, один за другим.

Будут появляться новые прецеденты. Некоторые прецеденты исчезнут. Часто полезно затратить некоторое время, чтобы обсудить, действительно ли представлены все необходимые прецеденты.

Созданная модель может быть задокументирована с использованием Rose или RequisitePro и



может быть сгенерирован отчет Обзор модели прецедентов.

Пример моделирования: деловые прецеденты

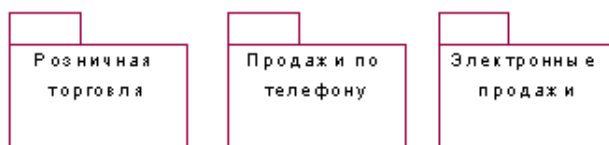
Обратите внимание на то, как хорошо структурируют свои модели специалисты Classics, Inc.

Знакомясь с работой своих коллег, часто приходится видеть диаграммы, размером и числом деталей напоминающие схемы ламповых радиоприемников. (Извините, сегодня, наверное, не все знают, что такое ламповый радиоприемник!) При объектно-ориентированном подходе большой насыщенности диаграмм легко избежать путем их структуризации.

Посмотрите, как сделали это в Classics, Inc.:

Деловые прецеденты (процессы)

Основные формы торговли



Прецеденты управления и поддержки

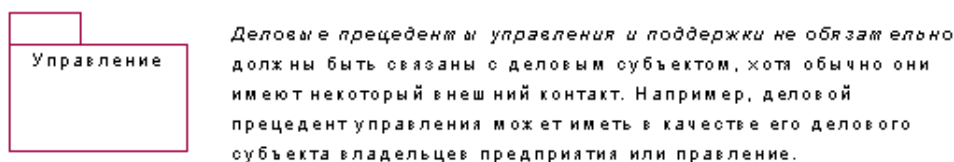


Диаграмма деловых прецедентов Classics, Inc. верхнего уровня

Диаграмма деловых прецедентов верхнего уровня содержит только пакеты прецедентов. Тем не менее, диаграмма дает представление об основных формах деятельности предприятия.

Давайте теперь рассмотрим диаграмму прецедентов, соответствующую пакету Электронные продажи. Скорее всего, именно эту часть Системы обработки и выполнения заказов Classics, Inc. мы будем продолжать изучать в этой и в последующих главах. При необходимости, мы ознакомимся с другой нужной диаграммой прецедентов. Это будет только полезно.

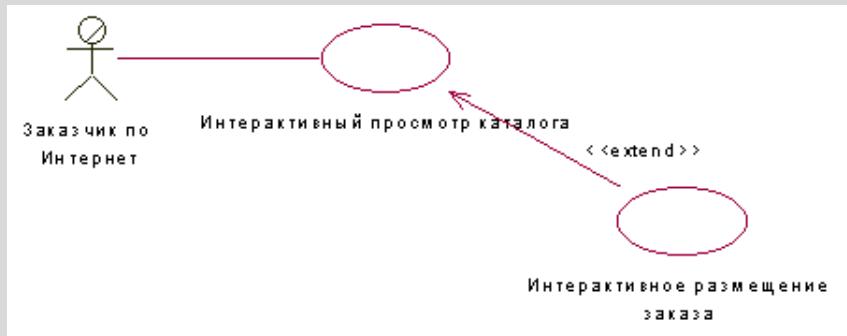
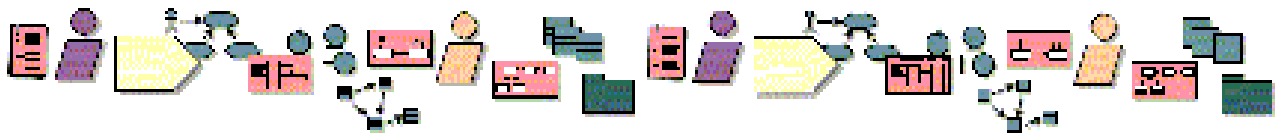


Диаграмма прецедентов электронных продаж

Из диаграммы видно, что Заказчик по Интернет взаимодействует с одним основным прецедентом Интерактивный просмотр каталога. Если же он решил сделать покупку, запускается прецедент расширения Интерактивное размещение заказа.

Известно, что руководство компании Classics, Inc. планирует впоследствии расширить возможности электронной торговли и добавить прецеденты предварительного прослушивания выбранных записей и проверки состояния заказа. Но это они хотят сделать в следующей итерации.

Описание схемы потока работ делового прецедента

Часто, чтобы понять цель делового прецедента, Вам нужна схема потока работ. Она может быть выполнена в форме пошагового описания. Человеку, который позже будет выполнять спецификацию делового прецедента, даже если это тот же самый человек, будет необходимо это пошаговое описание.

В простых случаях это может быть текстовое описание. Но достаточно часто поток работ состоит из основного потока и одного или нескольких альтернативных подпотоков. Простого текстового описания в таких случаях бывает недостаточно, и необходимо использовать диаграмму действий.

Пример моделирования: описания потоков работ

Специалисты компании Classics, Inc. весьма рационально отнеслись к описанию потоков работ деловых прецедентов.

Посмотрите на диаграмму прецедентов, соответствующую пакету Продажи по телефону:

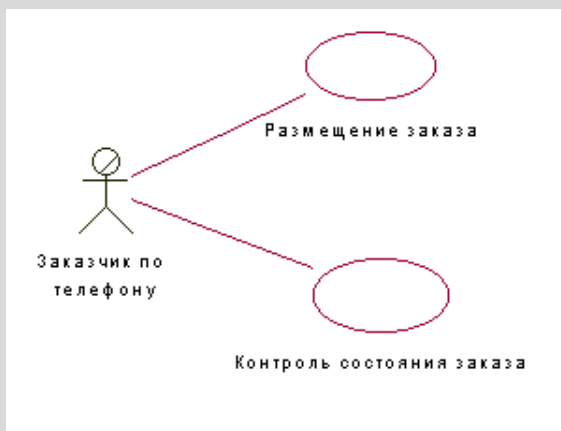
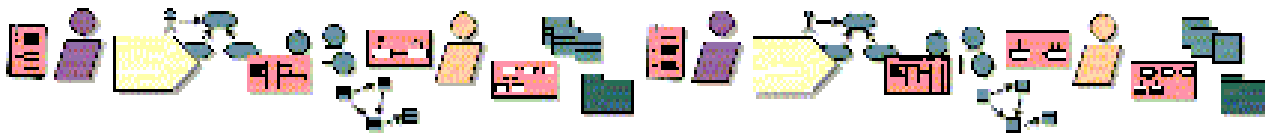


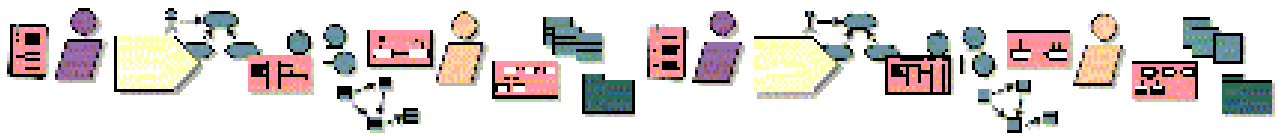
Диаграмма прецедентов продаж по телефону

Размещение заказа по телефону – довольно тривиальная задача. Специалисты компании Classics, Inc. описали поток работ этого прецедента в виде текста в диалоговом окне спецификации прецедента Rational Rose:

The dialog box has a title bar 'Use Case Specification for Размещение заказа'. It contains several tabs: 'General', 'Diagrams', 'Relations', 'Files', and 'Synchronizer'. The 'General' tab is active. It includes fields for 'Name' (set to 'Размещение заказа'), 'Package' (set to 'Продажи по телефону'), 'Stereotype' (set to 'business use-case'), and 'Rank'. There is an 'Abstract' checkbox which is unchecked. A large text area labeled 'Documentation:' contains the following text: 'Заказчик звонит по номеру приема заказов Classics, Inc., и, если линия свободна, говорит с клерком телефонных продаж. Заказчик сообщает клерку, какие товары он хотел бы заказать. Если заказчик не знает точный номер позиции, клерк помогает заказчику найти нужный товар, пользуясь базой данных товаров. В конечном счете, заказчик подтверждает спецификацию, и просит предоставить счет. Клерк запрашивает имя заказчика, адрес и информацию для оплаты. Клерк подтверждает оплату'. At the bottom are buttons for 'OK', 'Cancel', 'Apply', 'Browse', and 'Help'.

Диалоговое окно спецификации прецедента Размещение заказа

Для сравнения посмотрите на описание потока работ прецедента Просмотр интерактивного каталога:



Use Case Specification for Интерактивный пр... ? X

General Diagrams Relations Files Synchronizer

Name: Package: Электронные продажи

Stereotype:

Rank: ☐ Abstract

Documentation:

Заказчик по Интернет использует браузер сети, чтобы загрузить домашнюю страницу компании Classics Inc. Заказчик просматривает каталог, выбирая товары способом, подобным складыванию покупок на тележку в обычном магазине.

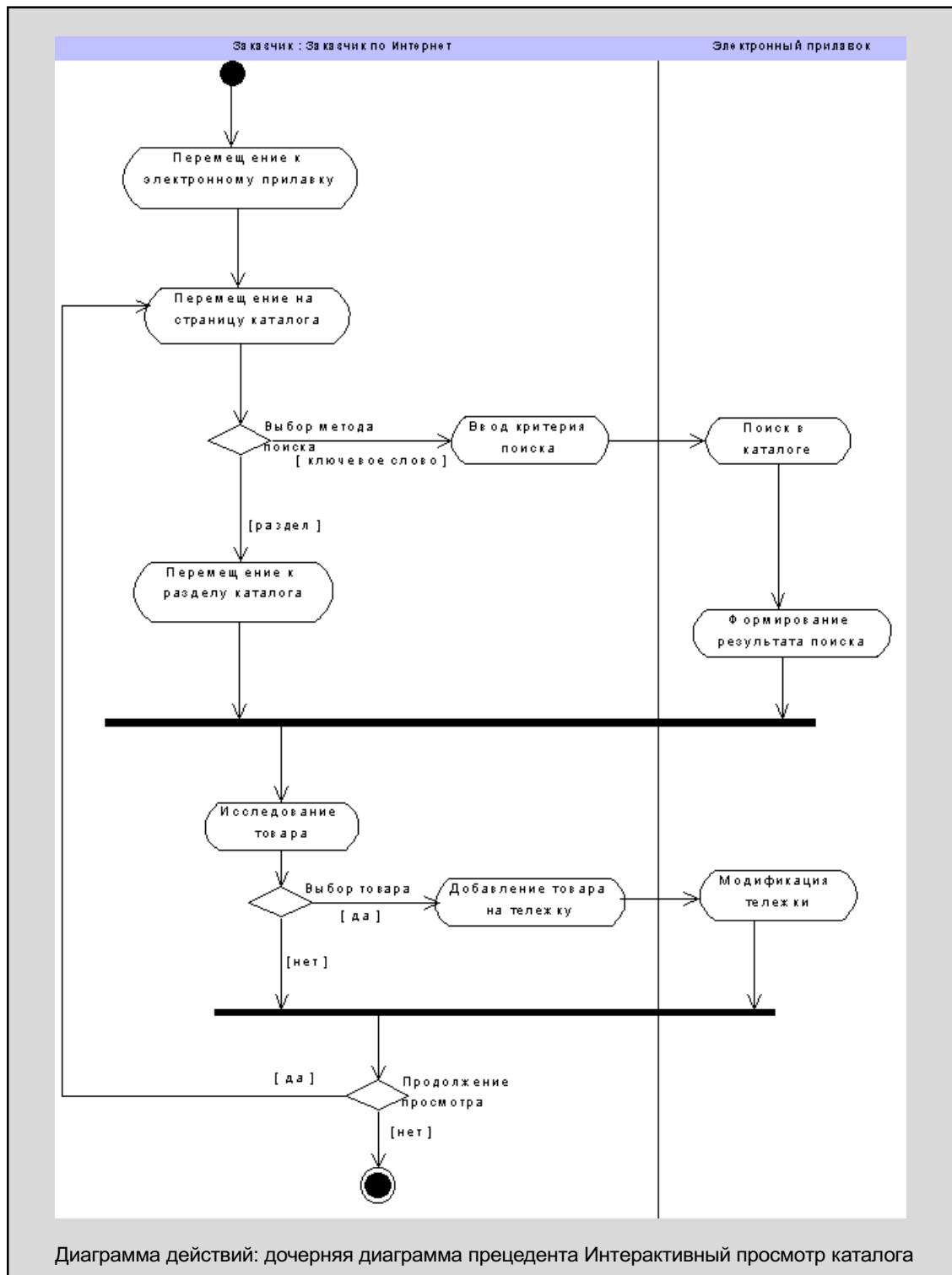
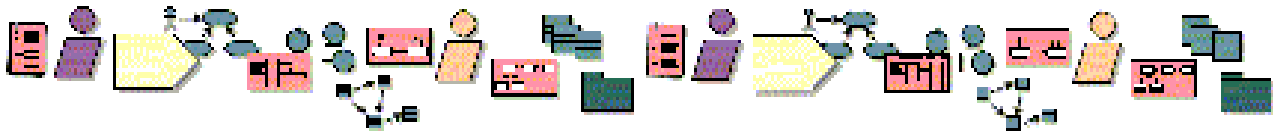
Заказчик может искать по секциям или по видам товаров, или использовать словесные описания товаров.

OK Cancel Apply Browse Help

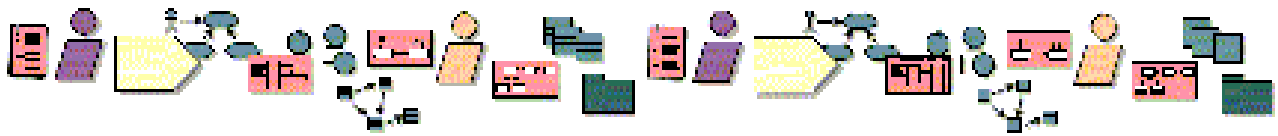
Диалоговое окно спецификации прецедента Интерактивный просмотр каталога

Мы видим, что текстового описания здесь явно недостаточно. Для понимания спецификации гораздо лучше использовать диаграмму действий, что и сделали специалисты компании Classics, Inc.

Посмотрите, что у них получилось:



Моделированию деловых объектов мы посвятим следующую главу.



Поток работ «Деловое моделирование» (окончание)

Модель деловых объектов

Напомним, что модель деловых объектов – это объектная модель, описывающая реализацию деловых прецедентов (см. стр. 8-10). Она описывает прецедент деловой сферы с внутренней точки зрения деловой сферы. Модель определяет, как люди, которые работают в деловой сфере, и вещи, которые они обрабатывают и используют (классы и объекты деловой сферы), должны быть связаны друг с другом, статически и динамически, чтобы произвести ожидаемые результаты.

Элементами модели деловых объектов могут быть деловые работники, деловые сущности и связи между ними. Для структуризации модели могут использоваться организационные модули и пакеты.

Все вместе объекты модели должны быть способны выполнить все прецеденты деловой сферы – **не больше и не меньше**.

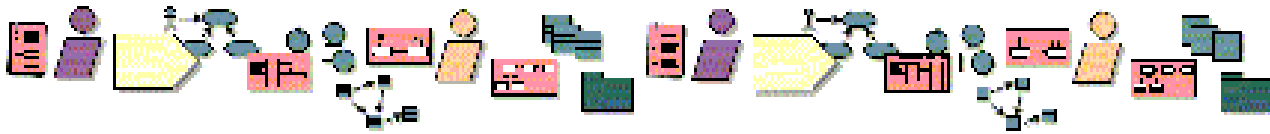
Идентификация организационных модулей

Выделите и кратко опишите каждый организационный модуль в пределах моделируемой деловой сферы.

Рекомендуется, чтобы Вы давали организационному модулю название, которое отражает компетентность его деловых работников. В качестве альтернативы можно использовать названия существующих организационных единиц.

Обсудите и убедитесь, что Вы договорились о том, какие организационные модули должны быть включены в вашу модель.

Детальное описание некоторых из идентифицированных Вами организационных модулей может иметь очень ограниченное значение для моделирования целевой системы. С другой стороны, они могут поставлять важную исходную информацию или принимать выходную информацию. Тогда, скорее всего, они должны моделироваться как деловые субъекты; это означает, что они являются внешними по отношению к моделируемой деловой сфере (см. стр. 8-5).



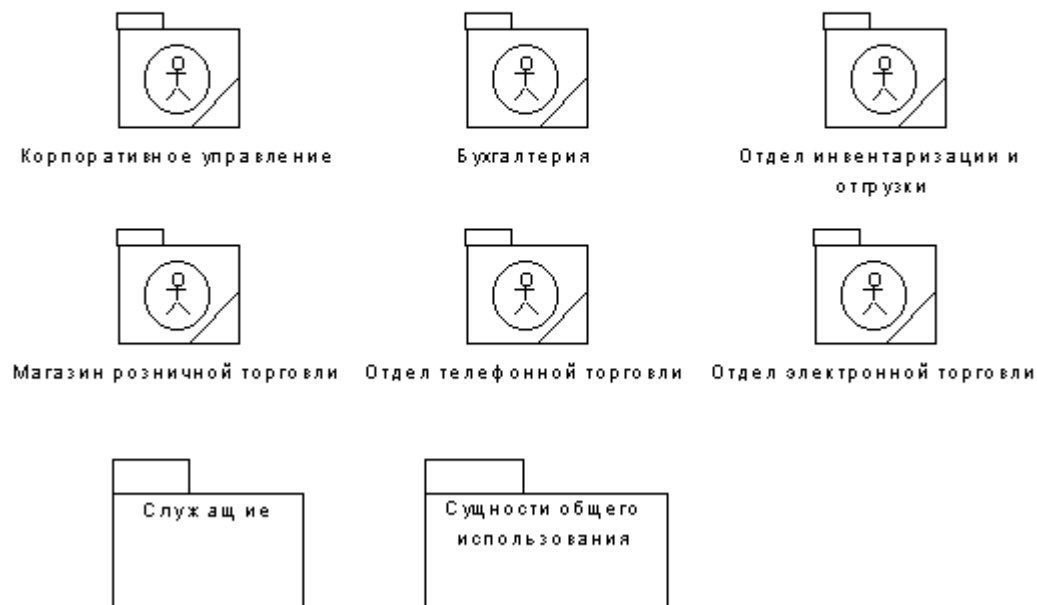
Пример моделирования: Структура модели деловых объектов

На следующем рисунке представлена главная диаграмма классов модели деловых объектов, которую разработали специалисты Classics, Inc.

Модель деловых объектов

Организационные модули

Организационный модуль включает деловых работников, деловые сущности и другие организационные модули, которые объединяются в соответствии с некоторым критерием

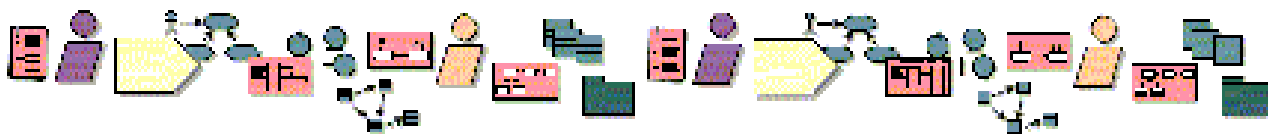


Главная диаграмма классов модели деловых объектов Classics, Inc.

Специалисты Classics, Inc. использовали организационные модули для структуризации модели деловых объектов. Судя по названиям, структура модели отражает организационную структуру компании (текущую или предполагаемую).

Сравните эту диаграмму с диаграммой деловых прецедентов на странице 9-2. Каждой из основных форм торговли из диаграммы прецедентов соответствует организационный модуль на диаграмме классов. Видимо, предполагается, что за реализацию пакетов прецедентов основных форм торговли отвечает соответствующий организационный модуль.

Сказанное выше не означает, что в реализации пакета прецедентов участвуют



только деловые работники и деловые сущности соответствующего подразделения компании или что подразделение компании отвечает только за реализацию соответствующего пакета прецедентов. Судя по диаграмме, это прекрасно понимают специалисты Classics, Inc.

Посмотрите на диаграмму. В структуре Classics, Inc. присутствует Отдел инвентаризации и отгрузки. Процессы отгрузки товаров заказчикам (выполнения заказов) свойственны всем основным формам торговли. Эти процессы «вынесены за скобки» и за их реализацию отвечает соответствующий организационный модуль.

С другой стороны, в любой компании существуют процессы, которые касаются всех служащих и всех подразделений. Вспомните, хотя бы, о начислении зарплат и об учете рабочего времени. Каждый служащий составляет лист учета рабочего времени, каждый администратор проверяет листы учета рабочего времени подчиненных и передает подтвержденные сведения в бухгалтерию. Эти процессы также «вынесены за скобки». Специалисты Classics, Inc. смоделировали реализацию этих процессов в пакете Сущности общего назначения.

Общее правило состоит в следующем: любой элемент (организационный модуль, деловой работник, деловая сущность или связь) могут быть непосредственными элементами только одного организационного модуля (любой элемент модели должен быть определен только один раз!). Но это не ограничивает возможности использования одних и тех же элементов в разных представлениях и на различных диаграммах.

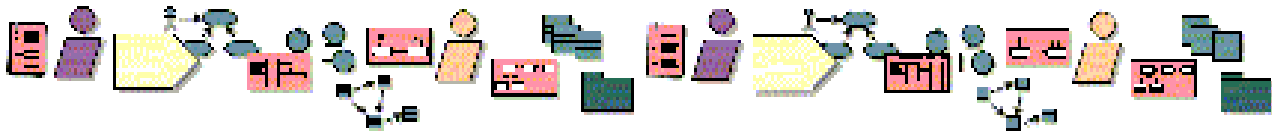
Взгляните на диаграмму еще раз. Она содержит пакет под названием Служащие. Первой задачей моделирования деловых объектов является определение всех участников реализации прецедентов. Можно предположить, что специалисты Classics, Inc. начали именно с этого, по ходу поиска включая найденных деловых работников в упомянутый пакет.

Поиск деловых работников

Деловой работник представляет роль или набор ролей в деловой сфере. Деловой работник взаимодействует с другими работниками и управляет деловыми сущностями, участвуя в реализации делового прецедента (см. стр. 8-11).

Для каждой роли, которую Вы можете идентифицировать в организации, должен быть определен деловой работник. Дайте ему краткое описание.

Продуктивный способ поиска деловых работников состоит в том, чтобы работать группой, используя грифельную доску и листки самоклеющейся бумаги.

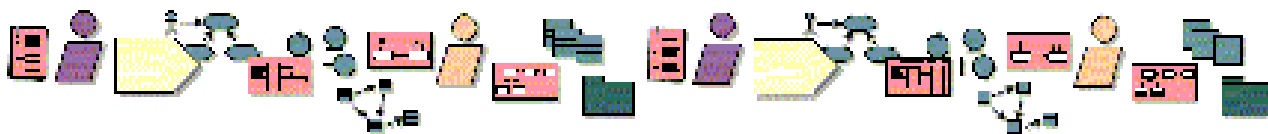


Изучите поток работ каждого делового прецедента. Просмотрите каждый раздел, чтобы понять ответственности, которые соответствовали бы одному деловому работнику. Убедитесь, что Вы не пропустили никаких деловых работников, и что все найденные Вами деловые работник являются «внутренностью» той части деловой сферы, которую Вы моделируете.

Не расстраивайтесь, если сначала деловых работников окажется слишком много или слишком мало: Вы еще вернетесь к этому после идентификации деловых сущностей.

Характеристики хорошего делового работника:

- Его название и описание ясны и понятны.
- Каждый деловой работник имеет связь, по крайней мере, с одной деловой сущностью.
- Каждый деловой работник имеет связи с другими деловыми работниками.
- Связи деловых работников не зависят друг от друга.
- Каждый деловой работник участвует по крайней мере в одном деловом прецеденте.
- Каждая связь используется в потоке работ по крайней мере одного делового прецедента.
- Каждая из операций делового работника выполняется в потоке работ по крайней мере одного делового прецедента.



Пример моделирования: Деловые работники

Результаты работы специалистов Classics, Inc. по идентификации деловых работников отображены на диаграмме классов под названием Служащие.

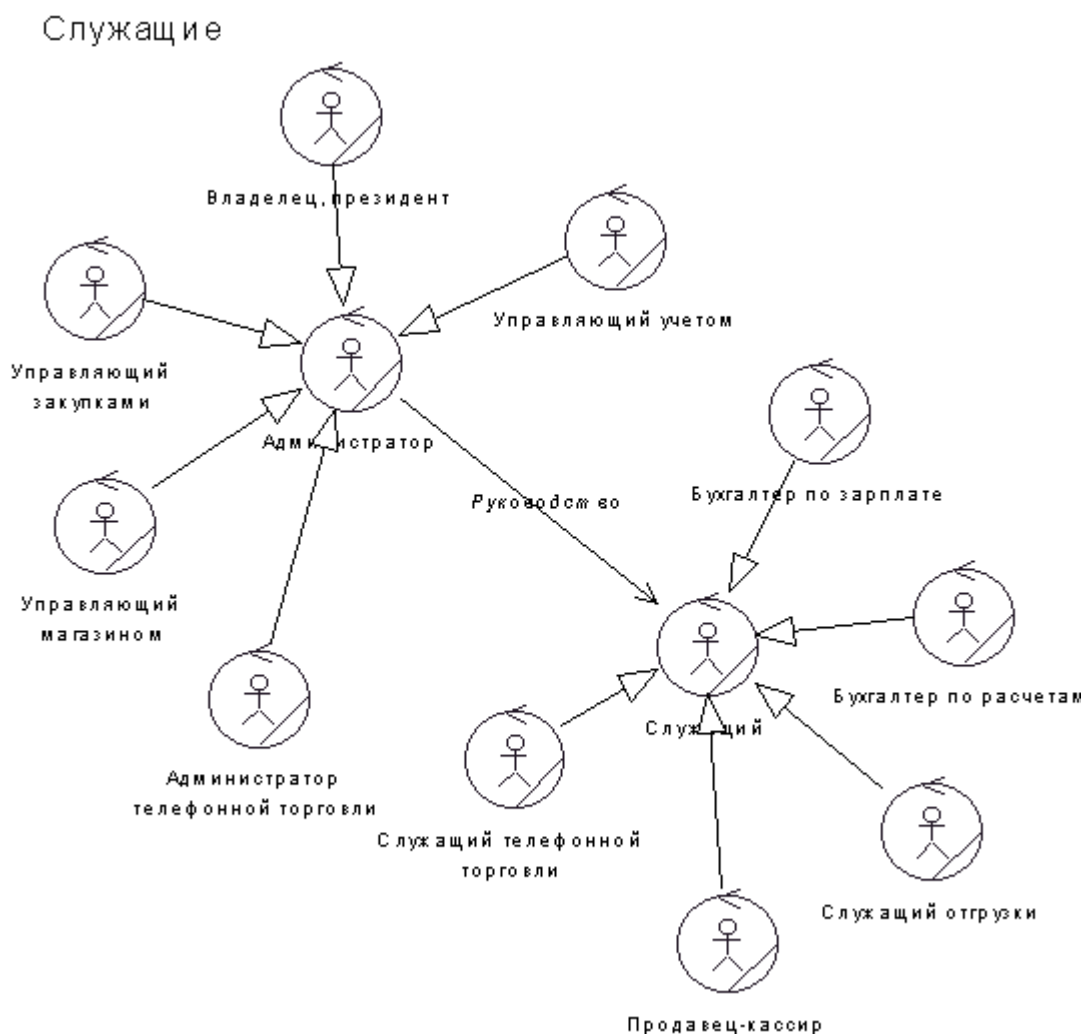
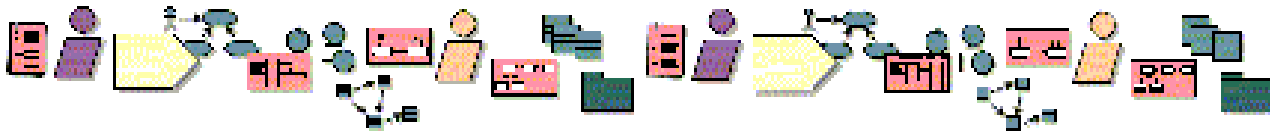


Диаграмма классов Служащие представляет деловых работников Classics, Inc.

Специалисты Classics, Inc. обратили внимание на то, что помимо своей основной роли, указанной в названии, каждый деловой работник имеет некоторые дополнительные обязанности, связанные с его положением служащего или администратора.

Например, все служащие обязаны раз в неделю представлять своему руководителю (администратору) лист учета рабочего времени (см. стр. 10-2).



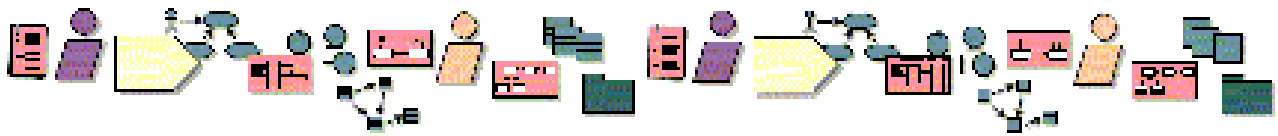
Эта их обязанность и другие общие свойства определены для делового работника под названием Служащий, который является обобщением для соответствующих деловых работников (посмотрите на связи обобщения).

Аналогично, деловой работник Администратор является обобщением для всех деловых работников – руководителей.

Посмотрите, как специалисты Classics, Inc. описали делового работника Администратор в диалоговом окне спецификации в Rational Rose:

Окно спецификации делового работника Администратор

Деловой работник Администратор телефонной торговли является потомком делового работника Администратор. Он наследует все свойства Администратора и имеет дополнительные свойства.



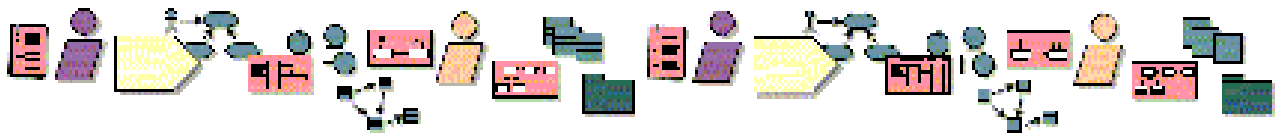
Посмотрите, как специалисты Classics, Inc. конкретизировали делового работника Администратор телефонной торговли:

Окно спецификации делового работника Администратор телефонной торговли

Хотелось бы напомнить, что деловой работник – это роль или набор ролей в деловой сфере. В отличие от этого, конкретный служащий компании (индивидуальный работник) может играть несколько ролей: один из служащих телефонной торговли может выполнять обязанности соответствующего администратора.

В заключение этого раздела, небольшая ложка дегтя.

1. Посмотрите на диаграмму. Из названий деловых работников следует, что за электронную торговлю никто не отвечает. Хотя в структуре модели есть организационный модуль Отдел электронной торговли. Нет ли здесь ошибки?
2. Посмотрите на диаграмму еще раз. Вам не кажется, что деловые работники Администратор и Служащий также имеют общие свойства? За примером



сущность.

Пример моделирования: Деловые сущности

В этом разделе мы обсудим следующие диаграммы классов модели деловых объектов Classics, Inc.:

- Телефонные продажи (диаграмма организационного модуля Отдел телефонной торговли)
- Электронные продажи (диаграмма организационного модуля Отдел электронной торговли)
- Общее руководство (диаграмма пакета Сущности общего использования)

Телефонная и электронная торговля

Рассмотрите следующую диаграмму классов:

Телефонные продажи

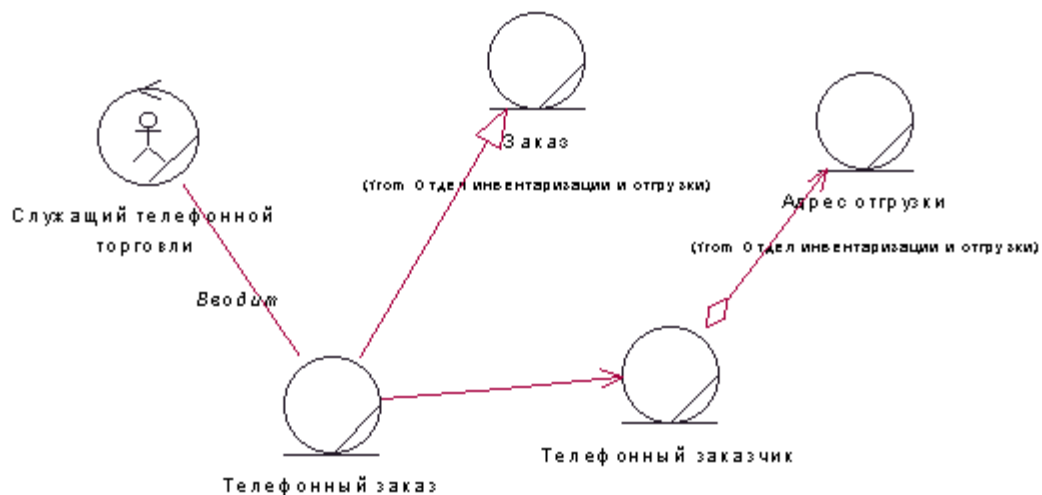
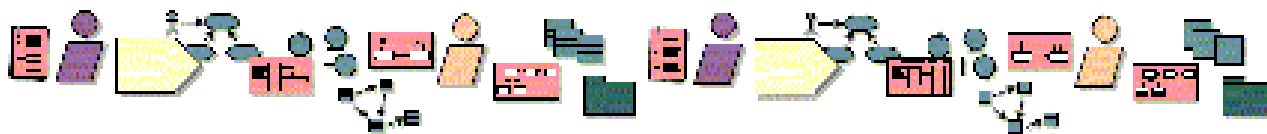


Диаграмма классов Телефонные продажи.

Главным объектом здесь является деловая сущность Телефонный заказ, которая представляет более общую сущность Заказ. Обратите внимание, что деловая сущность Заказ принадлежит организационному модулю Отдел инвентаризации и отгрузки. Это совершенно правильное решение специалистов Classics, Inc. Общие для всех основных форм торговли “вещи” вынесены за скобки.

По тем же соображениям и деловая сущность Адрес отгрузки, являющаяся частью деловой сущности Телефонный заказчик, также принадлежит организационному модулю Отдел инвентаризации и отгрузки.



Единственным действующим лицом телефонных продаж является деловой работник Служащий телефонной торговли, который, по указаниям заказчика, фиксирует (вводит в систему) информацию о заказе (Телефонный заказ) и заказчике (Телефонный заказчик). Эта информация используется в организационном модуле Отдел инвентаризации и отгрузки для комплектования и упаковки заказанных товаров и отправки их заказчику.

Посмотрите теперь на диаграмму классов Электронные продажи:

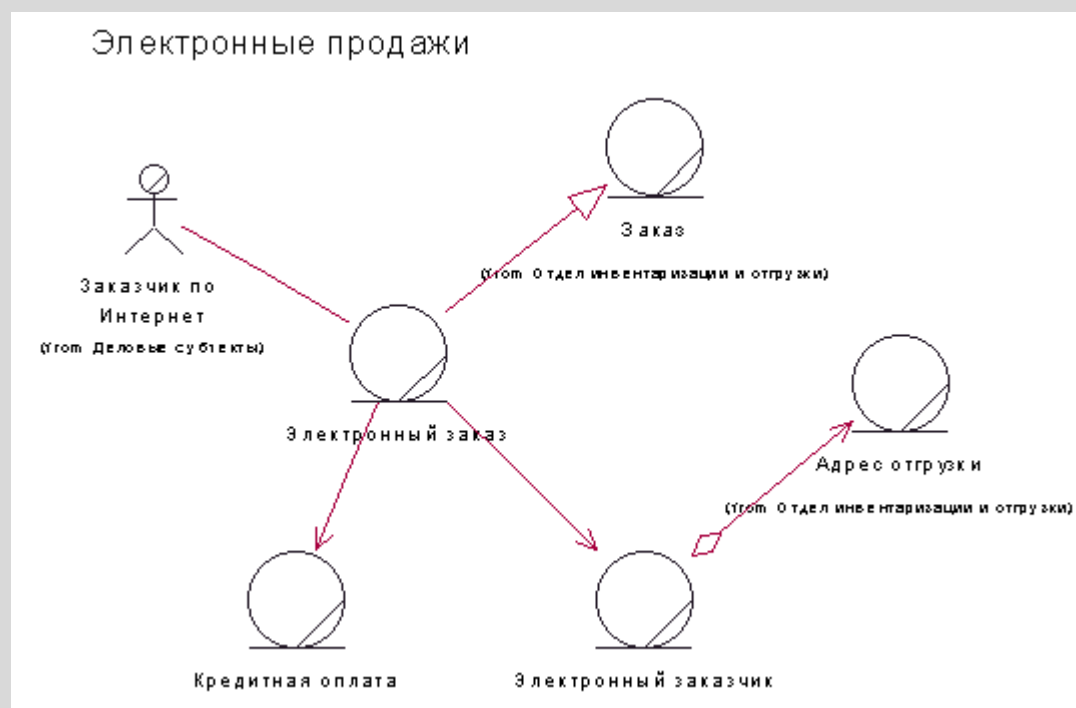
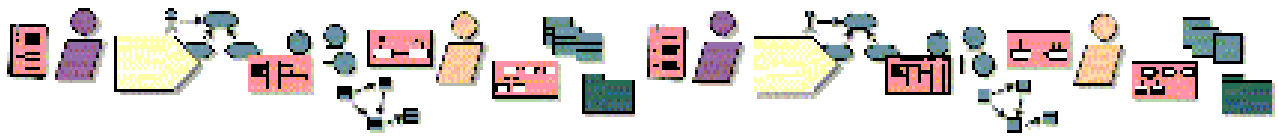


Диаграмма классов Электронные продажи

Диаграмма очень похожа на предыдущую, поэтому нет смысла ее подробно описывать.

Деловая сущность Электронный заказ является специализацией деловой сущности Заказ применительно к электронной торговле, а деловая сущность Электронный заказчик отличается от деловой сущности Телефонный заказчик хотя бы тем, что первая обязательно содержит адрес электронной почты заказчика.

На диаграмме появилась новая деловая сущность Кредитная оплата, фиксирующая оплату стоимости заказа с помощью кредитной карточки. Обратите внимание, что эта деловая сущность не принадлежит непосредственно организационному модулю Отдел электронной торговли, а заимствована из



организационного модуля Магазин розничной торговли. По-видимому, там вопрос кредитной оплаты был решен раньше, и специалисты Classics, Inc. решили не ломать себе голову над этой проблемой на этапе делового моделирования.

Существенным отличием является то, что организационный модуль Отдел электронной торговли не содержит непосредственно ни одного делового работника. Размещением заказа в системе занимается деловой субъект Заказчик по Интернет.

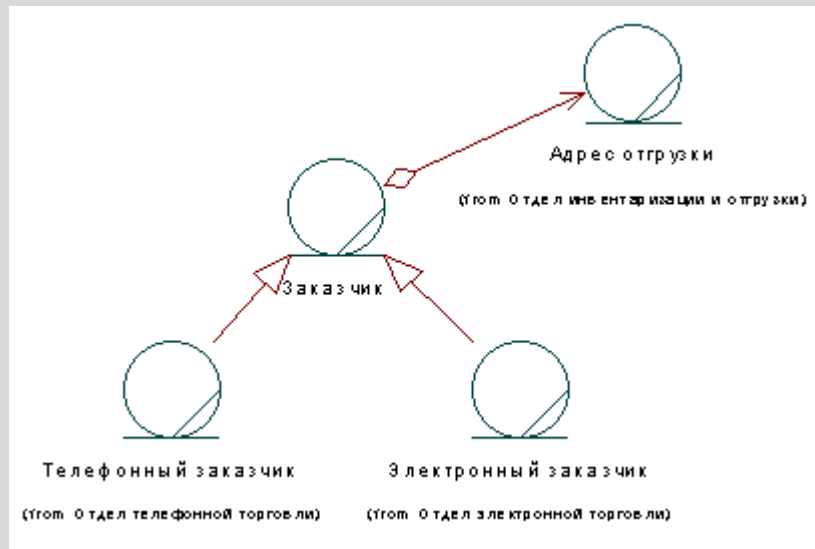
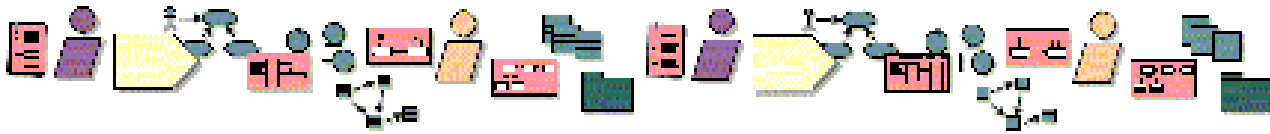
Таким образом, отвечая на первый вопрос (см. стр. 10-5), мы должны снять со специалистов Classics, Inc. одно из обвинений: Отдел электронной торговли действительно не содержит ни одного делового работника. Однако корректность названия этого организационного модуля вызывает большие сомнения.

Примечание: Рекомендации Rational Unified Process по использованию UML для делового моделирования не предусматривают размещение деловых субъектов на диаграммах классов, т.к. деловые субъекты и диаграммы классов с деловыми работниками и деловыми сущностями видимы из разных “точек зрения”, соответственно внешней и внутренней. Поэтому на диаграмме Электронные продажи, которую разработали специалисты Classics, Inc., деловой субъект Заказчик по Интернет и соответствующая ассоциация с деловой сущностью Электронный заказ не показаны. С другой стороны, ни Rational Unified Process, ни инструментальные средства Rational Rose напрямую этого не запрещают. Поэтому я взял на себя смелость несколько откорректировать диаграмму, чтобы ситуация с отсутствием делового работника стала более понятна.

В заключение раздела хотелось бы обратить внимание на способ использования специалистами Classics, Inc. связи объединения (“целое-часть”).

Посмотрите еще раз на обе диаграммы выше. Вы видите, что деловая сущность Адрес отгрузки является частью двух деловых сущностей: Телефонный заказчик и Электронный заказчик. Трудно себе представить, чтобы Тормозная колодка была одновременно частью двух разных Автомобилей. В нашем же случае все выглядит вполне естественно: информация об адресе отгрузки является составной частью информации о телефонном и об электронном заказчике.

Все правильно, но на мой взгляд, при объектно-ориентированной разработке более последовательным в этой ситуации было бы определение деловой сущности-обобщения Заказчик, как показано на следующей диаграмме:



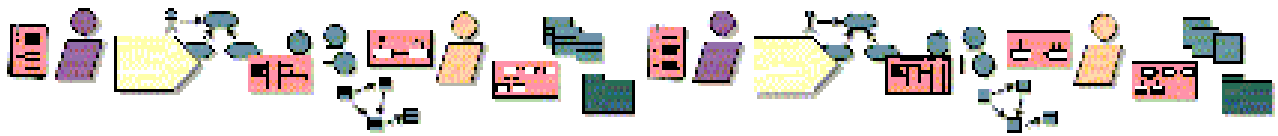
Альтернативный способ представления информации о заказчике и адресе отгрузки заказа

Общее руководство

Диаграмма Общее руководство содержится в пакете Сущности общего использования (см. диаграмму на стр. 10-2). На ней представлены объекты, которые касаются всех служащих компании: индивидуальная информация о каждом служащем, графики работы служащих и сведения о фактически отработанном рабочем времени, сведения для расчета зарплаты и о фактических выплатах и т.п.

Диаграмма достаточно проста. Я предлагаю подробнее исследовать только один аспект.

Помните, на стр. 10-6 мы говорили о том, что деловые работники Администратор и Служащий имеют некоторое общее свойство: раз в неделю они хотят получить за свой труд достойное вознаграждение? Сейчас мы имеем возможность узнать, предусмотрели ли это специалисты Classics, Inc.



Общее руководство

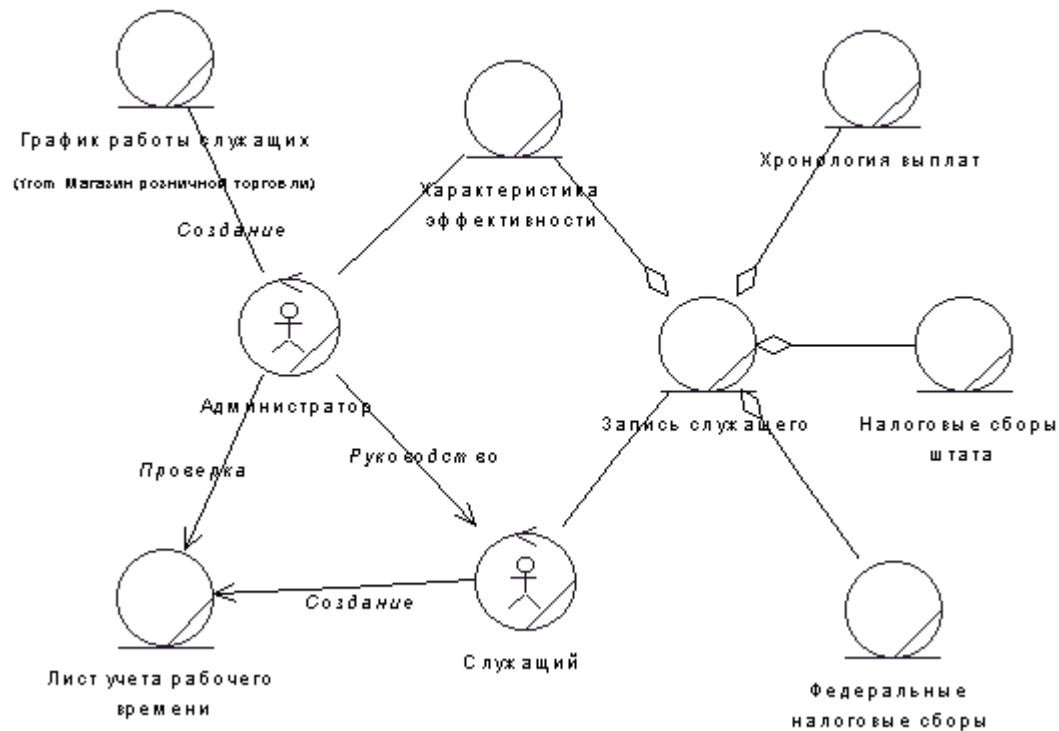
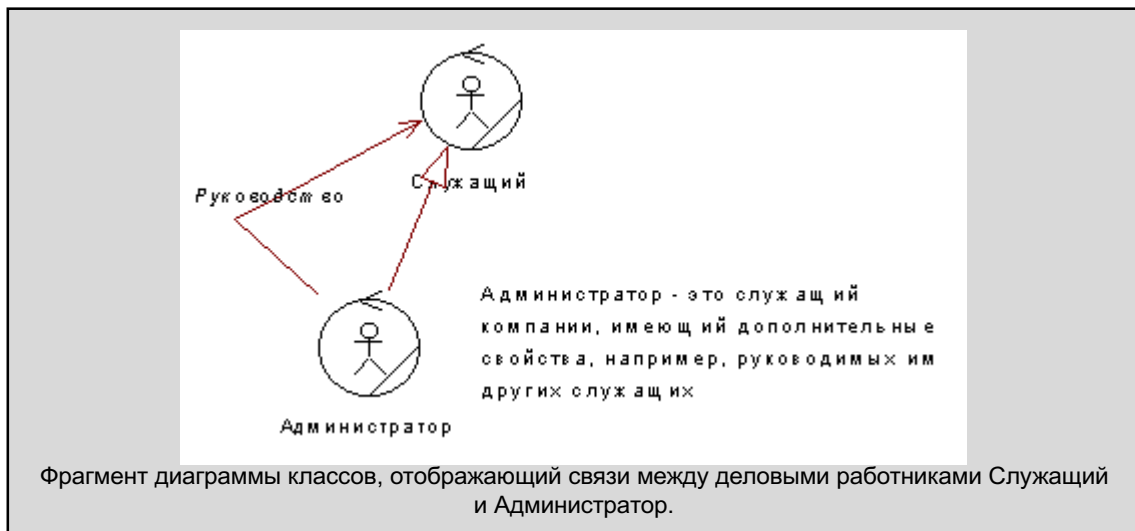
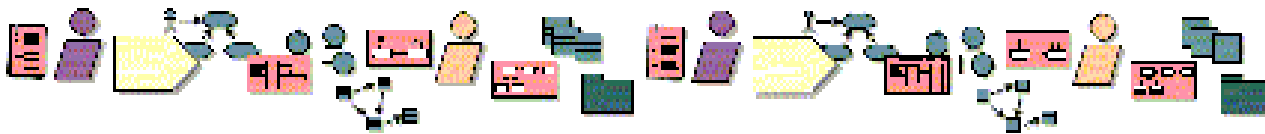


Диаграмма классов Общее руководство

Основанием для расчета зарплаты являются деловые сущности **Запись служащего** и **Лист учета рабочего времени**. Деловой работник **Служащий** имеет связи ассоциации с указанными деловыми сущностями. Из этого можно сделать вывод, что кто-то рассчитает служащему зарплату. Деловой работник **Администратор** таких связей не имеет. Значит ли это, что руководящий персонал Classics, Inc. работает «за идею»?

Скорее всего, нет. Видимо, специалисты Classics, Inc. считают, что любой руководитель является и служащим компании. Но из модели этого не следует.

Что можно посоветовать для исправления этого упущения в модели? Самое простое – это определить связь обобщения от делового работника **Администратор** к деловому работнику **Служащий**, как показано ниже:



Определение реализаций деловых прецедентов

Выделите деловых работников и деловые сущности, которые участвуют в выполнении каждого делового прецедента.

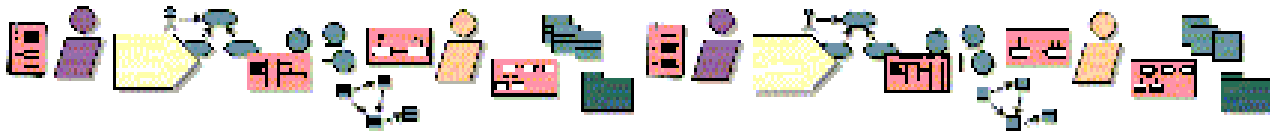
Представьте деловых работников и деловые сущности, участвующие в реализации делового прецедента, в диаграмме сотрудничества. Показывайте только те связи, которые необходимы для понимания того, как взаимодействуют деловые работники и деловые сущности в потоке работ реализации делового прецедента.

Вместо использования групповой диаграммы сотрудничества Вы можете представлять участвующих деловых работников и деловые сущности в диаграмме последовательности. Диаграмма последовательности может заменять сложное текстовое описание. Поскольку она часто оказывается слишком детальной, используйте диаграмму последовательности только для потоков работ или частей потоков работ, которые являются сложными.

Для пояснения значения групповой диаграммы сотрудничества Вы можете описать поток работ каждой реализации делового прецедента в терминах его элементов, взаимодействующих деловых работников, обрабатывающих деловые сущности. Это необязательно, и имеет смысл только для наиболее сложных потоков работ или частей потоков работ.

- Начните с описания нормального потока работ реализации делового прецедента.
- Когда нормальный поток работ описан и уже относительно устоялся, начните описывать любые альтернативные и необязательные потоки работ.
- Определите цели представления деловых работников и деловых сущностей в терминах стоимости и времени производственного цикла.

Проанализируйте жизненный цикл каждой деловой сущности. Каждая деловая сущность должна



Деловое моделирование и системное моделирование обращаются к двум различным областям проблем на двух различных уровнях абстракции. Отсюда вытекает общее правило: информационные системы не должны иметь никакого прямого присутствия в деловых моделях.

С другой стороны, служащие, действующие как работники, используют информационные системы для связи друг с другом и с субъектами и для обращения к информации относительно деловых сущностей. Всякий раз, когда имеется некоторая связь, ассоциация или атрибут, потенциально имеется также некоторая поддержка информационной системой.

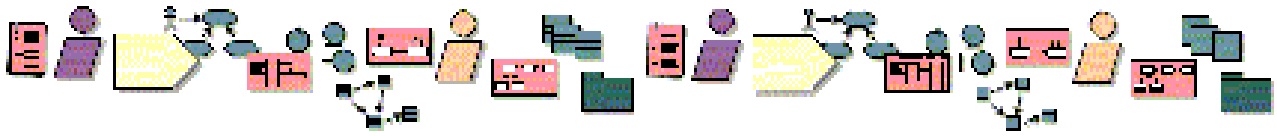
Указанные два контекста моделирования имеют следующие отношения:

- Служащий, действующий как определенный работник, соответствует системному субъекту информационной системы. Он, вероятно, поддерживается лучше всего, если информационные системы структурированы так, чтобы вся его работа в деловом прецеденте поддерживалась одним прецедентом информационной системы.
- Иначе, если деловой прецедент большой, имеет большую продолжительность или соединяет работу из нескольких независимых областей, прецедент информационной системы мог бы поддерживать одну операцию работника.
- Вещи, с которыми работают служащие, смоделированные как деловые сущности, часто имеют свои представления в информационных системах. В объектной модели информационной системы эти деловые сущности проявляются как классы сущностей.
- Связи ассоциации и объединения между деловыми сущностями часто отображаются в соответствующих связях ассоциации и объединения между классами сущностей в модели проекта.
- Таким образом, потенциально, прецедент информационной системы обращается и управляет классами сущностей модели проекта, представляющими деловые сущности, к которым обращается поддерживаемый деловой прецедент.
- Наконец, деловой субъект, который непосредственно использует информационную систему деловой сферы, также становится системным субъектом информационной системы.

Эти отношения существенны при идентификации требований к информационным системам, которые поддерживают деловую сферу.

Подробности информационной системы в модели деловых объектов

Общее правило состоит в том, что информационные системы не должны быть смоделированы явно в модели деловых объектов; они являются только инструментальными средствами в руках работников. Rational Unified Process допускает одно исключение из этого правила, которое касается информационных систем, используемых непосредственно заказчиками. Если это



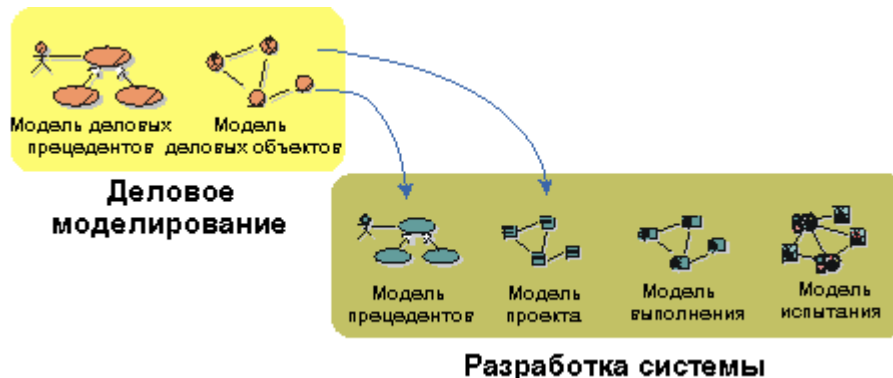
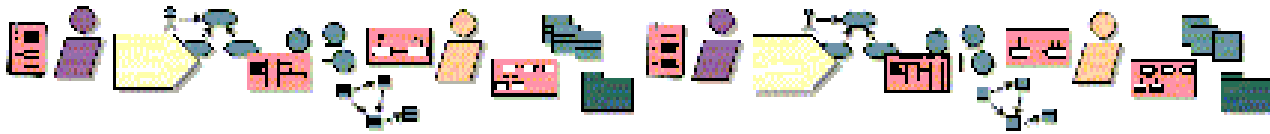
взаимодействие формирует главную часть деловых услуг, оно может быть настолько коммерчески важно, что Вы захотите показать его в деловой модели. Телефонное банковское обслуживание – хороший пример информационной системы такого типа.

С позиций делового моделирования предлагается следующий подход:

- Представьте информационную систему в виде полностью автоматизированного делового работника, который взаимодействует с субъектом.
- Если информационная система связывается с любым другим деловым работником или с деловой сущностью, рассмотрите возможность проиллюстрировать эти отношения связью или ассоциацией. Возможно, система информирует делового работника относительно своего продвижения или использует информацию относительно деловой сущности.
- Кратко опишите в модели деловых объектов делового работника со списком услуг, которые предоставляет информационная система.
- Моделируйте все подробности и характеристики информационной системы и ее среды в модели информационной системы.
- Введите соглашение об именах так, чтобы полностью автоматизированный деловой работник был легко идентифицирован среди других деловых работников; например префикс или суффикс, подобно «автоматизированный <название работника>» или «<название работника> (вычислительная система)». Вы можете даже определять стереотип со специфическим значком.

Переход от деловых моделей к системным

Подход к деловому моделированию, представленный в Rational Unified Process, предлагает короткий и прямой способ порождения требований для поддержки информационной системы. Чтобы иметь возможность сформировать правильную систему, очень важно хорошо понимать деловые процессы. Это еще более важно, если Вы учитываете также роли и ответственности людей, а также определения тех «вещей», которые при этом обрабатываются. Отсюда происходит более глубокое представление организации (зафиксированное в модели деловых объектов), где Вы можете увидеть самую тесную связь с тем, что должны представлять модели системы.



Отношения между деловыми моделями и моделями поддержки информационной системы.

Деловые модели и архитектура системы

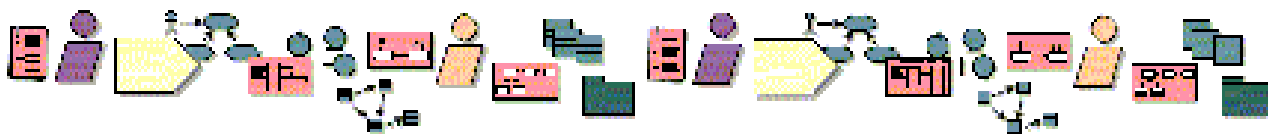
С архитектурной точки зрения особенно полезно иметь деловые модели, если вы намереваетесь сформировать систему одного из следующих видов:

- Заказную систему для одной или более компаний в определенном типе индустрии (банки, страховые компании).
- Семейство прикладных программ для свободного рынка (системы заказа услуг, системы составления счетов, системы управления воздушным движением).

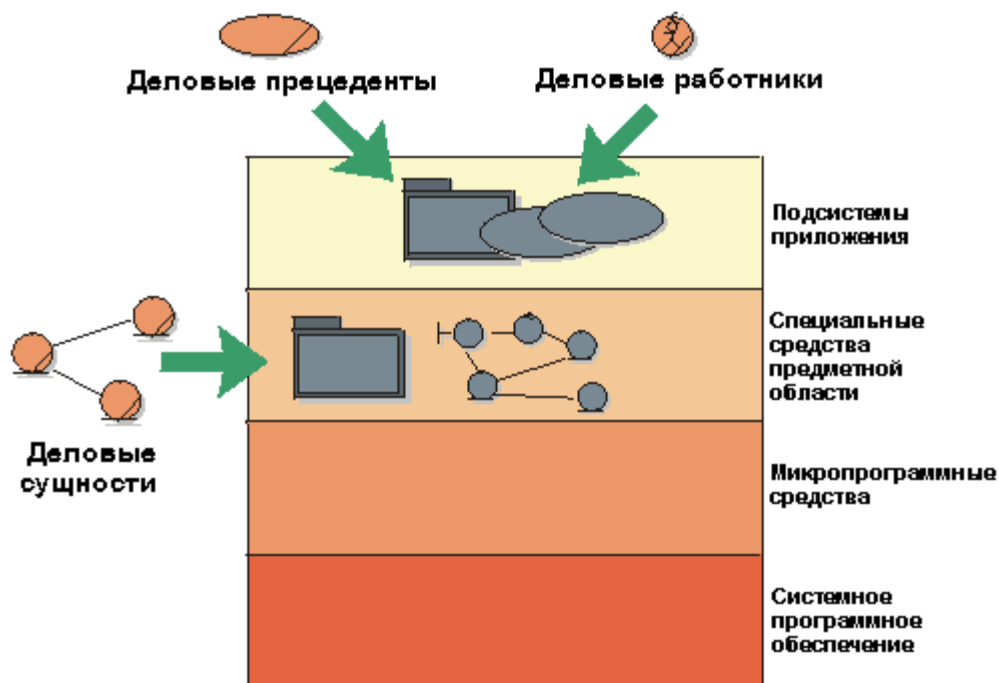
Деловые модели могут дать исходную информацию по видам прецедентов и логической картине представления в модели анализа. Вы можете также найти ключевой механизм на уровне анализа (механизмы анализа).

Необходимо рассмотреть следующие моменты:

- Для каждого делового прецедента, который должен поддерживаться системой, идентифицируйте подсистему в модели анализа. Эта подсистема находится на прикладном уровне и должна рассматриваться первой итерацией прототипа. Например, если в вашей модели деловых прецедентов есть процессы заказа и составления счетов, Вы должны идентифицировать на прикладном уровне вашей модели анализа подсистемы Заказ и Составление счетов. Но Вы можете оспорить это и считать их не подсистемами, а отдельными системами. Неважно, это является вопросом контекста. Если Вы рассматриваете всю вашу систему информационной поддержки как одну систему с несколькими прикладными программами, которые совместно используют общую архитектуру, Заказ и Составление счетов будут прикладными подсистемами. Если же ваша задача состоит в формировании только приложения Управление заказами, то Управление заказами и есть ваша система, и предложенная выше рекомендация не имеет смысла. То есть, данная выше рекомендация имеет смысл, только если Вы рассматриваете всю систему информационной поддержки в организации как одну цельную систему.



- Для каждого делового работника, который должен поддерживаться системой, идентифицируйте прецеденты, представляющие то, что должно быть автоматизировано.
- Для каждой деловой сущности, которая должна поддерживаться системой, идентифицируйте классы сущности в модели анализа. Некоторые из них - кандидаты, которые нужно рассматривать как ключевые механизмы («компоненты сущности») в системе.
- Для «кластеров» деловых сущностей (групп деловых сущностей, которые используются исключительно в пределах одного делового прецедента, или, иначе, групп тесно связанных деловых сущностей), создают подсистему на уровне деловой спецификации.

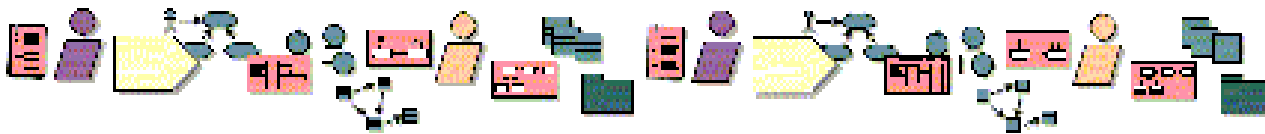


В четырехуровневой архитектуре системы деловые модели дадут исходную информацию двум верхним слоям.

Деловые модели и субъекты в системе

Чтобы идентифицировать прецеденты информационной системы, начните с деловых работников в модели деловых объектов. Для каждого делового работника выполните следующие шаги:

- Решите, будет ли деловой работник использовать информационную систему.
- Если да, то идентифицируйте субъекта в модели прецедентов информационной системы. Дайте субъекту то же название, что и деловому работнику.
- Если деловой работник имеет ответственность в пределах делового прецедента (одно или несколько действий), которая предполагает использование информационной системы,



идентифицируйте прецедент информационной системы для каждой такой ответственности. Дайте прецеденту название, соответствующее ответственности, которую он поддерживает.

- Если ответственность в пределах прецедента деловой системы фрагментирована, Вы можете идентифицировать для него несколько прецедентов информационной системы. «Фрагментированная» ответственность охватывает много видов действий, выполняемых в разное время, которые не имеют никакой прямой причинно-следственной связи.
- Повторите эти шаги для всех деловых работников.

Таким же образом Вы должны рассмотреть каждого делового субъекта. Если деловой субъект будет использовать систему непосредственно, Вы должны идентифицировать субъекта информационной системы, соответствующего деловому субъекту.

Деловые модели и классы сущностей в модели анализа

Деловая сущность, которая должна быть представлена в информационной системе, будет соответствовать сущности в модели анализа информационной системы. Однако в некоторых случаях может быть правильнее позволить атрибутам деловой сущности соответствовать сущностям в модели информационной системы. К деловой сущности могут обращаться несколько деловых работников. Следовательно, соответствующие сущности в системе могут участвовать в нескольких прецедентах информационной системы.

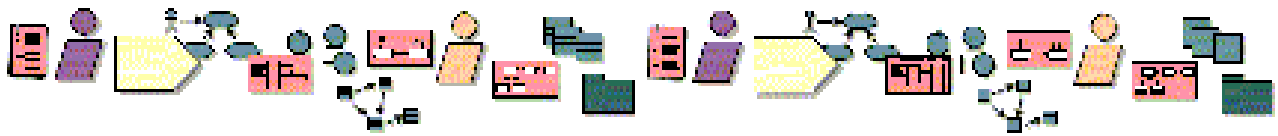
Трансляция взаимодействий между работниками в требования к системе

Как Вы должны интерпретировать связь между работниками в деловой модели? Вы должны выяснить, как информационные системы могут поддерживать общающихся работников. Информационная система может устранить потребность транспортировки информации между работниками, производя информацию, доступную в информационной системе.

Различные контексты делового моделирования

Вы можете встретить три главные типичные ситуации:

- Вы выполняете деловое моделирование, чтобы помочь определить требования к одному конкретному приложению, которое создается на пустом месте. Имеются другие приложения, поддерживающие деятельность организации, которые мы должны знать (наследуемые системы). Необходимо решить, должно ли новое приложение заменить некоторые функциональные возможности в старых системах или дополнить наследуемые системы.
- Вы выполняете деловое моделирование, чтобы помочь определить требования для улучшения уже существующего приложения.
- Вы выполняете деловое моделирование, чтобы помочь определить требования к информационным системам в полностью новой сфере деятельности. Целью работы не обязательно является формирование новых приложений. Иногда следует определить, можно



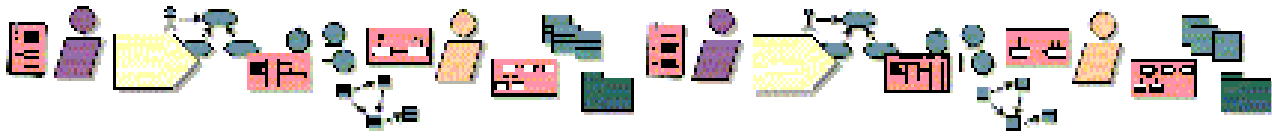
ли использовать приложения, существующие в других сферах деятельности (с модификаций или без модификации).

Что общего в этих «типичных ситуациях», так это то, что цель состоит в определении **подмножества** требований к автоматизации, а не **всех** требований. Это означает, что прежде, чем Вы сможете определить, какие из деловых работников станут системными субъектами вашего приложения, Вы должны идентифицировать, какую из типичных ситуаций Вы пробуете разрешить, а также получить информацию о любых уже существующих приложениях, которые могут сосуществовать с тем, которое Вы разрабатываете.

Итоговая таблица

Следующая таблица суммирует связи между деловыми моделями и моделями информационной системы.

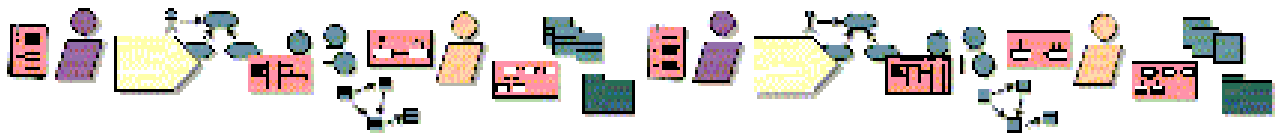
Информационная система	Как искать кандидатов, используя информацию из деловой модели	Деловая сфера
Субъект	Кандидатов субъектов следует искать среди деловых работников.	Деловой работник
Субъект	Других кандидатов субъектов следует искать среди различных деловых субъектов (заказчики, продавцы), которые будут использовать систему непосредственно.	Деловой субъект
Прецедент	Кандидатов прецедентов следует искать среди деловых операций работников. Ищите операции (и области ответственности), которые включают взаимодействия с информационной системой. Идеально, один прецедент информационной системы должен поддерживать все действия работника в пределах одной реализации прецедента деловой модели.	Операция делового работника
Класс сущности	Кандидатов класса сущности следует искать среди деловых сущностей. Ищите деловые сущности, которые должны поддерживаться или быть представлены в информационной системе.	Деловая сущность
Класс сущности	Кандидатов класса сущности следует искать среди атрибутов в модели деловых объектов. Ищите атрибуты, которые должны поддерживаться или быть представлены в информационной системе.	Атрибут
Связь между сущностями классов	Связи между деловыми сущностями часто указывают на соответствующие связи между классами в модели информационной системы.	Связь между деловыми сущностями



Выводы

- Вы можете моделировать деловую сферу для понимания ее структуры и динамики, для гарантии того, что все совладельцы имеют одинаковое понимание организации и для получения требований к системе поддержки организации.
- Для делового моделирования могут приспособливаться и использоваться методы разработки программного обеспечения.
- Существует множество вариантов делового моделирования, и выбор подходящего зависит от назначения системы, которую Вы разрабатываете, и от контекста проекта.
- Из моделей деловой сферы Вы можете получить требования к программному обеспечению.
- Для делового моделирования Вы можете использовать инструментальные средства Rational, в частности – Rational Rose.

В следующей главе мы продолжим обсуждение основных потоков работ Rational Unified Process.



Поток работ «Требования»

Введение

Цели потока работ

Rational Unified Process формулирует следующие цели для основного потока работ Требования:

- Добиться одинакового понимания с заказчиком и пользователями о том, что должна делать система.
- Дать разработчикам системы наилучшее понимание требований к системе.
- Определить границы системы.
- Обеспечить базу для планирования технического содержания итераций.
- Определить интерфейс пользователей с системой.

Для достижения этих целей необходимо собрать и проанализировать **Запросы совладельцев**, разработать артефакты **Видение**, **Модель прецедентов**, **Прецеденты** и **Дополнительные спецификации**, которые описывают, что будет делать система.

Документ Запросы совладельцев - это «список пожеланий» различных совладельцев проекта (заказчиков, пользователей, других заинтересованных лиц) относительно содержания системы, вместе с информацией о том, как каждое пожелание рассмотрено в проекте.

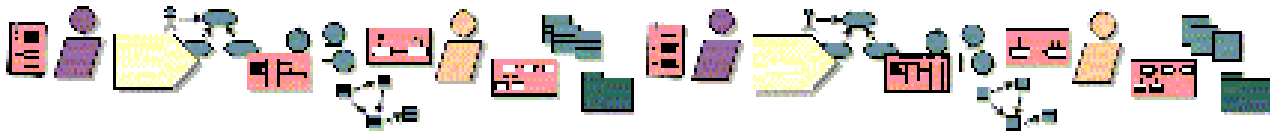
Документ Видение описывает все цели проекта и основные возможности системы.

Модель прецедентов демонстрирует одинаковое понимание системы заказчиком, пользователями и разработчиками. Она позволяет:

- Заказчику и пользователям проверить, будет ли система такой, как они ожидали.
- Разработчикам системы делать именно то, что ожидают заказчик и пользователи.

Модель прецедентов состоит из прецедентов и субъектов. Каждый прецедент в модели подробно, по шагам, описывает, как система взаимодействует с субъектами, и что система делает в прецеденте. Прецеденты функционируют как единая «красная нить» на всем протяжении жизненного цикла программы; одна и та же модель прецедентов используется при анализе системы, проектировании, выполнении и испытаниях (см. стр. 4-6).

Дополнительные спецификации - важное дополнение к модели прецедентов, так как вместе они



фиксируют все необходимые требования к программному обеспечению (функциональные и не функциональные), которые составляют полные технические требования к системе.

Законченные определения прецедентов и Дополнительные спецификации для конкретной «возможности» или некоторой части системы могут объединяться в **Спецификации требований к программному обеспечению** (Software Requirements Specification - SRS).

В дополнение к уже упомянутым, должны быть разработаны следующие артефакты:

- **Глоссарий**
- **Иллюстрированный сценарий прецедента**
- **Прототип интерфейса пользователя**

Глоссарий важен потому, что он определяет общую терминологию для всех моделей и текстовых описаний требований к системе.

Иллюстрированные сценарии прецедентов и прототип интерфейса пользователя - это результаты моделирования интерфейса пользователя и прототипирования, которые выполняются параллельно с другими действиями разработки требований.

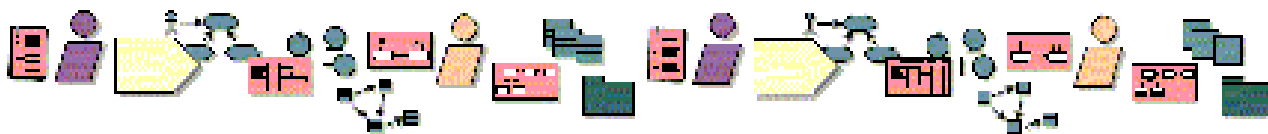
Поток работ Требования связан с другими потоками работ процесса.

- Поток работ **Деловое моделирование** представляет организационный контекст системы.
- Поток работ **Анализ и проектирование** получает из Требования свои первичные исходные данные (модель прецедентов и глоссарий). В ходе анализа и проектирования могут быть обнаружены недостатки модели прецедентов; в этих случаях генерируются и применяются к модели прецедентов **запросы изменения**.
- Поток работ **Испытание** тестирует систему, чтобы проверить код модели прецедентов. Прецеденты и Дополнительные спецификации обеспечивают исходную информацию для требований, используемых при планировании и проектировании испытаний.
- Поток работ **Среда** разрабатывает и обслуживает артефакты поддержки, которые используются при моделировании прецедентов, типа **Указаний по моделированию прецедентов** и **Указаний по разработке интерфейса пользователя**.
- Поток работ **Управление проектом** планирует проект, разрабатывая **План управления требованиями**, и каждую итерацию (разрабатывая **План итерации**). Модель прецедентов важна для планирования действий итерации.

Требования

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

Существует много различных видов классификаций требований. Rational Unified Process



использует для этой цели так называемую **FURPS+** модель, использующую акроним FURPS для обозначения главных категорий требований с их подкатегориями, как показано ниже:

- **Functionality** (Функциональность)
- **Usability** (Применимость)
- **Reliability** (Надежность)
- **Performance** (Производительность)
- **Supportability** (Пригодность к эксплуатации)

Значок «+» в названии FURPS+ помогает Вам не забыть включать также и такие требования, как

- проектные ограничения,
- требования выполнения,
- требования к интерфейсу и
- физические требования.

(См. стандарт IEEE Std 610.12.1990)

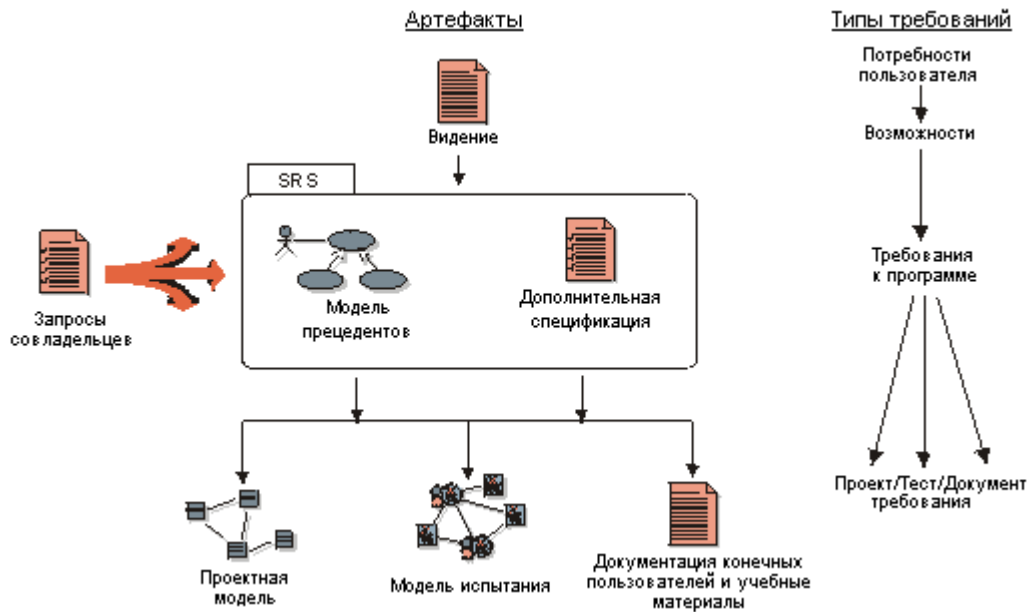
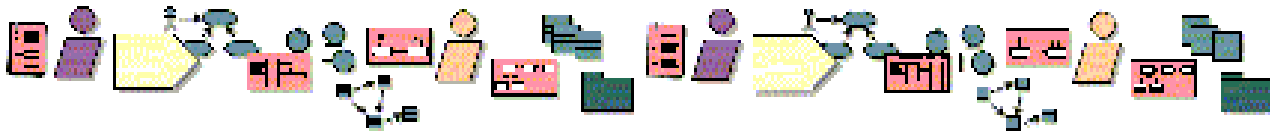
Функциональные требования определяют действия, которые должна быть способна выполнить система, без рассмотрения физических связей. Лучше всего они описываются в модели прецедентов и в прецедентах. Функциональные требования, таким образом, определяют внешнее поведение системы типа “ввода – вывода”.

Многие требования не являются функциональными. Они описывают только атрибуты системы или атрибуты среды. Такие требования называются **не функциональными требованиями**.

Я думаю, Вы уже успели убедиться, что одной из причин неудач традиционных процессов разработки программных систем (например, процесса «водопада»), является представление требований неизменными. Rational Unified Process с самого начала отвергает это предположение и вводит понятие «управление требованиями». Управление требованиями – это систематический подход к обнаружению, документированию, организации и сопровождению изменяющихся требований к системе (см. стр. 4-12).

Традиционно требования рассматривались как текстовые предписания, касающиеся одной из категорий, упомянутых выше на этой странице. Каждое требование – это условие или возможность, которой должна соответствовать система.

Для эффективного управления требованиями мы должны больше знать о том, что мы поддерживаем как требования. Rational Unified Process вводит понятие **типов требований**, чтобы разделить различные уровни абстракции и целей представления требований. Каждый тип требования должен иметь уникальный набор атрибутов, связанных с каждым требованием, определенным на этом уровне.



Уровни абстракции и типы требований

Мы должны следить за неоднозначными пожеланиями и за формальными запросами наших совладельцев, чтобы удостовериться, что мы знаем, как о них позаботиться. Документ Видение помогает нам отслеживать ключевые потребности пользователя и возможности системы.

Указания Видения проекта определяют необходимые пользователям возможности программы и требования к ней. Прецеденты или другие формы модели и Дополнительные спецификации управляют требованиями к проекту.

Требования разделены на функциональные и не функциональные требования, и представлены в моделях и диаграммах анализа и проектирования.

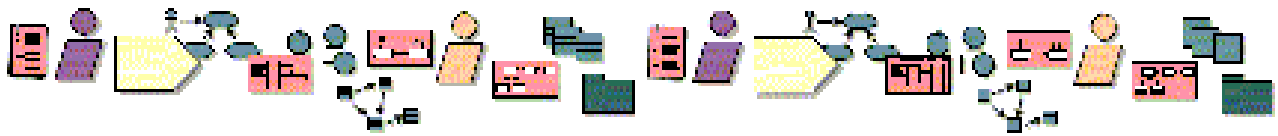
Требования к испытаниям, производные от вышеупомянутых, - это процедуры тестирования и другие артефакты испытаний.

Точно так же появляются требования, касающиеся содержания документации поддержки конечных пользователей и учебных материалов.

Трассируемость требований

Цель установления **трассируемости** требований состоит в том, чтобы помочь:

- Оценить последствия для проекта, вызванные изменением требования



-
- Оценить последствия неудачного испытания требования (то есть, если испытание терпит неудачу, требование не может быть удовлетворено)
 - Управлять контекстом проекта
 - Проверять, что все требования к системе выполнены
 - Проверять, что приложение делает только то, что оно должно было делать
 - Управлять изменениями

Рассмотрите еще раз рисунок на стр. 11-4.

Трассируемость помогает Вам проследить, как запросы совладельцев оттранслированы в набор свойств в документе Видение и в Модель прецедентов, которые в свою очередь детализированы в прецедентах и в Дополнительных спецификациях.

Далее Вы должны проследить, как эти детальные технические требования оттранслированы в проект, как они проверены и как они документированы для пользователя.

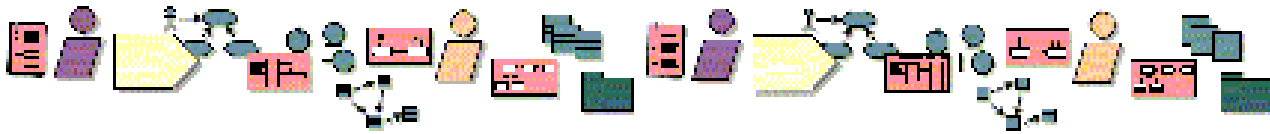
Трассируемости должны быть установлены так, чтобы они помогли Вам ответить на следующий типовой набор вопросов:

- Покажите мне потребности пользователя, которые не связаны с возможностями программы.
- Покажите мне состояние испытаний по всем прецедентам в итерации #n.
- Покажите мне все дополнительные требования, связанные с испытаниями, состояние которых не проверено.
- Покажите мне результаты всех испытаний, которые потерпели неудачу, в порядке их критичности.
- Покажите мне свойства, намеченные для этого выпуска, в удовлетворении которых нуждается пользователь, и их состояние.

Инструментальная поддержка

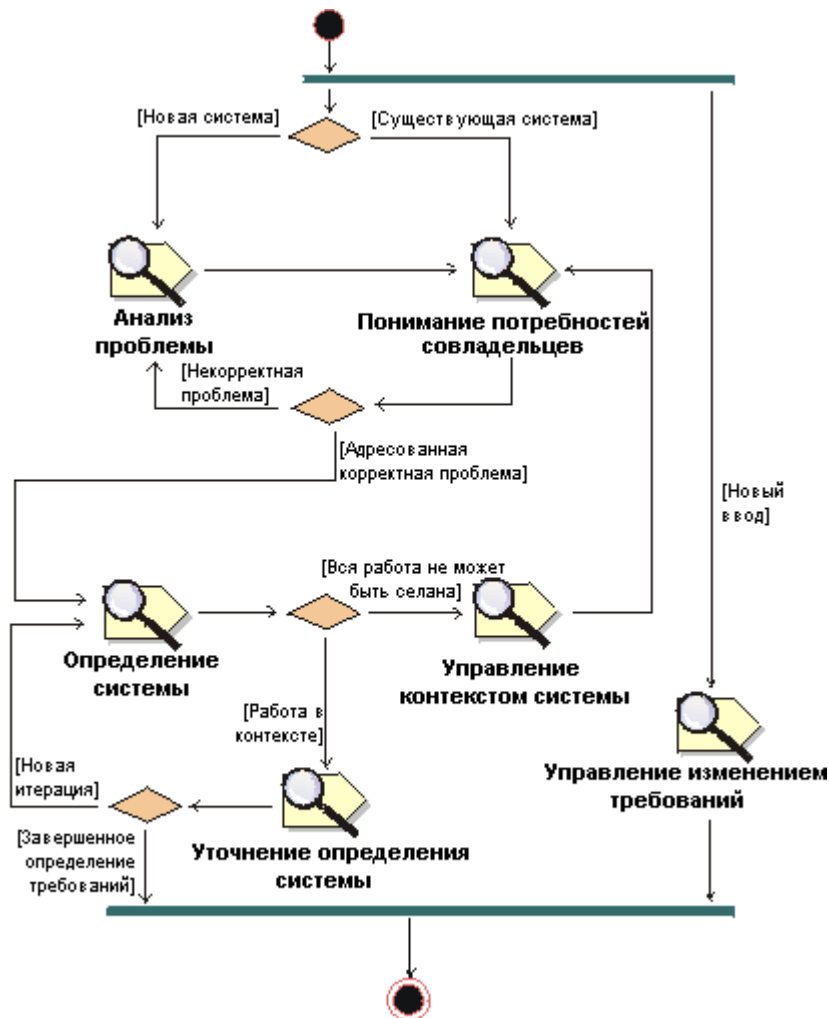
Для поддержки трассируемости требований Rational Unified Process предлагает использовать продукт Rational Requisite Pro (см. стр. 5-7).

Ключевым понятием Rational Requisite Pro в помощи для управления изменениями требований является понятие «сомнительной» связи трассируемости. Когда требование (или другой элемент трассируемости) изменяется с одного из концов связи трассируемости, все ссылки, связанные с этим требованием помечаются как «сомнительные». Это пометка возлагает на ответственного работника обзор изменений и определение того, должны ли измениться также и связанные объекты. Это понятие помогает также при анализе конфликтов потенциальных изменений.



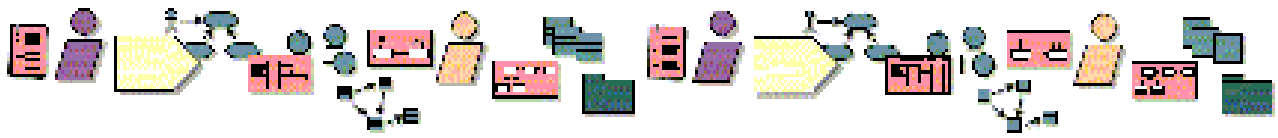
Краткий обзор потока работ

Для объяснения основного потока работ Требования интерактивная версия Rational Unified Process представляет следующую диаграмму действий:



Представление деталей потока работ Требования на диаграмме действий

Состояния действия (см. стр. 8-16) на этой диаграмме представляют детали потока работ (см. стр. 3-5).



Детали потока работ представляют ключевые навыки, которые должны применяться для эффективного управления требованиями. Детали Анализ проблемы и Понимание потребности совладельцев фокусируются на стадии Начало, а детали Определение системы и Уточнение определения системы – на стадии Уточнение. Детали Управление контекстом системы и Управление изменением требований выполняются непрерывно в течение всего проекта.

Детали потока работ представлены в порядке логической последовательности. Но они, как указано выше, применяются непрерывно в различном порядке, необходимом в проекте. На диаграмме действий они указаны в той последовательности, которая наиболее вероятна в первой итерации нового проекта.

Анализ проблемы

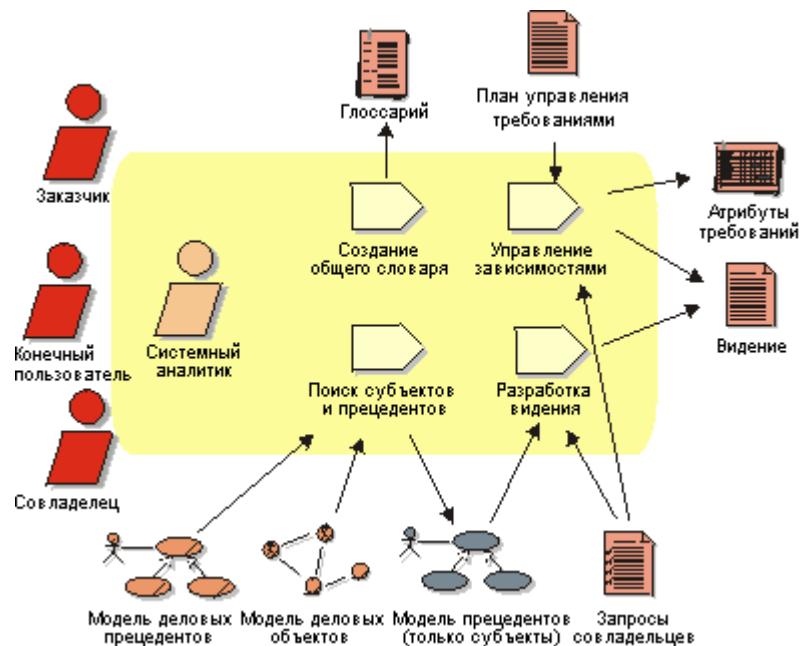
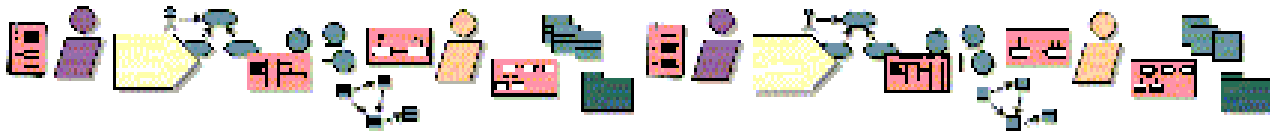


Диаграмма деталей потока работ Анализ проблемы

Целями этой части потока работ являются:

- Произвести для проекта документ Видение
- Договориться о возможностях и целях системы



Анализ проблемы выполняется для того, чтобы понять проблему, начальные потребности совладельцев и предложить решения высокого уровня. Это акт рассуждения и анализа с целью найти «проблему позади проблемы». В ходе анализа проблемы приобретает взаимное согласие о реальной проблеме и о том, кто является ее совладельцами. Кроме того, Вы определяете, какие из деловых перспектив ограничивают решение, а также деловые ограничения решений. Вы должны также проанализировать деловые прецеденты проекта так, чтобы получить хорошее понимание того, какие преимущества ожидаются от инвестиции в формируемую систему.

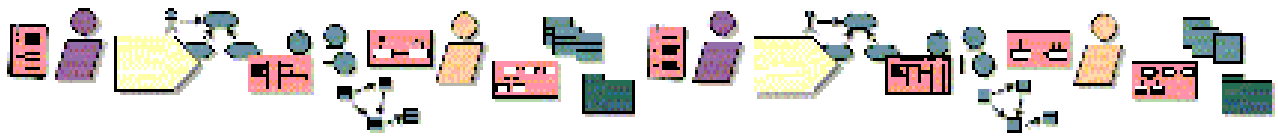
Эта часть потока работ может несколько раз повторяться в течение начала и раннего уточнения. Когда запросы от совладельцев будут поняты, тогда будут развиваться решения деловых процессов и технические решения.

Первичное действие в анализе проблемы – это разработка документа Видение, который идентифицирует высокоуровневое представление о создаваемой системе ее будущего пользователя или заказчика. В Видении начальные требования выражены как главные возможности, которыми должна обладать система, чтобы решить наиболее важные проблемы. Возможностям должны быть назначены атрибуты типа обоснования, относительного значения или приоритета, источника запроса и так далее, так, чтобы можно было начинать управлять зависимостями. По мере разработки Видения, системный аналитик идентифицирует пользователей и взаимодействующие системы - субъекты системы.

Если Вы разработали модель деловых прецедентов и модель деловых объектов, посмотрите еще раз раздел «Переход от деловых моделей к системным» на стр. 10-17.

Члены проектной группы, участвующие в анализе проблемы, должны иметь опыт общения с заказчиками и обнаружения проблемы позади проблемы. Конечно, знакомство с целевой технологией желательно, но не существенно. Требуется активное участие различных совладельцев проекта.

Для поиска проблемы позади проблемы Rational Unified Process рекомендует использовать метод мозговой атаки и диаграммы «рыбной кости» или диаграммы Парето. Все эти методы подробно изложены в разделе «Рабочие рекомендации» интерактивной версии Rational Unified Process.



Понимание потребностей совладельцев

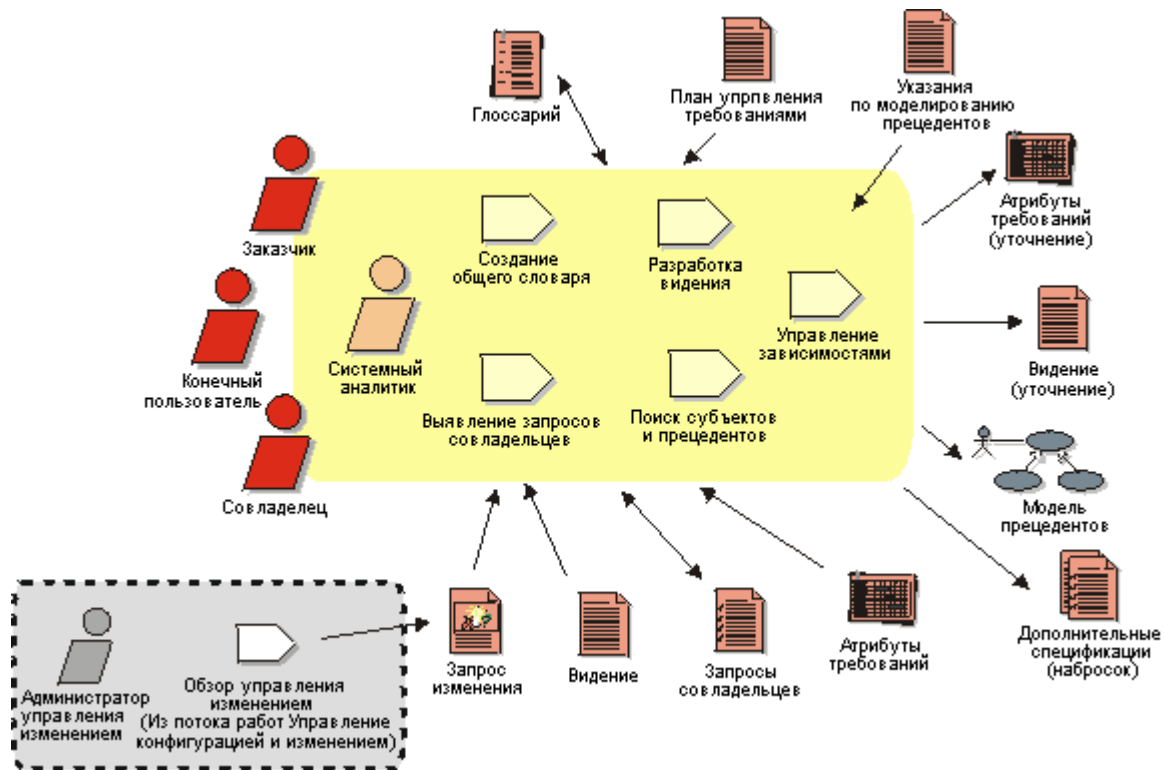
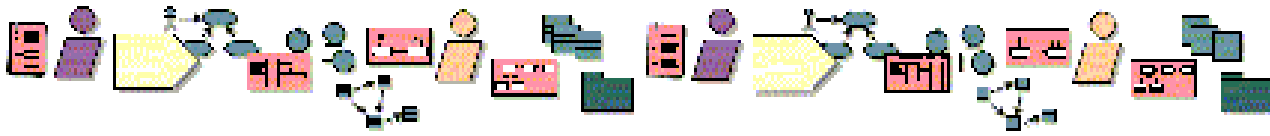


Диаграмма деталей потока работ Понимание потребностей совладельцев

Цель состоит в том, чтобы собрать и выявить информацию от совладельцев проекта. Собранные запросы совладельцев могут быть расценены как «список пожеланий», которые будут использоваться в качестве первичной информации для определения модели прецедентов, прецедентов и дополнительных спецификаций.

Как правило, это действие выполняется главным образом в итерациях стадий Начало и Уточнение, однако запросы совладельцев должны собираться в течение всего проекта, используя Запросы изменения в процессе Управления запросом изменения.

Требования происходят из многих источников, примерами которых могут быть заказчики, партнеры, конечные пользователи и специалисты предметной области. Вы должны знать, как лучше установить, какими должны быть эти источники и как получить к ним доступ, а также как лучше получить от них информацию. Если вы разрабатываете информационную систему, которую нужно использовать внутри вашей компании, Вы можете включить в вашу группу разработки людей с опытом конечного пользователя и знанием предметной области. Достаточно часто проводят обсуждения на уровне деловой модели. Если вы разрабатываете изделие на



продажу, Вы можете привлекать к работе ваших специалистов по маркетингу, чтобы лучше понять потребности заказчиков на рынке.

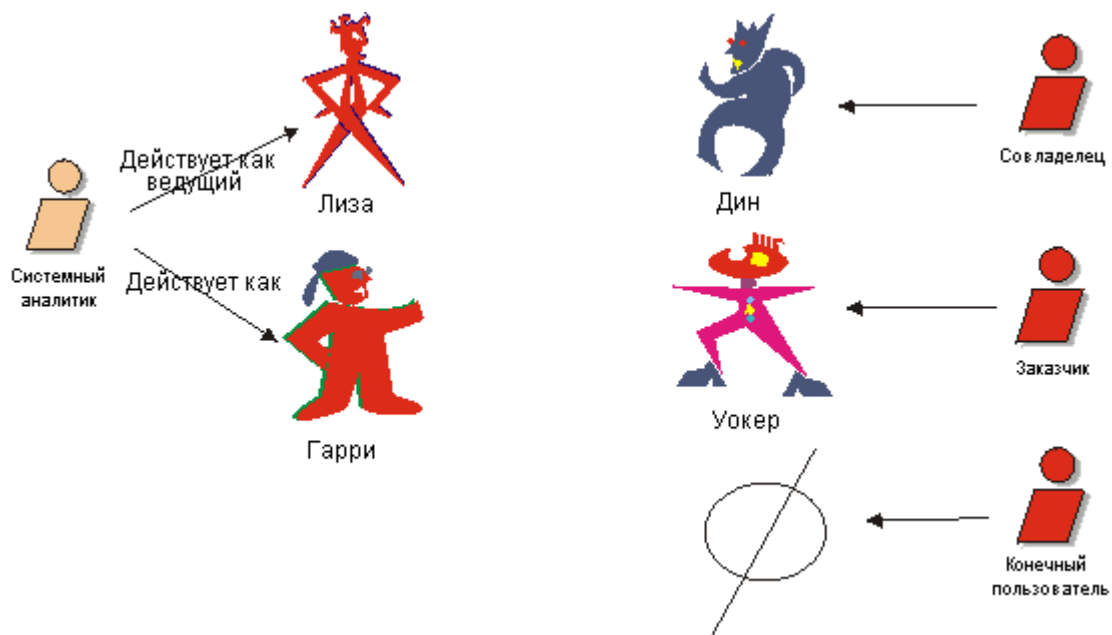
Первичные результаты - коллекции расположенных по приоритетам Запросов совладельцев, которые позволяют усовершенствовать документ Видение и лучше понять атрибуты требований. Кроме того, в течение этого потока работ Вы можете начать обсуждение системы в терминах ее прецедентов и субъектов. Другой важный выход - модифицированный Глоссарий терминов, который устанавливает общий словарь для членов группы.

Члены проектной группы, участвующие в этой части потока работ, должны иметь опыт общения с заказчиками и выявления информации. Конечно, знакомство с целевой технологией желательно, но не существенно.

Действия по выявлению информации могут происходить с использованием методов интервью, мозговой атаки, концептуального прототипирования, анкетных опросов и анализа конкурентоспособности.

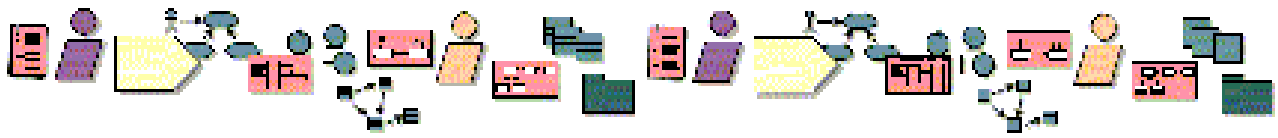
Пример

В стадии Начало проекта АТМ (автоматической кассовой машины) Лиза и Гарри действуют как системные аналитики.

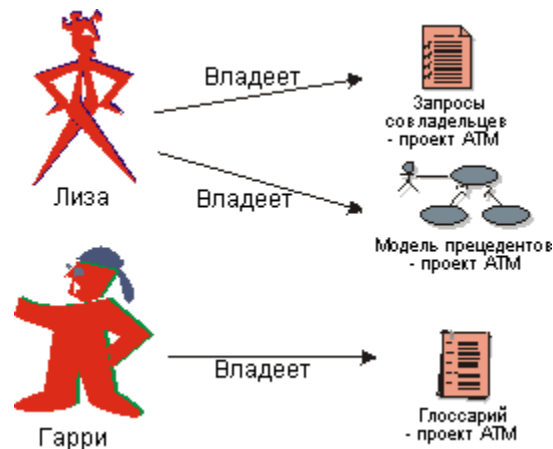


Участники деталей потока работ Понимание потребности совладельцев

Лиза назначена ведущим системным аналитиком, так как она имеет предшествующий опыт проведения различных семинаров по требованиям. Дин - один из поборников проекта АТМ,



и поэтому был приглашен на семинар как совладелец. Уокер представляет операторов АТМ, которые будут использовать новую систему. Они не приглашали специально конечных пользователей, но это не вызывало никакого беспокойства, так как все члены этой группы имели опыт использования банкоматов.



С этих пор Лиза и Гарри действуют как системные аналитики. Оба они отвечают за завершение артефактов этого шага. Лиза назначена владельцем документа Запросы совладельцев и модели прецедентов, а Гарри отвечает за Глоссарий.

Определение системы

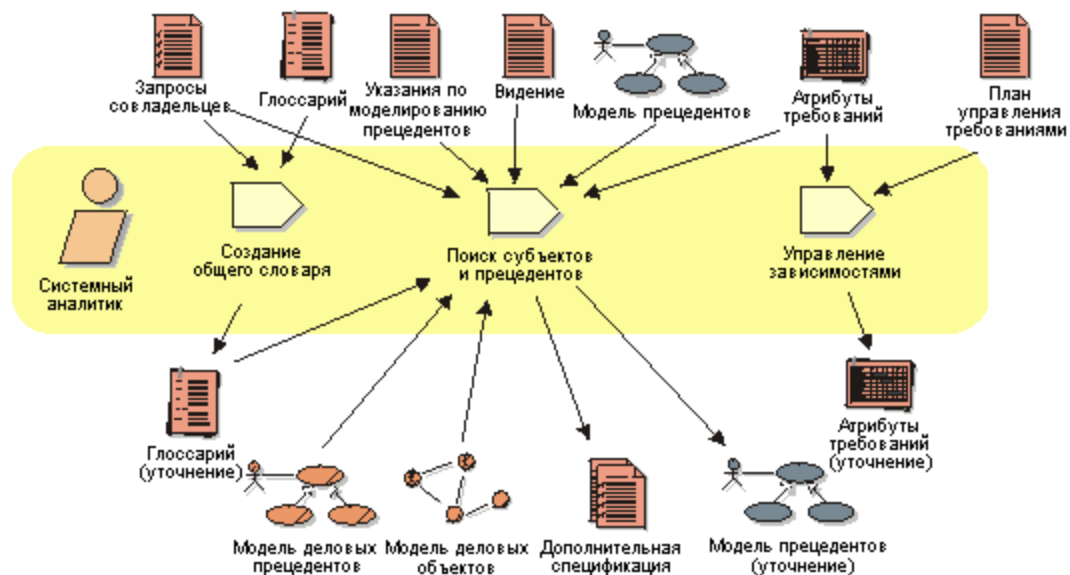
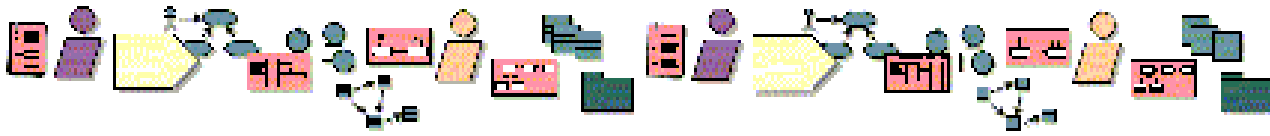


Диаграмма деталей потока работ Определение системы



Целями этой части потока работ являются:

- Добиться одинакового понимания системы всеми участниками проектной группы.
- Выполнить высокоуровневый анализ результатов сбора данных о потребностях совладельцев.
- Более формально документировать результаты анализа требований в моделях и документах.

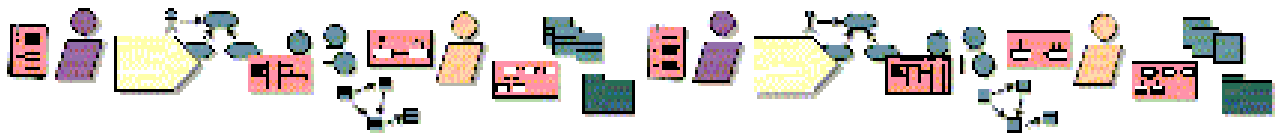
Как правило, эта часть потока работ выполняется только в итерациях стадий Начало и Уточнение.

Для определения системы способами трансляции и реорганизации определения запросов совладельцев должно быть сформировано выразительное описание системы. В начале определения системы решения строятся на том, что составляют требования, формат документирования, формальности языка, степень специфики требований (сколько и в какой степени детализации), запросы приоритетов и оценки усилий (две очень разные оценки, обычно назначаемые различными людьми в отдельных упражнениях), технические и управленческие риски и начальный контекст. Часть этих действий может включать начальные прототипы и модели проекта, непосредственно связанные с наиболее важными запросами совладельцев. Результатом определения системы является описание системы с использованием естественного языка и графического представления.

В этой работе участвуют все члены проектной группы.

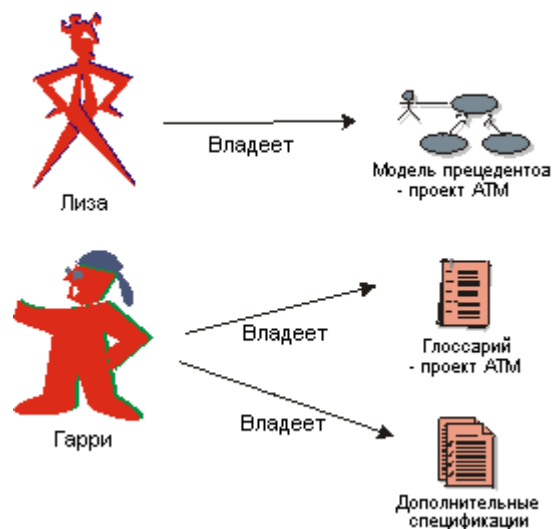
При определении системы могут применяться следующие типовые методы:

- Семинар по требованиям
- Семинар по прецедентам
- Иллюстрированный сценарий

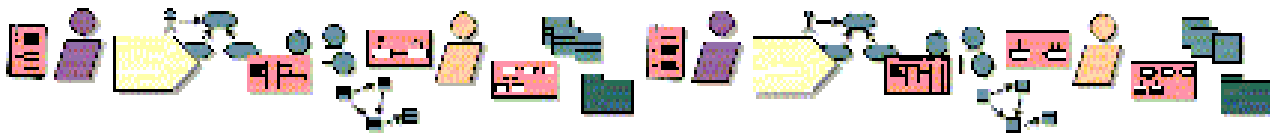


Пример

В стадии Начало Лиза и Гарри действуют как системные аналитики. На этом шаге они встречаются, чтобы проанализировать результаты своих встреч с совладельцами и заказчиками, и договориться о первых набросках модели прецедентов и Дополнительных спецификаций.



И далее Лиза и Гарри действуют как системные аналитики, они оба отвечают за завершение артефактов. Лиза - владелец модели прецедентов, а Гарри - владелец Глоссария и Дополнительных спецификаций.



Управление контекстом системы

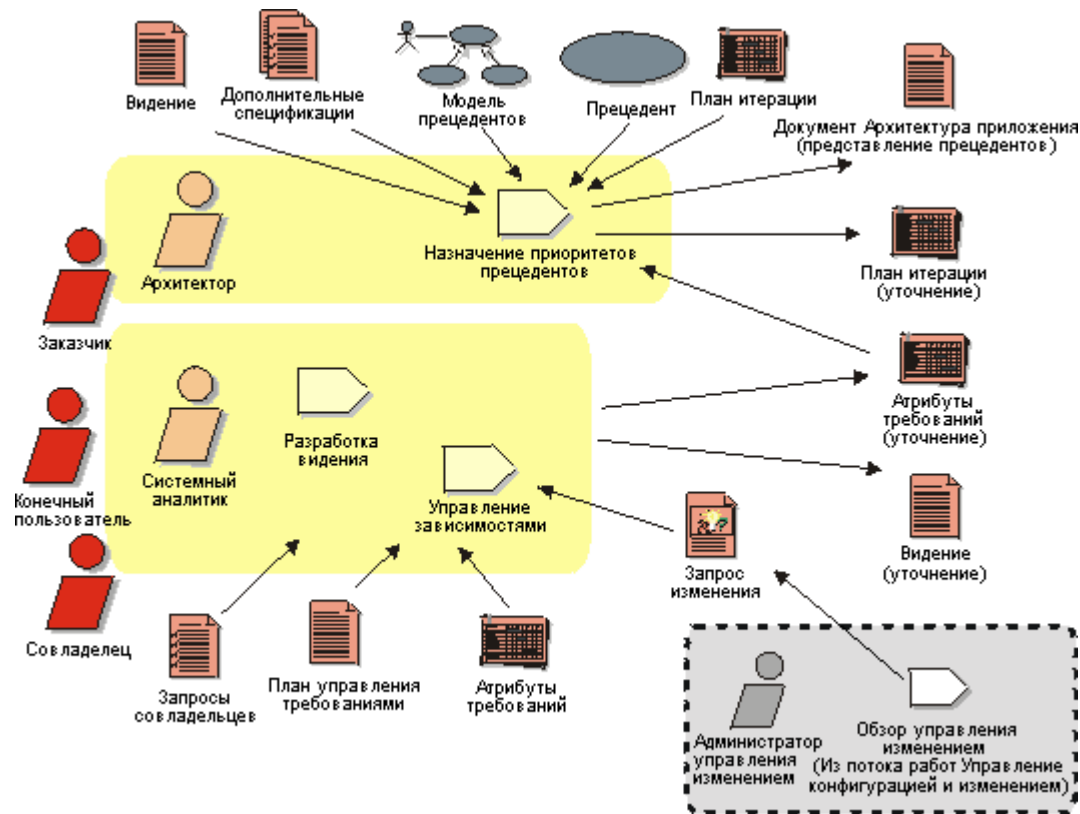
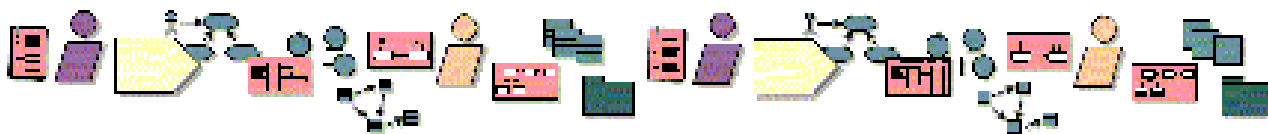


Диаграмма деталей потока работ Управление контекстом системы

Целями этой части потока работ являются:

- Определить исходную информацию для выбора требований, которые должны быть включены в текущую итерацию.
- Определить набор возможностей и прецедентов (или сценариев), которые представляют некоторые существенные, центральные функциональные возможности.
- Определить, какие аргументы требований и трассируемости необходимо поддерживать.



Управление контекстом – непрерывная деятельность, которая требует итерационной разработки в приращениях, и которая разбивает контекст проекта на меньшие, более управляемые части.

Для управления контекстом при принятии решений о включении или не включении требования особенно полезно использование атрибутов требования, типа приоритета, усилия и риска. Уделение внимания атрибутам больше, чем самим требованиям, помогает снизить накал страстей на переговорах с совладельцами, которые иначе могут перерасти в спор.

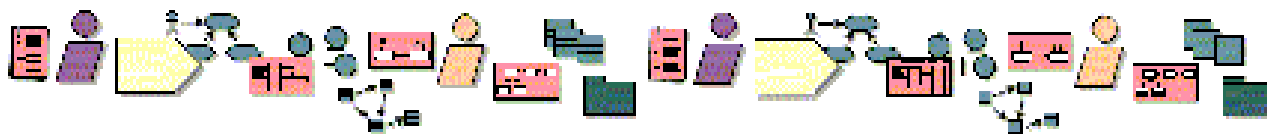
Хотя контекст проекта должен управляться непрерывно, лучшее понимание функциональных возможностей системы, полученных при идентификации большинства субъектов, прецедентов и дополнительных спецификаций, позволяет системному аналитику более точно устанавливать атрибуты требований, и дает возможность архитектору идентифицировать архитектурно существенные прецеденты.

Исходным для управления контекстом, не участвующим в других частях потока работ Требования, является План итерации, разрабатываемый параллельно в потоке работ Управление проектом. Контекст проекта, определенный в этой части потока работ, будет иметь значительное воздействие на План итерации, поскольку самые высокие элементы риска в пределах контекста будут запланированы в ранних итерациях. Другими важными результатами управления контекстом являются начальный вариант документа Архитектура приложения и усовершенствованное Видение, которое отражает лучшее понимание функциональных возможностей системы и проектных ресурсов со стороны системных аналитиков и ключевых совладельцев.

В этой части потока работ должны участвовать все члены группы архитектуры.

Для руководителей группы очень полезно иметь опыт в ведении переговоров, а также иметь лоббиста проекта в организации и на стороне заказчика. Лоббисты изделия/проекта должны иметь организационную власть отказаться от расширений контекста вне доступных ресурсов или выделить дополнительные ресурсы для обеспечения расширения контекста.

Группа архитектуры должна провести коллективное обсуждение того, как лучше расположить требования по приоритетам.



Уточнение определения системы

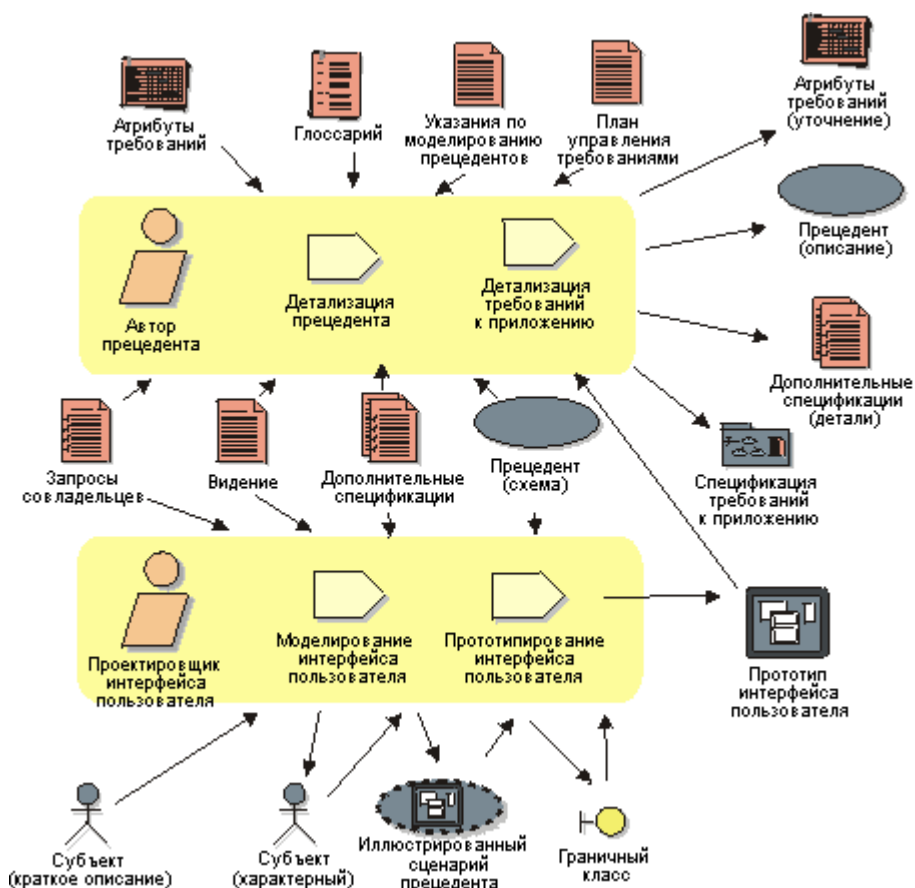
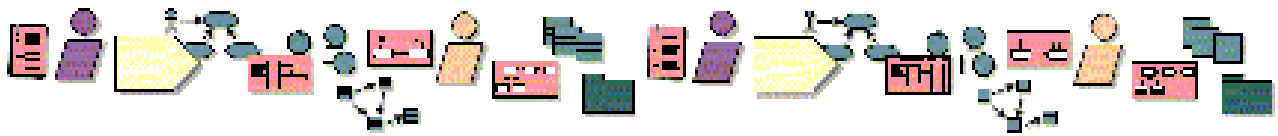


Диаграмма деталей потока работ Уточнение определения системы

Цель этой части потока работ состоит в дальнейшем совершенствовании требований, чтобы:

- Подробно описать поток событий прецедента.
- Детализировать Дополнительные спецификации.
- Моделировать и прототипировать интерфейс пользователя.

Детальное определение системы должно быть представлено таким образом, чтобы ваши совладельцы могли понимать его и соглашаться или не соглашаться с ним. Оно требует не только освещения функциональных возможностей, но также и соответствия всем юридическим



или нормативным требованиям, применимости, достоверности, эффективности, пригодности и надежности 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. Люди могут быть увлечены, но они не дадут хорошей отдачи, так как у них нет полного понимания. Вы должны приложить максимум усилий для понимания аудиторией тех документов, которые Вы производите для описания системы. Часто может возникать потребность производить различные виды описаний 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.

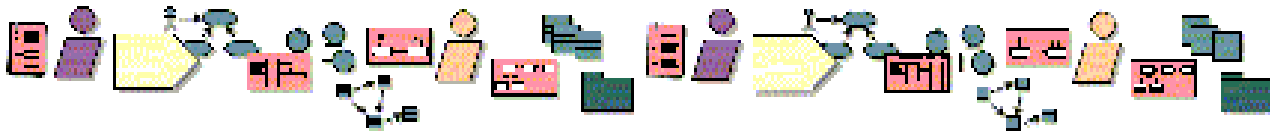
Другой компонент детального определения системы - это констатация того, как система должна быть испытана. Планы проведения и определения предметов испытания говорят о том, какие возможности системы будут проверены.

Усовершенствование системы начинается со схем прецедентов, кратких описаний субъектов и уточненного понимания контекста проекта, отраженного в Видении в форме списка расположенных по приоритетам возможностей, которые, как предполагается, являются достижимыми в рамках имеющихся ресурсов и сроков. Выход этого потока работ - более глубокое понимание функциональных возможностей системы, выраженных в детальных описаниях прецедентов, исправленных и детализированных Дополнительных спецификациях и в элементах интерфейса пользователя.

Авторы прецедентов должны быть квалифицированы в описании прецедентов и иметь хорошее представление о прикладной области. Часто полезно включать в вашу группу авторов прецедентов людей, которые позже будут действовать как проектировщики и тестировщики.

В больших проектах проектировщики интерфейса пользователя – это часто отдельная группа людей, сосредоточенная только на разработке интерфейса пользователя.

Авторы прецедентов и проектировщики интерфейса пользователя должны работать вместе, чтобы совместно определять детальные требования к системе. Хотя многие действия могут выполняться по отдельности, необходимо использовать сквозной контроль, чтобы убедиться, что элементы не противоречат друг другу.



Управление изменением требований

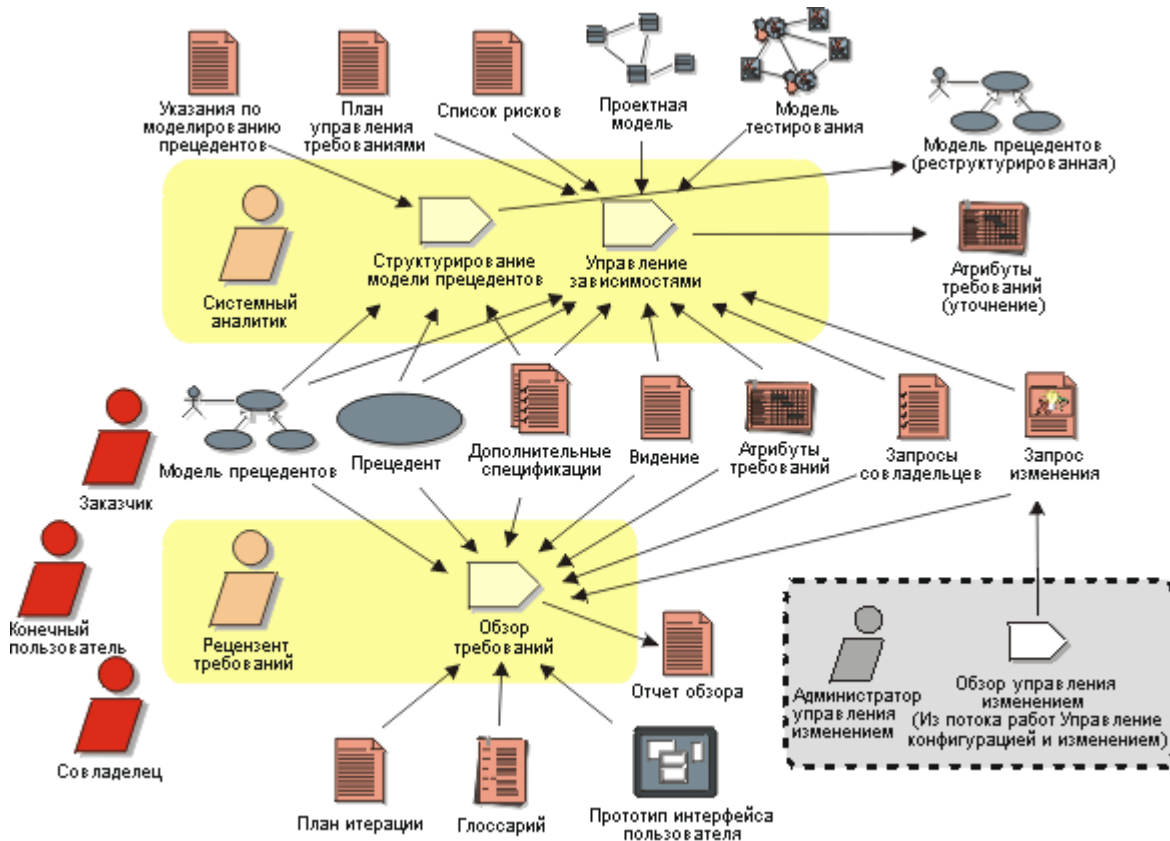
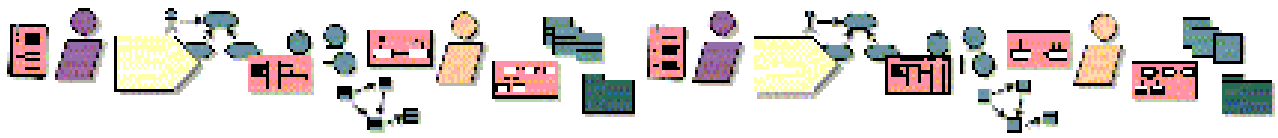


Диаграмма деталей потока работ Управление изменением требований

Целями этой части потока работ являются:

- Оценить формально представленные запросы изменения и определить их противоречия с существующим набором требований.
- Структурировать модель прецедентов.
- Установить соответствующие атрибуты требований и трассируемости.
- Формально убедиться, что результаты потока работ Требования соответствуют взглядам заказчика системы.

Независимо от того, насколько Вы осторожны при определении ваших требований, всегда найдутся вещи, требующие изменения. Что делает изменяющиеся требования сложными для управления? Изменение требования означает не только то, что должно быть потрачено большее или меньшее количество времени на реализацию новой специфической возможности, но также и то, что изменение в одном требовании может вступить в конфликт с другими требованиями.



Вы должны убедиться, что Вы придаете вашим требованиям структуру, которая является гибкой к изменениям, и что Вы используете ссылки трассируемости для представления зависимостей между требованиями и другими артефактами жизненного цикла разработки. Управление изменением включает действия подобные установлению опорной линии, определению зависимостей, которые важно проследить, устанавливая трассируемость между связанными предметами и управлением изменением.

Изменение требований естественно приводит к противоречиям с моделью, произведенной в потоке работ Анализ и проектирование, а также с моделью тестирования, созданной как часть потока работ Испытание. Связи Трассируемости между требованиями, идентифицированные в действии Управление зависимостями этого и других потоков работ - ключ к пониманию этих противоречий.

Другая важная концепция - отслеживание хронологии требований. Фиксируя характер и обоснование изменений требований, рецензенты (в данном случае эту роль играет любой член группы разработки) воспринимают информацию, необходимую для того, чтобы ответить на изменение должным образом.

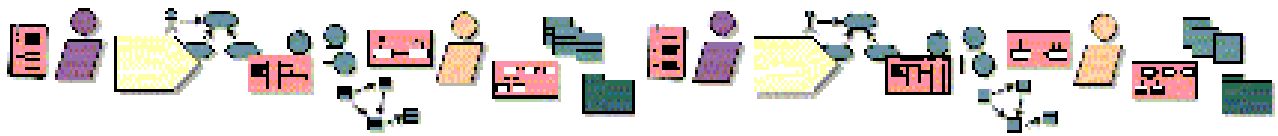
Привлеките расширенную группу (совладельцев, представителей заказчика, специалистов по проблемной области и других). Старайтесь эффективно управлять ресурсами рецензентов – не привлекайте всю расширенную группу, если Вы не можете гарантировать, что это даст положительный эффект для проекта.

Общая квалификация членов расширенной группы должна охватывать хорошее знание прикладной области, технические трудности проекта, а также навыки в управлении требованиями и моделировании прецедентов.

Основная группа разработки должна провести несколько раундов внутреннего сквозного контроля, чтобы прояснить явные несообразности прежде, чем работа будет формально проинспектирована и подвергнута экспертизе расширенной группой.

Вы должны разделить материал так, чтобы группа не делала обзор всего сразу. Совещание обзора не должно занимать больше одного дня (например, проведите отдельно обзор интерфейса пользователя).





Поток работ «Требования» (продолжение)

Артефакты управления требованиями

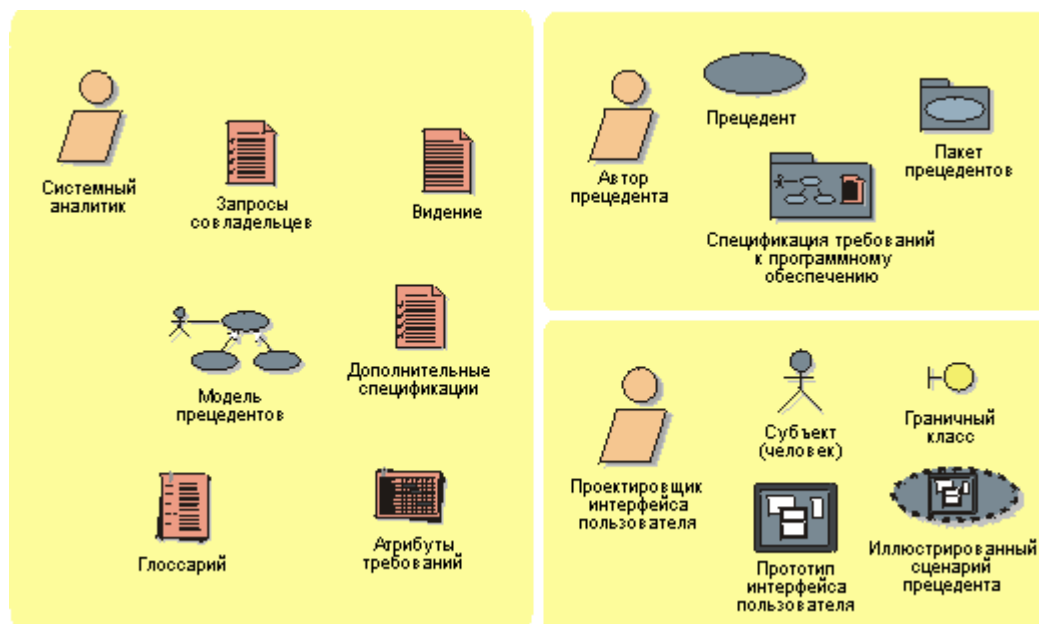
Все артефакты управления требованиями уже упоминались в предыдущем разделе “Краткий обзор потока работ”. Сейчас мы перейдем к более подробному описанию некоторых из них.

Хотелось бы организовать изложение в некотором логическом порядке, но ...

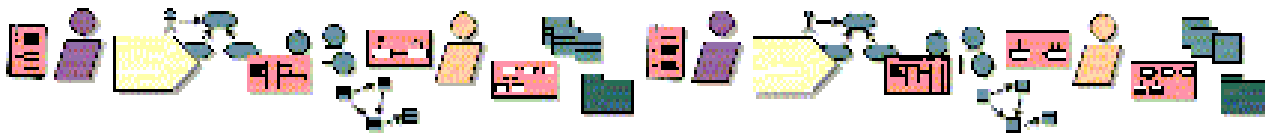
При внимательном просмотре диаграмм, иллюстрирующих поток работ, Вы могли обратить внимание, что одни и те же артефакты много раз встречаются на диаграммах и являются исходными и выходными для различных действий потока работ. Артефакты “живут активной жизнью” и изменяются (уточняются) в течение всего цикла разработки проекта. Артефакты могут разрабатываться (уточняться) параллельно, и могут быть зависимы друг от друга.

Давайте упорядочим рассмотрение артефактов следующим образом:

- Сгруппируем артефакты по работникам, ответственным за их создание и поддержание в актуальном состоянии:



Артефакты потока работ “Требования” и работники, ответственные за их создание и поддержание в актуальном состоянии



- В каждой группе будем рассматривать артефакты в том порядке, в котором они, скорее всего, будут создаваться (инициироваться) в совершенно новом проекте, начинаемом с чистого листа группой, впервые встретившейся с этой предметной областью.

Работник: Системный аналитик

Системный аналитик ведет и координирует выявление требований и моделирование прецедентов, выделяя функциональные возможности и границы системы. Например, устанавливая, какие субъекты и прецеденты существуют и как они взаимодействуют.

Артефакт: Глоссарий

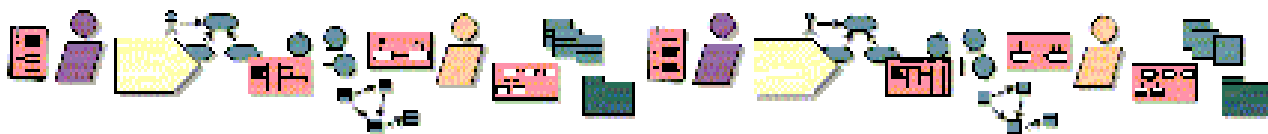
Краткий обзор

В потоке работ Требования Вы должны определить общий словарь, используя наиболее общие термины из прикладной области. Далее Вы должны постоянно использовать общий словарь во всех текстовых описаниях системы, особенно в описаниях прецедентов. Таким образом, Вы сохраняете текстовые описания непротиворечивыми, и избегаете недоразумений среди участников проекта относительно использования и значения терминов. Вы должны документировать словарь в Глоссарии.

Для поиска общих терминов в прикладной области рассмотрите термины, используемые в требованиях и употребляемые в группе разработки системы. Сосредоточьтесь на терминах, описывающих следующие концепции:

- Деловые объекты, представляющие концепции, используемые в повседневной работе организации или в оперативной среде будущей системы. Во многих случаях список концепций этого вида уже существует.
- Реальные объекты, которые должна знать система. Это объекты естественного происхождения, такие как автомобиль, зажим, баллон, самолет, пассажир, заповедник или счет.
- События, которые должна знать система. Это может быть момент времени или хронологический инцидент, о котором система должна знать, типа назначенной встречи или факта появления ошибки.

Каждый термин обычно описывается как существительное с определением. Термины должны быть в единственном числе: «заказ» и «задание», а не «заказы» и «задания». Все заинтересованные стороны должны договориться об определениях терминов.



Синхронизация

Общий словарь, определенный в Глоссарии, должен постоянно использоваться во всех текстовых описаниях системы и при переговорах с совладельцами.

В некоторых проектах Глоссарий является единственным артефактом, описывающим деловую сферу проекта. Это случается тогда, когда ни деловое моделирование, ни моделирование предметной области не выполняется.

Поэтому очень важно как можно раньше договориться об общей терминологии.

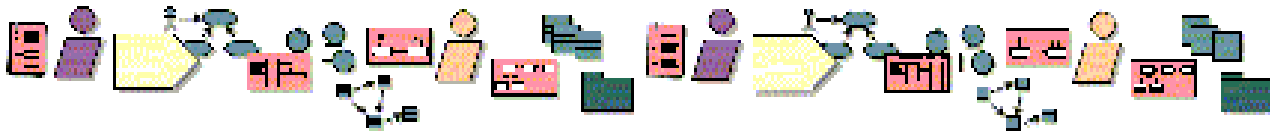
Глоссарий создается в стадии Начало, и к концу стадии Уточнение, в основном, стабилизируется.

Создание и уточнение Глоссария происходит в результате следующих действий:

- Фиксация общего словаря
- Назначение приоритетов прецедентов
- Рецензирование модели деловых прецедентов
- Обзор требований

Глоссарий предоставляет исходную информацию для следующих действий:

- Архитектурный анализ
- Детализация делового прецедента
- Детализация прецедента
- Детализация требований к программному обеспечению
- Поиск субъектов и прецедентов
- Поиск деловых субъектов и прецедентов
- Поиск деловых работников и сущностей
- Рецензирование модели деловых объектов
- Рецензирование модели деловых прецедентов
- Обзор требований
- Структурирование модели деловых прецедентов
- Структурирование модели прецедентов
- Анализ прецедента



Инструментальная поддержка

Интерактивная версия Rational Unified Process содержит шаблон Microsoft Word для документа Глоссарий.

Шаблон Глоссарий.dot представлен в качестве отдельного приложения к этому выпуску.

Однако Вы можете не ограничиваться традиционным использованием этого шаблона для составления документа Word. Rational RequisitePro существенно расширяет возможности работы со словарем.

Rational RequisitePro позволяет выполнять следующие операции:

1. Создание документа Глоссарий.
2. Выполнение документа Глоссарий.
3. (Не обязательно) Маркирование требований в документе Глоссарий.

Рассмотрим несколько подробнее последнюю возможность.

Глоссарий не содержит требований по существу; однако, могут быть случаи, когда некоторые термины Глоссария можно использовать как названия для классов, особенно классов сущностей. Отмечая эти термины как требования, Вы можете автоматически создавать классы того же названия в Rational Rose с использованием Rational Synchronizer.

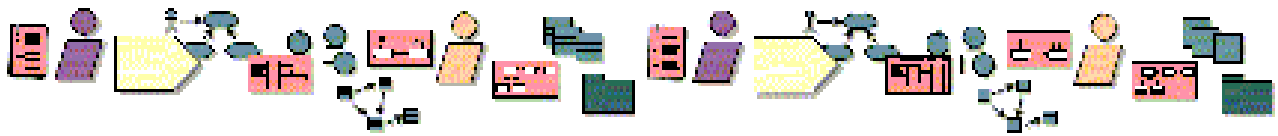
Вы можете также захотеть отслеживать факты переопределения терминов в Глоссарии и модифицировать некоторые документы, чтобы отразить эти изменения. Маркировка таких терминов как требований позволит Вам позже создать связи трассируемости между терминами Глоссария и возможностями изделия или требованиями к системе, которые должны быть перефразированы после того, как определение термина Глоссария было модифицировано.

Любой термин Вы можете пометить как требование типа Glossary Requirement Type (TERM). Это требование будет сохранено в базе данных Rational RequisitePro. Впоследствии Вы сможете установить нужные связи и трассировки с другими объектами требований.

Контрольные точки

Как и для других артефактов, для Глоссария Rational Unified Process предлагает ряд вопросов, положительный ответ на которые свидетельствует о высоком качестве артефакта. Это:

- Все ли термины имеют ясное и краткое определение?
 - Каждый ли термин глоссария включен где-нибудь в описания прецедентов? Если нет, это может означать, что отсутствует нужный прецедент или что существующие прецеденты не завершены. Более вероятно, что термин не включен потому, что в этом нет необходимости. В этом случае его нужно удалить.
 - Согласованно ли используются термины в кратких описаниях субъектов и прецедентов?
-



- Означает ли термин одно и то же во всех прецедентах?

Пример артефакта: Введение в RUP. Глоссарий

Рабочая группа проекта OPFS Classics, Ink. (см. стр. 9-1) разработала Глоссарий, в котором определены используемые в проекте термины. Но, я думаю, знакомиться с ними нет практического смысла. Лучше взять пример поближе к интересующей нас предметной области.

В качестве примера предлагается рассмотреть фрагмент словаря терминов, представленного в глоссарии интерактивной версии Rational Unified Process. Я надеюсь, что этот фрагмент будет расширяться, и к моменту завершения серии статей охватит весь глоссарий.

Документ Глоссарий.doc представлен в качестве отдельного приложения к этому выпуску.

Артефакт: Видение

Краткий обзор

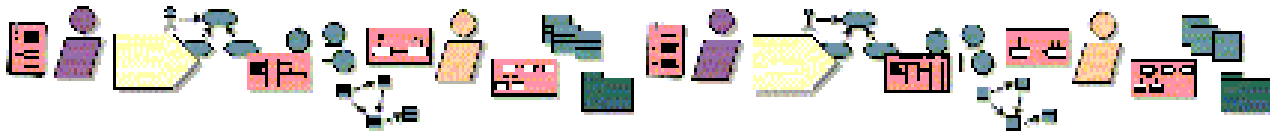
Документ Видение фиксирует требования и проектные ограничения очень высокого уровня, давая читателю понимание разрабатываемой системы. Он обеспечивает исходные данные для процесса рассмотрения проекта, и поэтому тесно связан с документом Деловые обстоятельства (см. стр. 6.3).

Видение сообщает основные «зачем и как», связанные с проектом, и критерии, на основании которых должны быть приняты все будущие решения.

Видение предоставляет основной исходный материал для последующих договоренностей о большей детализации технических требований. Может существовать также и формальная спецификация требований.

При разработке Видения выполняются следующие шаги:

- Достижение соглашения по решаемой проблеме
- Идентификация совладельцев
- Определение границ системы
- Идентификация ограничений, которые будут наложены на систему
- Формулирование постановки задачи
- Определение возможностей системы
- Оценка результатов



1. Достижение соглашения по решаемой проблеме

Один из самых простых способов достигнуть соглашения по определению проблемы – записать ее и посмотреть, все ли с этим согласны.

Спросите Ваших заказчиков или их представителей: В чем состоит проблема?

Затем повторите вопрос несколько иначе: Почему возникла эта проблема? В чем состоит проблема в действительности?

Не принимайте первую формулировку проблемы. Продолжайте спрашивать «почему?» Отыщите коренные причины или «проблему позади проблемы». Реальная проблема часто скрывается за тем, что воспринимается как проблема.

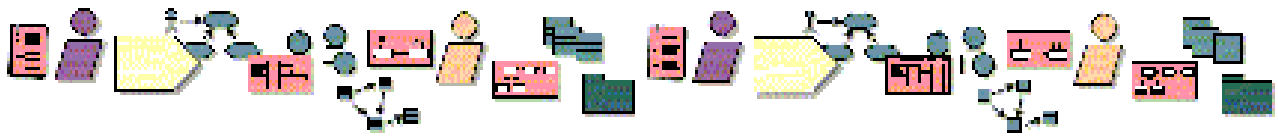
Иногда интервьюируемые могут так сосредоточиться на предполагаемом решении, что это заставит их сразу сформулировать основную проблему. В таком случае может быть полезно исследовать выгоды от этого решения, а затем попытаться найти проблемы, решаемые таким образом. Затем Вы можете исследовать, действительно ли эти проблемы являются «реальными» проблемами в организации.

Простой метод поиска проблемы позади проблемы – использование диаграмм «рыбьего скелета» или Парето. Оба метода описаны в разделе Рабочие рекомендации интерактивной версии Rational Unified Process.

2. Идентификация совладельцев

В зависимости от степени знакомства группы разработки с проблемной областью, идентификация совладельцев может быть шагом тривиальным или нетривиальным. Часто это просто интервью у людей, принимающих решения, потенциальных пользователей и других заинтересованных сторон. Полезно задать им следующие вопросы:

- Кто является пользователями системы?
- Кто является выгодным покупателем системы?
- На что еще будут воздействовать результаты, которые производит система?
- Кто оценит и благословит систему, когда она будет поставлена и развернута?
- Имеются ли любые другие внутренние или внешние пользователи системы, чьи потребности должны быть удовлетворены?
- Кто будет лоббировать новую систему?
- Имеется ли кто-либо еще из заинтересованных лиц?
- Хорошо, а имеется ли кто-либо еще?



3. Определение границ системы

Граница системы определяет границу между системой и реальным миром, который окружает систему. Информация, в форме вводов и выводов, циркулирует взад и вперед между системой и пользователями, которые находятся вне системы. Все взаимодействия с системой происходят через интерфейсы между системой и внешним миром.

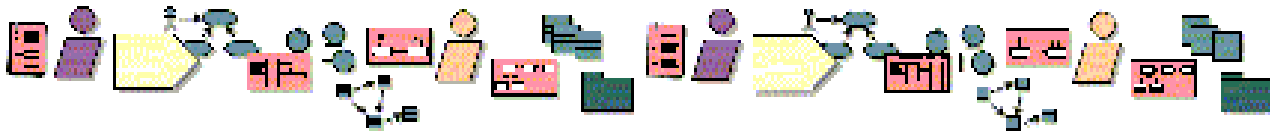
Во многих случаях границы системы очевидны. Например, границы пользователя, который персонально работает на Windows 2000®, относительно хорошо определены. Имеется только один пользователь и одна платформа. Интерфейсы между пользователем и прикладной программой состоят из диалогов интерфейса пользователя, к которым пользователь обращается, чтобы ввести информацию в систему, и каких-либо выводимых отчетов и путей связи, которые система использует для документирования или передачи итоговой информации.

Обычно для того, чтобы определить и описать границы системы, очень полезно использовать субъекты. О субъектах мы будем говорить в одном из следующих выпусков.

4. Идентификация ограничений, которые будут наложены на систему

Существует множество источников ограничений, которые должны быть рассмотрены. Ниже представлен список потенциальных источников ограничений и вопросов, которые должны быть выяснены:

- Политический: Имеются ли внутренние или внешние политические проблемы, которые затрагивают потенциальные решения? Межведомственные проблемы?
- Экономический: Какие финансовые или бюджетные ограничения применяются? Учитывается ли стоимость проданных товаров или изделий? Лицензирование выпусков?
- Среды: Имеются ли ограничения или регуляторы среды? Правовые? Другие стандарты, которыми мы ограничены?
- Технический: Ограничены ли мы в нашем выборе технологий? Вынуждены ли мы работать в рамках существующих платформ или технологий? Запрещено ли использование новых технологий?
- Выполнения: Определен ли график? Ограничены ли мы существующими ресурсами? Можем ли мы использовать внешнюю рабочую силу? Можем ли мы расширить ресурсы? Временными работниками? Постоянными?
- Системный: Решение должно базироваться на наших существующих системах? Мы должны поддержать совместимость с существующими решениями? Какие операционные системы и среды должны поддерживаться?



5. Формулирование постановки задачи

Полным составом группы поработайте над таблицами и заполните следующий шаблон для каждой проблемы, которую Вы идентифицировали:

Проблема: <описание проблемы>

затрагивает: <совладельцы, которых затрагивает проблема>.

Воздействие: <как воздействует проблема>.

Успешное решение: <список некоторых ключевых выгод от успешного решения>.

Цель этого шаблона состоит в том, чтобы помочь Вам отличить решения/ответы от проблем/вопросов.

Пример:

Проблема: несвоевременное и неверное решение вопросов послепродажного обслуживания

затрагивает: наших заказчиков, представителей поддержки заказчика и специалистов по обслуживанию оборудования.

Воздействие: неудовлетворенность заказчика, ощутимый недостаток качества, недовольство пользователей и потери дохода.

Успешное решение: обеспечит в реальном времени доступ к базе данных для устранения неисправности представителями поддержки и облегчит своевременную отправку специалистов по обслуживанию оборудования только туда, где действительно необходима их помощь.

6. Определение возможностей системы

Основываясь на выгодах, перечисленных в ваших постановках проблем, разработайте список возможностей, которые Вы хотите иметь в системе. Опишите их кратко, и дайте им атрибуты, чтобы помочь определить их общее состояние и приоритет в проекте. Подробнее об атрибутах см. Артефакт: Атрибуты требований в одном из следующих выпусков.

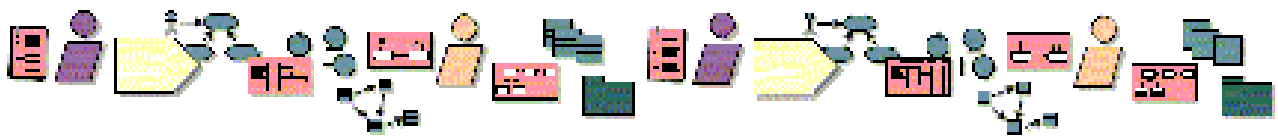
7. Оценка результатов

На этом шаге Вы должны проконтролировать документ Видение, чтобы проверить, как идет ваша работа, но не рассматривать его подробно. Ознакомьтесь с контрольными точками для документа Видение ниже в этом разделе.

Синхронизация

Документ Видение создается как можно раньше в стадии Начало и используется как основание для написания документа Деловые обстоятельства (см. стр. 6-3) и первого наброска списка рисков (см. стр. 6-4).

Документ Видение служит исходным для моделирования прецедентов, он модифицируется и поддерживается как отдельный артефакт на протяжении всего проекта.

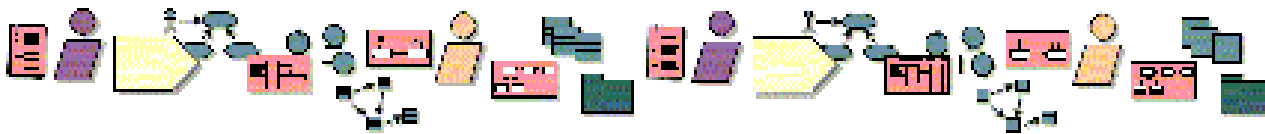


Создание и уточнение Видения происходит в результате следующих действий:

- Оценка итерации
- Разработка Видения
- Управление зависимостями
- Обзор требований

Видение предоставляет исходную информацию для следующих действий:

- Фиксация общего словаря
- Составление плана разработки программного обеспечения
- Определение организации проекта и укомплектование персоналом
- Детализация делового прецедента
- Детализация прецедента
- Детализация требований к программному обеспечению
- Изложение деловых обстоятельств
- Разработка плана итерации
- Разработка плана измерений
- Разработка плана приемки изделия
- Разработка плана обеспечения качества
- Разработка материалов для обучения
- Выявление запросов совладельцев
- Поиск субъектов и прецедентов
- Поиск деловых субъектов и прецедентов
- Поиск деловых работников и сущностей
- Управление зависимостями
- Моделирование интерфейса пользователя
- Назначение приоритетов прецедентов
- Рецензирование оценки проекта
- Рецензирование плана проекта
- Обзор требований
- Написание плана СМ



■ Написание примечаний к выпуску

Инструментальная поддержка

Так же, как и в случае с артефактом Глоссарий, интерактивная версия Rational Unified Process содержит шаблон Microsoft Word для документа Видение.

Шаблон Видение.dot представлен в качестве отдельного приложения к этому выпуску.

Кроме того, если у Вас есть Rational RequisitePro, Вы можете выполнять следующие операции:

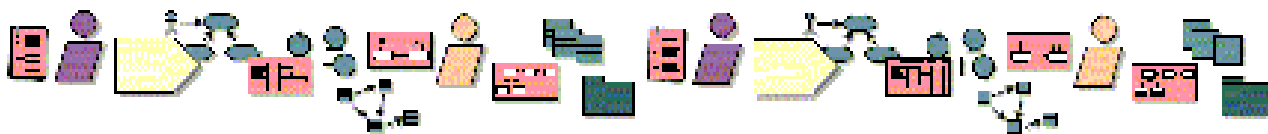
1. Создание документа Видение.
2. Выполнение документа Видение.
3. Маркирование требований в документе Видение.

Рассмотрим последнюю операцию.

RequisitePro позволяет создавать требования из возможностей, выделенных в разделе Возможности изделия. Это даст Вам возможность управлять атрибутами требований в матрице атрибутов, и поможет проследить связи трассируемости между возможностями изделия и требованиями к программе.

Для создания требования в документе Видение, если у Вас есть RequisitePro, сделайте следующее:

1. В разделе Возможности изделия документа Видение выберите текст, который определяет требование.
2. В рабочей области Word нажмите RequisitePro > Requirement > Create. Появляется диалоговое окно Requirement.
3. Выберите тип требования возможности (FEAT).
4. Нажмите на закладку Attributes и установите атрибут приоритета (Priority) в одно из предопределенных значений: высокий (High), средний (Medium) или низкий (Low). Очень полезно записывать приоритеты совладельца при выполнении Действия: Выявление запросов совладельцев.
5. Перейдите к атрибуту Origin и установите инициатора этой конкретной возможности. Это поможет Вам отслеживать совладельцев, предложивших реализовать определенные возможности изделия, способствует контактам с этими совладельцами для разрешения неясных вопросов или переговоров. Очень полезно записывать названия совладельца при выполнении Действия: Выявление запросов совладельцев.
6. Выберите атрибут Rationale и напечатайте причину для возможности. При работе над этой возможностью Вам будет легче сосредоточиться на выполнении назначенной потребности.



7. Нажмите ОК.
8. Повторите эти шаги для каждой возможности изделия, выделенной в разделе Возможности изделия документа Видение.

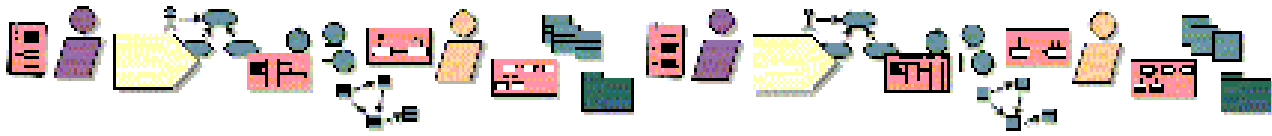
Для просмотра требований и атрибутов:

1. На панели инструментов нажмите Window > Show Views.
2. В рабочей области Views нажмите File > New View.
3. Выберите тип представления Attribute Matrix и тип требования FEAT, который Вы использовали в вашем документе Видение.
4. Нажмите ОК. Вы можете просматривать ваши требования возможностей и значения их атрибутов.

Контрольные точки

Если Вы сможете ответить «Да» на все перечисленные ниже вопросы, Вы разработали хороший документ Видение:

- Полностью ли исследованы «проблемы позади проблемы?»
- Правильно ли сформулирована постановка задачи?
- Является ли список совладельцев полным и правильным?
- Все ли согласились с определением границ системы?
- Если границы системы были выражены субъектами, все ли субъекты определены и правильно описаны?
- Достаточно ли исследованы ограничения, которым подчиняется система?
- Все ли виды ограничений учтены, например политические, экономические, окружающей среды и т.п?
- Все ли главные возможности системы идентифицированы и определены?
- Все ли проблемы будут разрешены при реализации идентифицированных возможностей?
- Совместимы ли возможности с идентифицированными ограничениями?



Пример артефакта: Система обработки и выполнения заказов.

Видение

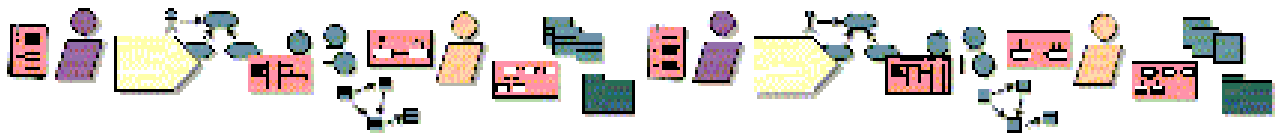
Рабочая группа проекта OPFS Classics, Ink. (см. стр. 9-1) разработала документ Видение. Данный документ представлен в качестве отдельного приложения к этому выпуску (см. файл OPFS_Vision.doc).

При рассмотрении документа нужно учитывать следующее:

Это всего лишь учебный пример. Не будьте к нему слишком строгим! Он предназначен для того, чтобы посмотреть, что и как принято писать в этом документе.

Этот документ исполнялся без использования Rational RequisitePro, и, следовательно, не имеет действующих меток требований. Эти метки имитируются двойным подчеркиванием текста требований (как принято в RequisitePro). В RequisitePro требования помечаются, кроме того, номерами (например, FEAT1.4).

В следующем выпуске мы продолжим обсуждение артефактов требований.



Поток работ «Требования» (продолжение 2)

Артефакт: Запросы совладельцев

Краткий обзор

Артефакт Запросы совладельцев содержит «список пожеланий», которые должны использоваться в качестве первичной информации для определения модели прецедентов, прецедентов и дополнительных спецификаций. Хотя подавляющее большинство «пожеланий» обычно выясняется в итерациях стадий Начало и Уточнение, запросы совладельцев должны собираться в течение всего проекта.

Цель документа Запросы совладельцев состоит в том, чтобы зафиксировать запросы к выполняемому проекту так, как они были первоначально сформулированы. Хотя за этот документ отвечает системный аналитик, ему будут помогать специалисты по маркетингу, конечные пользователи, заказчики – все, кто рассматривается как совладелец законченного проекта.

Артефакт может содержать любой тип запросов совладельцев к разрабатываемой системе. Кроме того, он должен содержать ссылки на все типы внешних источников, которым должна подчиниться система. Примерами таких внешних источников являются:

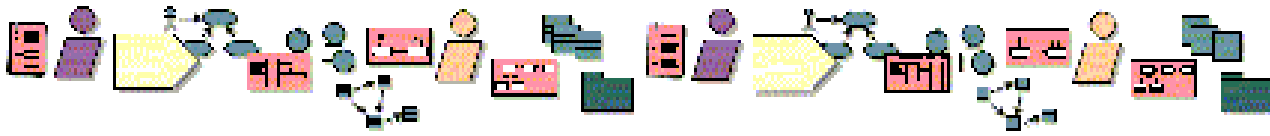
- Инструкции по работе
- Предложения и поручения
- Деловые правила
- Законы и другие правовые акты
- Наследуемые системы
- Деловые модели
- Любые результаты сеансов выявления требований и семинаров

Синхронизация

Запросы совладельцев собираются главным образом в течение стадий начала и уточнения, однако их сбор должен продолжаться в течение всего жизненного цикла проекта для планирования расширений и модификаций изделия.

Создание и уточнение артефакта Запросы совладельцев происходит в результате действия:

- Выявление запросов совладельцев



Запросы совладельцев предоставляют исходную информацию для следующих действий:

- Создание общего (делового) словаря
- Создание общего словаря
- Детализация делового прецедента
- Детализация прецедента
- Разработка документа Видение
- Поиск субъектов и прецедентов
- Поиск деловых субъектов и прецедентов
- Моделирование интерфейса пользователя

Инструментальная поддержка

Интерактивная версия Rational Unified Process содержит шаблон Microsoft Word для документа Запросы совладельцев.

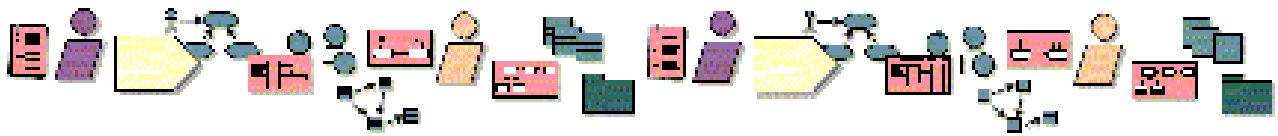
Шаблон Запросы совладельцев.dot представлен в качестве отдельного приложения к этому выпуску.

Rational RequisitePro существенно расширяет возможности работы с документом. Стиль работы аналогичен тому, который описан для артефакта Видение (см. стр. 12-10). Для запросов совладельцев существует предопределенный тип требований (SN – Stakeholder Needs).

Контрольные точки

Как и для других артефактов, для Запросов совладельцев Rational Unified Process предлагает ряд вопросов, положительный ответ на которые свидетельствует о высоком качестве артефакта. Это:

- Правильно ли выбраны совладельцы, привлеченные к созданию этого артефакта?
- Все ли прошлые запросы были пересмотрены для этого выпуска системы?
- Правильно ли был установлен набор источников информации?
- Подходящие ли методы применялись для извлечения информации?
- Достаточно ли полно отражает содержание этого артефакта все аспекты, представляющие интерес для проекта?



Артефакт: Модель прецедентов

Краткий обзор

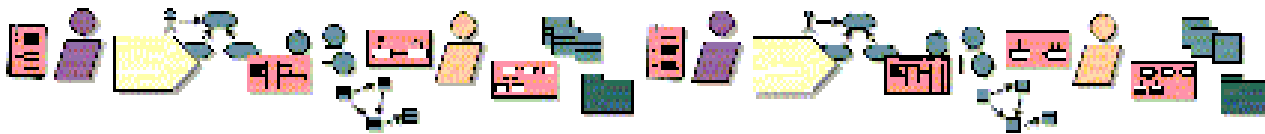
Модель прецедентов – одно из ключевых понятий Rational Unified Process. Мы уже многократно ссылались на эту модель (см. например, стр. 4-4 – 4-7), и еще много раз будем ссылаться. Частным случаем модели прецедентов является модель деловых прецедентов (см. стр. 9-3 – 9-11).

Модель прецедентов используется следующим образом:

- Заказчик одобряет модель прецедентов. Получив это одобрение, Вы будете уверены, что система – это то, что хочет получить заказчик. Вы можете также использовать модель при обсуждении системы с заказчиком в ходе разработки.
- Потенциальные пользователи используют модель прецедентов для лучшего понимания системы.
- Архитектор использует модель прецедентов для идентификации архитектурно существенных функциональных возможностей.
- Проектировщики используют модель прецедентов для получения кратких сведений о системе. Например, когда Вы совершенствуете систему, Вам для облегчения этой работы понадобится документация по модели прецедентов.
- Руководитель проекта использует модель прецедентов для планирования работ по моделированию прецедентов и дальнейшему проектированию.
- Сотрудники организации, не работающие непосредственно над проектом, но заинтересованные в информации о его продвижении (администрация и т.д.), используют модель прецедентов для понимания того, что делается.
- Рецензенты просматривают модель прецедентов, чтобы обеспечить соответствующую регулярную обратную связь с разработчиками.
- Проектировщики используют модель прецедентов как основание для своей работы.
- Тестировщики используют модель прецедентов, чтобы как можно раньше иметь возможность планировать испытания (прецедента и его интеграции в систему).
- Те, кто будут разрабатывать следующую версию системы, будут использовать модель прецедентов, чтобы понять, как работает существующая версия.
- Авторы документации (технические писатели) используют прецеденты как основание при написании руководств для пользователей системы.

Толкование

Модель прецедентов – это модель назначенных системе функций и ее среды; она служит соглашением между заказчиком и разработчиками. Прецеденты – это тот ориентир, который направляет все действия по разработке системы. Одна и та же модель прецедентов – результат



потока работ Требования – используется как исходная информация в потоках работ Анализ и проектирование и Испытания.

Диаграмма ниже показывает часть модели прецедентов системы рециркуляторной машины.



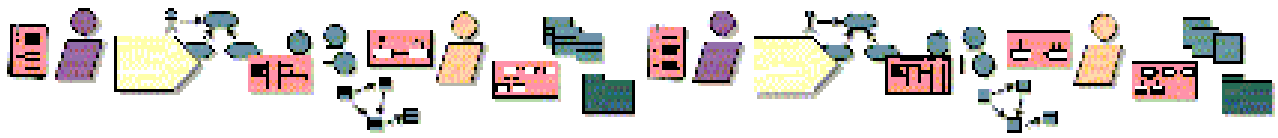
Диаграмма прецедентов показывает пример модели прецедентов с субъектами и прецедентами.

Существует множество способов моделирования системы, каждый из которых обслуживает различную цель. Однако, наиболее важная роль модели прецедентов состоит в том, чтобы объяснить поведение системы заказчику или конечному пользователю. Следовательно, модель должна быть проста для понимания.

Пользователи и любая другая система, которая может взаимодействовать с системой – это субъекты. Поскольку они представляют пользователей системы, субъекты помогают ограничить систему и дают более ясное представление того, что предполагается делать. Прецеденты разрабатываются исходя из потребностей субъектов. Это гарантирует, что система будет такой, как ожидали пользователи.

Достаточно часто модель прецедентов вырождается в функциональную декомпозицию системы. Чтобы избежать этого, обратите внимание на следующие признаки:

- «Маленькие» прецеденты означают, что описания потоков событий ограничиваются только одним или несколькими предложениями.
- «Много» прецедентов, означает, что число прецедентов кратно скорее сотне, чем десятку.
- Названия прецедентов, которые являются конструкциями подобными «эта операция над этими специфическими данными» или «выполнение этой функции с этими специфическими данными». Например, «Ввод в банкомат персонального идентификатора» не нужно моделировать как отдельный прецедент системы банкомата, так как никто не использовал бы систему, чтобы делать только это. Прецедент – это законченный поток событий, который приводит к чему-либо полезному для субъекта.



Чтобы уйти от функциональной декомпозиции, Вы должны убедиться, что модель прецедентов помогает ответить на вопросы, подобные следующим:

- Каков контекст системы?
- Зачем создается система?
- Чего хочет достичь пользователь при использовании системы?
- Какое значение имеет система для пользователей?

Синхронизация

Модель прецедентов, прежде всего, устанавливает функциональные требования к системе и предоставляет исходные данные для анализа и проектирования архитектуры. Ее желательно использовать как можно раньше в стадиях Начало и Уточнение для выделения контекста системы.

Субъекты и прецеденты отыскиваются с использованием запросов заказчиков и потенциальных пользователей как самой важной информации. Когда они обнаружены, прецеденты и субъекты должны быть кратко описаны. Прежде, чем описывать прецеденты подробно, модель прецедентов должна быть рассмотрена заказчиком, чтобы проверить, что все прецеденты и субъекты найдены, и что вместе они могут обеспечить то, что хочет заказчик.

Когда субъекты и прецеденты найдены, можно начинать подробно описывать потоки событий каждого прецедента. Эти описания показывают, как система взаимодействует с субъектами и что система делает в каждом конкретном случае. Описания выполняются, как правило, в стадии Конструирование. В итерационной среде разработки Вы будете выбирать подмножество прецедентов, которые будут детализированы в каждой итерации.

Наконец, законченная модель прецедентов (включая описания прецедентов) рассматривается, и разработчики и заказчики используют ее, чтобы договориться о том, что должна делать система.

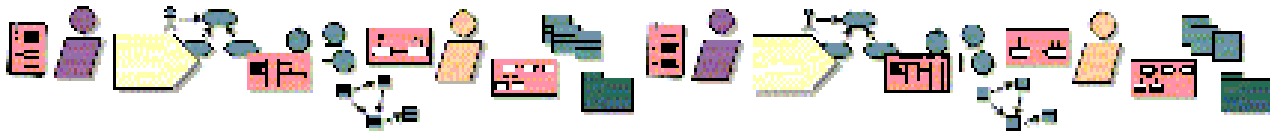
Поскольку модель прецедентов является очень мощным инструментом планирования, она вообще используется во всех стадиях цикла разработки.

Создание и уточнение Модели прецедентов происходит в результате следующих действий Rational RequisitePro:

- Выявление запросов совладельцев
- Поиск субъектов и прецедентов
- Обзор требований
- Структурирование модели прецедентов

Модель прецедентов предоставляет исходную информацию для следующих действий:

- Архитектурный анализ



- Фиксация общего словаря
- Проектирование испытаний
- Детализация прецедента
- Детализация требований к программному обеспечению
- Разработка материалов поддержки конечного пользователя
- Управление зависимостями
- Планирование испытания
- Назначение приоритетов прецедентов
- Обзор требований
- Структурирование модели прецедентов
- Анализ прецедента
- Написание примечаний к выпуску

Не функциональные требования

Нетрудно увидеть, что прецеденты – это очень хороший способ фиксации функциональных требований к системе. Но что же делать с не функциональными требованиями? Что они такое и где они фиксируются?

Не функциональные требования чаще всего выступают как требования применимости, надежности, эффективности и т.д. Часто это требования, которые указывают на необходимость согласованности с некоторыми юридическими и нормативными актами. Это могут быть проектные ограничения из-за используемой операционной системы, среды платформы, проблем совместимости или применяемых стандартов. Вообще, можно считать, что любое требование, которое учитывает не больше, чем один параметр проекта, должно быть расценено как проектное ограничение.

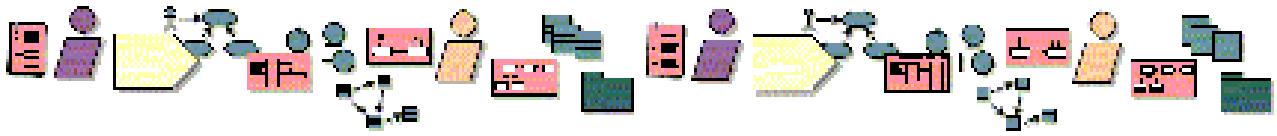
Многие не функциональные требования относятся к отдельному прецеденту и должны быть свойствами этого прецедента. В таком случае они фиксируются в пределах потока событий прецедента или как специальное требование прецедента.

Пример:

В системе рециркуляторной машины не функциональным требованием, определенным для прецедента Возврат предметов залога, может быть:

Машина должна быть способна распознать предметы залога с надежностью большей, чем 95 процентов.

Часто не функциональные требования относятся ко всей системе. Такие требования фиксируются в Дополнительной спецификации (см. стр. 13-15).



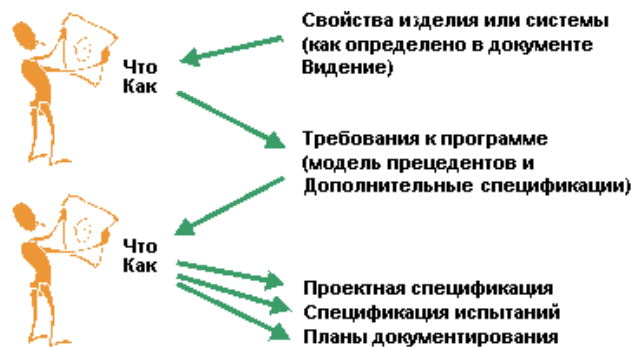
Пример:

В системе рециркуляторной машины не функциональным требованием, которое относится ко всей системе, может быть:

Машина обслуживает только одного пользователя одновременно.

Дилемма: «что» против «как»

Одна из наиболее трудных вещей – это научиться определять, на каком уровне детализации прецеденты должны «начинаться и заканчиваться». Где возникают возможности начинаться прецедентам, и где прецеденты заканчиваются, и начинается проектирование? Мы часто говорим, что прецеденты или требования к программному обеспечению должны заявить то, «что» делает система, а не то, «как» она это делает. Рассмотрите следующую иллюстрацию:



Потолок одного человека – это пол для другого.

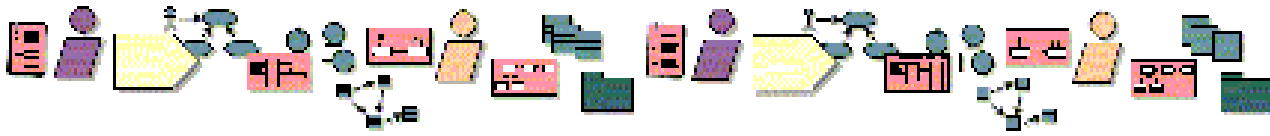
В зависимости от вашего фона, Вы будете использовать различный контекст для решения вопроса, что для Вас – «что» и что – «как». Это нужно учитывать при определении, действительно ли некоторые детали нужно включать в модель прецедентов.

Конкретные и абстрактные прецеденты

Существует различие между конкретными и абстрактными прецедентами. Конкретный прецедент инициализируется субъектом и представляет законченный поток событий. «Законченный» означает, что экземпляр прецедента исполняет полную операцию, которую запрашивает субъект.

Абстрактный прецедент никогда не обрабатывается сам по себе. Абстрактные прецеденты включаются в другие прецеденты (связь включения), расширяют другие прецеденты (связь расширения) или обобщают другие прецеденты (обобщение прецедентов). Когда иницируется конкретный прецедент, создается экземпляр прецедента. Этот экземпляр показывает также и поведение, определенное связанными с ним абстрактными прецедентами. Таким образом, из абстрактных прецедентов никакие отдельные экземпляры не создаются.

Различие между этими двумя видами прецедентов важно, так как субъекты будут «видеть» в системе и инициализировать только конкретные прецеденты.



Вы указываете, что это абстрактный прецедент, записывая его название курсивом.

Пример:



Прецедент Создание задачи включен в прецедент Регистрация заказа. Создание задачи - абстрактный прецедент.

В системе складской обработки абстрактный прецедент Создание задачи включен в прецедент Регистрация заказа. Когда инициализируется Регистрация заказа, создается экземпляр Регистрации заказа, который, кроме потока событий прецедента Регистрации заказа выполняет также поток событий, описанный во включенном прецеденте Создание задачи. Создание задачи никогда не выполняется отдельно, а всегда как часть Регистрации заказа (или любых других прецедентов, в которые он включен). Следовательно, Создание задачи является абстрактным прецедентом.

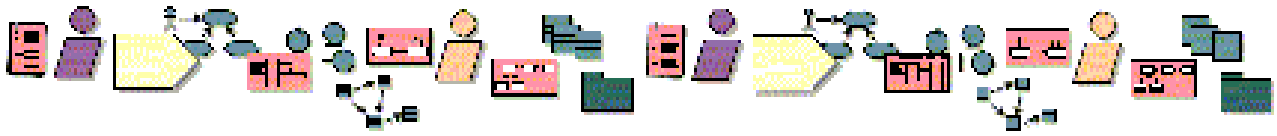
Структурирование модели прецедентов

Имеются три главных причины, по которым выполняется структурирование модели прецедентов:

- Упростить понимание прецедентов.
- Выделить разделы общего поведения, свойственного многим прецедентам
- Упростить обслуживание модели прецедентов.

Структурирование, однако, не первая вещь, которую Вы делаете. Нет никакого смысла в структурировании прецедентов, пока Вы не узнаете об их поведении немного больше, чем говорится в одном предложении краткого описания. Вы должны, по крайней мере, установить пошаговую схему потока событий прецедента, чтобы удостовериться, что Ваши решения основаны на достаточно точном понимании поведения.

Для структурирования прецедентов есть три вида связей. Вы будете использовать эти связи, чтобы выносить за скобки прецеденты, которые могут многократно использоваться в других прецедентах, как специализации или варианты прецедентов. Прецедент, который представляет



модификацию, называется добавляемым прецедентом. Прецедент, который при этом изменяется, называется основным прецедентом.

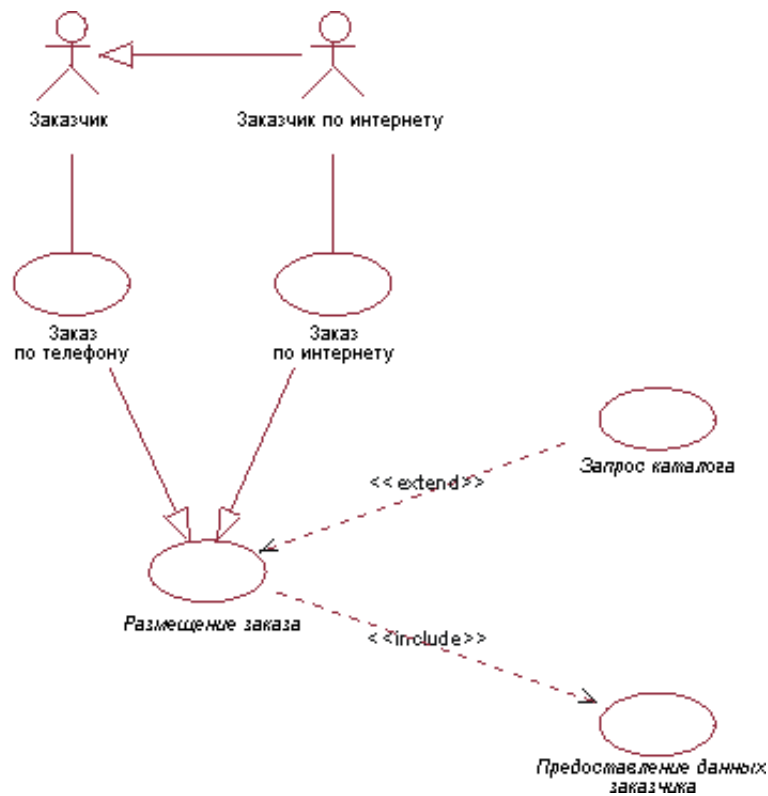
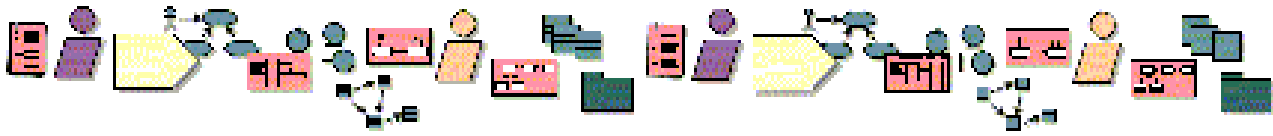
- Если имеется часть основного прецедента, представляющая функцию, от которой зависит только результат прецедента, а не способ получения результата, Вы можете вынести эту часть за скобки как добавляемый прецедент. Добавление, явно вставленное в основной прецедент, использует связь включения.
- Если имеется часть основного прецедента, которая является необязательной, или не необходимой для понимания основной цели прецедента, Вы можете вынести за скобки эту часть как добавляемый прецедент, чтобы упростить структуру основного прецедента. Добавление, неявно вставленное в основной прецедент, использует связь расширения.
- Если имеются прецеденты, которые имеют общее в поведении и структуру и подобие в цели, их общие части можно вынести за скобки как основной (исходный) прецедент, который наследуется добавляемыми (дочерними) прецедентами. Дочерние прецеденты могут вставлять новое поведение и изменять существующее поведение в структуре, которую они наследуют от исходного прецедента.

Пример:

Рассмотрите часть модели прецедентов для Системы управления заказом.

Полезно разделить обычного заказчика и заказчика по интернету, так как они имеют слегка различные свойства. Однако, так как заказчик по интернету обладает всеми свойствами обычного заказчика, Вы можете говорить, что заказчик по интернету – это специализация заказчика, обозначенного как субъект-обобщение.

Конкретные прецеденты на этой диаграмме – Заказ по телефону (инициализируемый субъектом Заказчик) и Заказ по интернету (инициализируемый Заказчиком по интернету). Эти прецеденты – два варианта общего прецедента Размещение заказа, который в этом примере является абстрактным. Прецедент Запрос каталога представляет необязательный сегмент поведения, которое не является частью основной цели Размещения заказа. Он выносится за скобки абстрактного прецедента, чтобы упростить прецедент Размещение заказа. Прецедент Предоставление данных заказчика представляет сегмент поведения, который выносится за скобки, так как его результат воздействует только на отдельную функцию прецедента Размещение заказа. Кроме того, прецедент Предоставление данных заказчика может многократно использоваться в других прецедентах. Прецеденты Запрос каталога и Предоставление данных заказчика в этом примере являются абстрактными прецедентами.



Эта диаграмма прецедентов отображает часть модели прецедентов для Системы управления заказами.

Ниже различия между тремя видами связей прецедента показаны более детально:

Вопрос: Как определяется направление связи?

Расширение – ссылка добавляемого прецедента на основной прецедент.

Включение – ссылка основного прецедента на добавляемый прецедент.

Обобщение – ссылка добавляемого прецедента (дочернего) на основной прецедент (исходный).

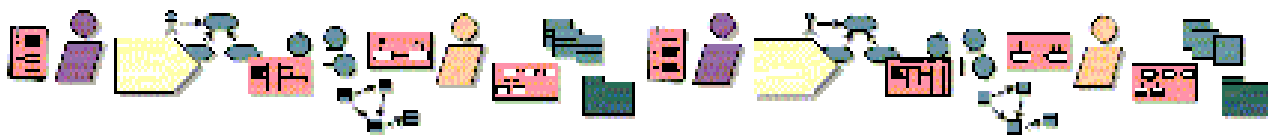
Вопрос: Может ли связь иметь множественность?

Расширение – да, с точки зрения добавления.

Включение – нет. Если Вы хотите включить один и тот же сегмент поведения больше, чем однажды, он должен быть объявлен в основном прецеденте.

Обобщение – нет.

Вопрос: Может ли связь иметь условие?



Расширение – да.

Включение – нет. Если Вы хотите выразить условие включения, Вы должны оговорить это явно в основном прецеденте.

Обобщение – нет.

Вопрос: Является ли добавляемый прецедент абстрактным?

Расширение – часто да, но не обязательно.

Включение – да.

Обобщение – часто нет, но это возможно.

Вопрос: Изменяет ли добавление поведение основного прецедента?

Расширение – неявно изменяет поведение основного прецедента.

Включение – явно изменяет эффект основного прецедента.

Обобщение – если обрабатывается основной прецедент (исходный), на него не влияет наличие дочернего. Чтобы получить результаты добавления, должен быть обработан добавляемый прецедент (дочерний).

Вопрос: Должен ли основной прецедент быть законченным и значимым?

Расширение – да.

Включение – вместе с добавлениями, да.

Обобщение – если он абстрактный, нет.

Вопрос: Должен ли добавляемый прецедент быть законченным и значимым?

Расширение – нет.

Включение – нет.

Обобщение – вместе с основным прецедентом (исходным), да.

Вопрос: Может ли добавляемый прецедент обращаться к атрибутам основного прецедента?

Расширение – да.

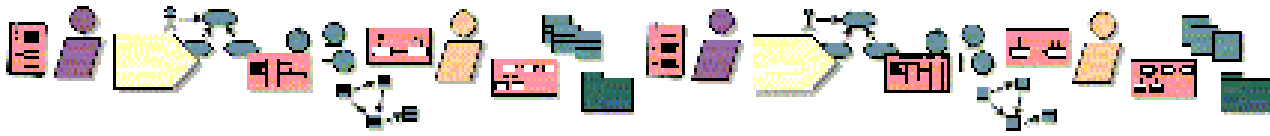
Включение – нет. Включение капсулировано, и «видит» только себя.

Обобщение – да, нормальными механизмами наследования.

Вопрос: Может ли основной прецедент обращаться к атрибутам добавляемого прецедента?

Расширение – нет. Основной прецедент должен быть правильным в отсутствие добавления.

Включение – нет. Основной прецедент знает только об эффекте добавления. Добавление капсулировано.



Обобщение – нет. Основной прецедент (исходный) должен в этом смысле быть правильным в отсутствие добавления.

Другой аспект организации модели прецедентов состоит в том, чтобы для упрощения понимания группировать прецеденты в пакеты. Модель прецедентов может быть организована как иерархия пакетов прецедентов, с «листьями», которые являются субъектами или прецедентами.



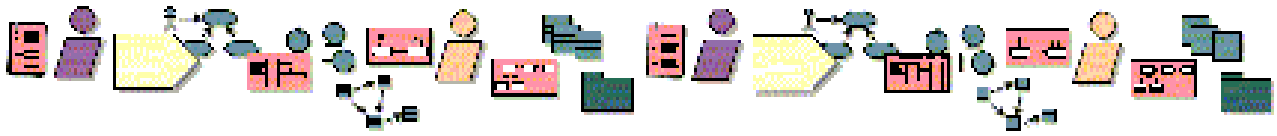
Эта диаграмма показывает иерархию модели прецедентов. Стрелки указывают возможное владение.

Всегда ли прецеденты связаны с субъектами?

Выполнение каждого прецедента связано с одним или несколькими субъектами. Экземпляр прецедента всегда начинается субъект, запрашивая какое-либо действие системы. Это подразумевает, что каждый прецедент должен иметь связи-ассоциации с субъектами. Такое правило обеспечивает создание системы для обеспечения только тех функциональных возможностей, которые необходимы пользователям, и ничего иного. Наличие прецедентов, которые никто не запрашивает – это признак того, что в модели прецедентов или в требованиях что-то неправильно.

Но для этого правила имеются некоторые исключения:

- Если прецедент абстрактный (не обрабатывается отдельно), его поведение не может включать взаимодействие с субъектом. В этом случае между этим абстрактным прецедентом и субъектами не будет связей-ассоциаций.
- Дочерний прецедент в отношениях обобщения не должен связываться с субъектом, если исходный прецедент полностью описывает связь субъекта.
- Основной прецедент в отношениях включения не должен связываться с субъектом, если прецедент включения полностью описывает связь субъекта.



- Прецедент может быть инициирован согласно расписанию (например, один раз в неделю или один раз в день). Это означает, что инициатором являются системные часы. Системные часы внутренние по отношению к системе, и прецедент инициализируется не субъектом, а внутренним событием в системе. Если никакое другое взаимодействие субъекта с прецедентом не происходит, он не будет иметь никаких связей с субъектами. Однако, для ясности, Вы можете использовать фиктивный субъект «Время», чтобы показать на диаграмме прецедентов, как инициируется прецедент.

Инструментальная поддержка

Как правило, модель прецедентов иллюстрируется одной, а чаще, многими, диаграммами прецедентов. Для работы с диаграммами используется Rational Rose.

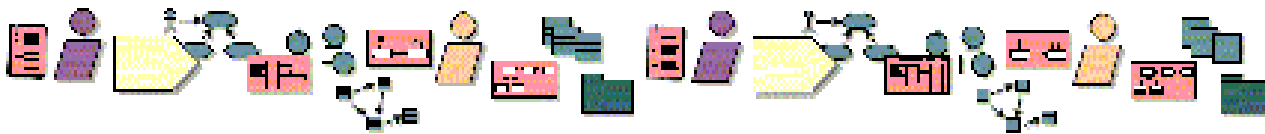
Диаграмма прецедентов отображает субъекты, прецеденты и их связи. Диаграммы прецедентов могут быть организованы в пакеты прецедентов, показывая только то, что является уместным в пределах отдельного пакета.

Никаких строгих правил относительно того, как иллюстрировать модель диаграммами прецедентов, не существует. Показывайте интересные, по Вашему мнению, связи в модели. Могут быть интересны следующие диаграммы:

- Субъекты, принадлежащие к одному и тому же пакету прецедентов.
- Субъект и все прецеденты, с которыми он взаимодействует. Диаграмма этого типа может функционировать как локальная диаграмма субъекта, и вероятно, быть связанной с ним.
- Прецеденты, которые обрабатывают одну и ту же информацию.
- Прецеденты, используемые одной и той же группой субъектов.
- Прецеденты, которые часто выполняются в одной последовательности.
- Прецеденты, которые принадлежат одному и тому же пакету прецедента.
- Наиболее важные прецеденты. Диаграмма этого типа может функционировать как резюме модели, и вероятно может быть включена в представление прецедентов верхнего уровня.
- Прецеденты, разработанные вместе (в пределах одного и того же приращения).
- Характерный прецедент и его связи с субъектами и с другими прецедентами. Диаграмма этого типа может функционировать как локальная диаграмма прецедента, и вероятно, быть связанной с ним.

Рекомендуется включить каждый субъект, прецедент и связь, по крайней мере, в одну из диаграмм. Если это сделает модель прецедентов более ясной, они могут быть частью нескольких диаграмм, и Вы можете показывать их несколько раз в одной и той же диаграмме.

Кроме диаграмм, для иллюстрации модели прецедентов выполняется ее **обзорное описание**.



Обзорное описание модели прецедентов должно:

- Перечислять основные прецеденты системы (причина создания системы).
- Суммировать важные технические факты относительно системы.
- Указывать границы системы – вещи, которые система, как предполагается, не делает.
- Указывать среду системы, например, целевые платформы и существующее программное обеспечение.
- Описывать все последовательности, в которых обычно выполняются прецеденты в системе.
- Определять функциональные возможности, не обработанные моделью прецедентов.

Пример:

Ниже следует пример обзорного описания модели прецедентов рециркуляторной машины:

Эта модель содержит три субъекта и три прецедента. Основной прецедент – Возврат предметов, который представляет главную цель работы рециркуляторной машины.

Прецеденты поддержки:

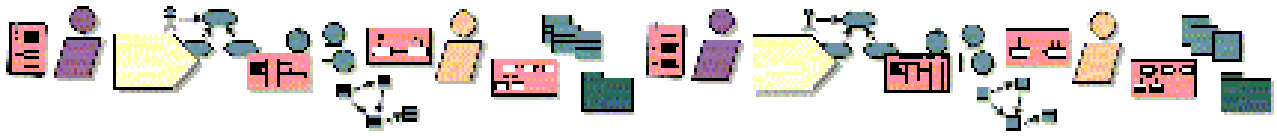
- Печать ежедневного отчета, который позволяет оператору получать отчет относительно того, сколько предметов возвращено.
- Администрирование предметами залога, который позволяет оператору изменять значение возмещения для типа предмета залога или добавлять новые типы предметов залога.

Rational RequisitePro содержит шаблон документа Обзор модели деловых прецедентов, который используется Rational SoDA для создания документа, базируясь на информации, имеющейся в модели Rational Rose

Контрольные точки

Вы имеете действительно хорошую модель прецедентов, если Вы сможете подтвердить все следующие утверждения:

- Раздел Введение описания модели прецедентов обеспечивает ясный, краткий обзор цели и функциональных возможностей системы.
- Модель прецедентов ясно представляет поведение системы; рассматривая модель, можно легко понять то, что делает система.
- Все прецеденты идентифицированы; прецеденты все вместе объясняют все требуемое поведение.
- Каждое функциональное требование отображается, по крайней мере, одним прецедентом.
- Все требования, которые не содержат функциональности, но которые должны быть удовлетворены определенными прецедентами, отображены в этих прецедентах.



- Модель прецедентов не содержит лишнего поведения; все прецеденты могут быть подтверждены прослеживанием их назад к функциональным требованиям.
- Все связи между прецедентами действительно нужны (то есть имеется подтверждение для всех связей включения, расширения и обобщения).
- Там, где модель большая, и/или ответственности частей модели распределены, используются соответствующие пакеты прецедентов.
 - Межпакетные зависимости сокращены или устранены для предотвращения конфликтов монопольного использования модельных элементов.
 - Пакетирование делает модель проще для понимания.

Артефакт: Дополнительная спецификация

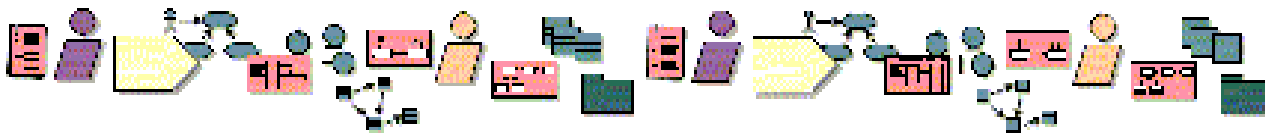
Краткий обзор

Дополнительные спецификации охватывают требования к системе, которые не охватываются в прецедентах модели прецедентов. Такие требования включают:

- Юридические и нормативные требования и применяемые стандарты.
- Атрибуты качества разрабатываемой системы, включая требования применимости, надежности, эффективности и пригодности к эксплуатации.
- Другие требования типа требований операционной системы и среды, совместимости и конструктивных ограничений.

Этот артефакт в разной мере используют следующие участники проекта:

- Системный аналитик создает и сопровождает Дополнительную спецификацию, которая служит средством общения между системным аналитиком, заказчиком и другими разработчиками.
- Проектировщики используют Дополнительную спецификацию как справочник при определении ответственности, действий и атрибутов классов и при приспособлении классов к среде выполнения.
- Кодировщики обращаются к Дополнительной спецификации за исходной информацией при выполнении классов.
- Руководитель проекта обращается к Дополнительной спецификации за исходной информацией при планировании итераций.
- Тестировщики используют Дополнительную спецификацию для проверки соответствия системы всем требованиям.



Синхронизация

Создание, сопровождение и использование Дополнительной спецификации идет «рука об руку» с созданием, сопровождением и использованием модели прецедентов, имея ввиду, что:

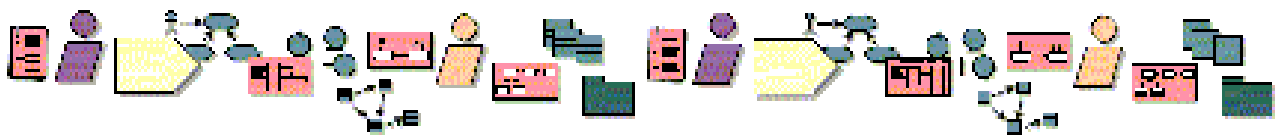
- Первоначально она рассматривается в стадии Начало, как дополнение к возможностям системы, определенным в модели прецедентов.
- Она постепенно усовершенствуется вместе с моделью прецедентов в течение стадий Уточнение и Конструирование.

Создание и сопровождение (изменение) артефакта Дополнительная спецификация в Rational Unified Process происходит в действиях:

- Разработка документа Видение
- Детализация прецедента
- Детализация требований к программному обеспечению
- Поиск субъектов и прецедентов
- Определение требования автоматизации
- Обзор требований

Артефакт Дополнительная спецификация предоставляет исходную информацию для следующих действий:

- Архитектурный анализ
- Проектирование класса
- Проектирование базы данных
- Описание архитектуры реального времени
- Проектирование испытания
- Детализация прецедента
- Детализация требований к программному обеспечению
- Идентификация проектных элементов
- Идентификация проектных механизмов
- Выполнение компонента
- Выполнение испытания
- Управление зависимостями
- Моделирование интерфейса пользователя



-
- Выполнение испытания модуля
 - Планирование испытания
 - Прототипирование интерфейса пользователя
 - Обзор требований
 - Обзор архитектуры
 - Обзор проекта
 - Структурирование модели выполнения
 - Структурирование модели прецедентов
 - Анализ прецедента
 - Проектирование прецедента

Инструментальная поддержка

Интерактивная версия Rational Unified Process содержит шаблон Microsoft Word для документа Дополнительная спецификация.

Шаблон Дополнительная спецификация.dot представлен в качестве отдельного приложения к этому выпуску.

Для дополнительных требований в Rational RequisitePro существует предопределенный тип требований – SUPL.

Контрольные точки

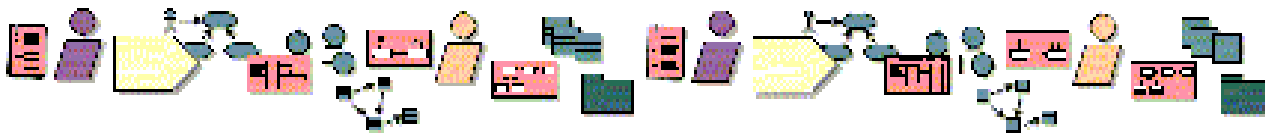
Чтобы детализировать все требования, которые не определены в пределах модели прецедентов, должны быть освещены следующие основные проблемы:

Функциональные возможности: Что делает, как предполагается, программное обеспечение? Эта информация должна включать:

- Контроль информации при вводе
- Общие ответы на аварийные ситуации, включая: переполнение, возможности коммуникаций, обработку ошибок и восстановление
- Влияние параметров
- Связи входной и выходной информации, включая последовательности ввода-вывода и формулы преобразования входной информации в выходную

Внешние интерфейсы: как программное обеспечение взаимодействует с людьми, аппаратными средствами системы, другими аппаратными средствами и другим программным обеспечением?

Эффективность: каково быстроедействие, доступность, время срабатывания, время выполнения



различных программных функций и т.д.? Все ли статические и динамические требования включены?

Логические требования базы данных: все ли логические требования к информации, которая должна быть помещена в базу данных, определены? Они могут включать:

- Типы информации, используемой различными функциональностями
- Частоту использования
- Возможности доступа
- Сущности данных и их связи
- Ограничения целостности
- Требования к сохранности данных

Соответствие стандартам: все ли требования происходят из существующих стандартов и указанных правил? Как они будут отслеживаться?

Атрибуты: достаточно ли полно определены надежность, доступность, взаимозаменяемость, правильность, надежность в эксплуатации, защита и т.д.?

Проектные ограничения, наложенные на выполнение: действительно ли указаны все требуемые стандарты, язык физической реализации, стратегия поддержания целостности базы данных, ограничения ресурсов, среды и т.д.?

Артефакт: Атрибуты требований

Краткий обзор

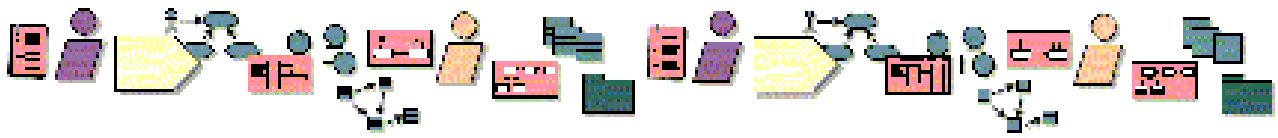
Артефакт Атрибуты требований предоставляет репозиторий текстов требований, их атрибутов и трассируемости для всех требований. Он должен обеспечить доступ для просмотра текущего состояния требований в следующих представлениях:

1. Матрицы атрибутов требований

Для каждого типа требований (см. стр. 11-3) представляется матрица, которая перечисляет требования по одной оси и все атрибуты этого типа требований по другой оси. Для каждого требования показываются значения его соответствующих атрибутов.

2. Матрицы трассируемости требований

Для каждого типа требований представляется матрица, которая перечисляет требования по одной оси и все трассируемые объекты по другой оси. Для каждой трассируемости показывается ее состояние (ОК или сомнительная). О сомнительности трассируемости см. стр. 11-5)



3. Дерево трассируемости требований

Дерево трассируемости обеспечивает графическое представление связей трассируемости к или от требований определенного типа требований.

Синхронизация

Конфигурация этого репозитория определяется в Плане управления требованиями. Репозиторий нужно установить и начать заполнять в конце стадии Начало. Содержание будет пополняться и модифицироваться в течение всего цикла разработки.

Создание и модификация атрибутов требований происходит в результате следующих действий:

- Разработка документа Видение
- Детализация прецедента
- Детализация требований к программному обеспечению
- Разработка плана управления требованиями
- Назначение приоритетов прецедентов
- Управление зависимостями

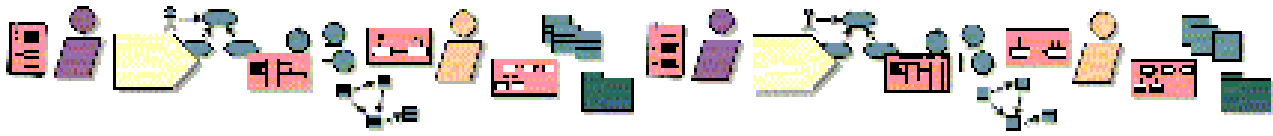
Атрибуты требований предоставляют исходную информацию для следующих действий:

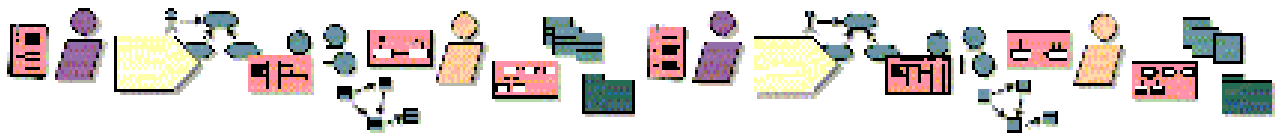
- Детализация требований к программному обеспечению
- Управление зависимостями
- Назначение приоритетов прецедентов

Инструментальная поддержка

Rational RequisitePro содержит все необходимые средства для поддержки артефакта Атрибуты требований.

В следующем выпуске мы закончим обсуждение артефактов и подведем итоги знакомства с потоком работ Требования.





Поток работ «Требования» (продолжение 3)

Артефакт: Субъект

Краткий обзор

Субъект определяет связанный набор ролей, которые пользователи могут играть при взаимодействии с системой. Пользователем может быть человек или внешняя система.

На иллюстрации на стр. 12-1 артефакт Субъект находится в зоне ответственности работника Проектировщик интерфейса пользователя. Действительно:

- **проектировщик интерфейса пользователя** отвечает за целостность описания субъектов-людей, гарантируя что:
 - Для каждого субъекта зафиксированы характеристики, необходимые для формирования интерфейса пользователя
 - Каждый субъект имеет правильные связи-ассоциации с прецедентами, с которыми он взаимодействует
 - Каждый субъект – часть правильных отношений обобщения
 - Каждый субъект определяет связующую роль и независим от других субъектов
 - Локальная диаграмма прецедентов, описывающая субъекта, читаема и совместима с другими свойствами
- **системный аналитик** имеет подобные обязанности для не-человеческих субъектов, кроме описания характеристик (первый пункт в этом списке).

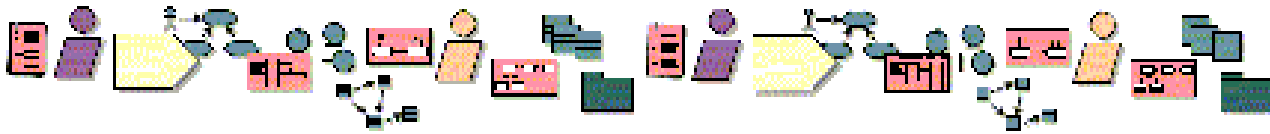
И все же впервые субъекты появляются из-под пера системного аналитика при выполнении им действия Поиск субъектов и прецедентов (см. иллюстрацию на стр. 11-7). Поэтому я думаю, что общие свойства этого артефакта нужно рассматривать в группе с артефактами, создаваемыми системным аналитиком.

Толкование

Экземпляр субъекта – это кто-то или что-то вне системы, что взаимодействует с системой.

Класс субъекта определяет набор экземпляров субъектов, в которых все экземпляры субъектов играют одинаковую роль относительно системы.

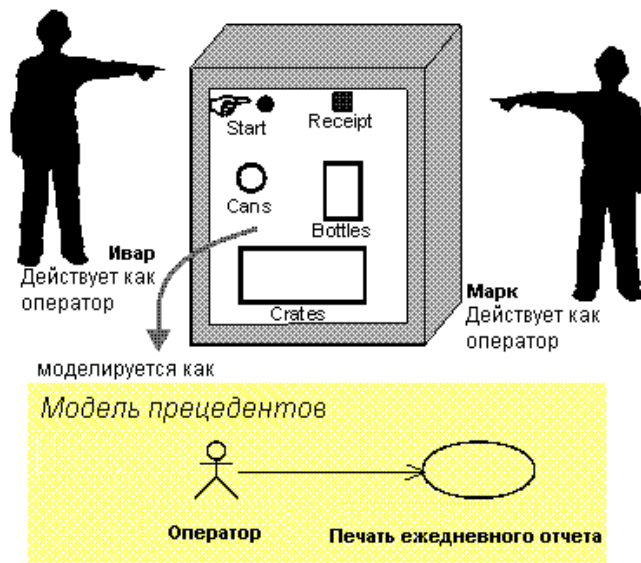
Чтобы полностью понять цель системы, Вы должны знать **для кого** предназначена эта система, то есть, кто будет использовать систему. Различные типы пользователей представляются как



субъекты.

Субъект – это что-то, что обменивается данными с системой. Субъектом может быть пользователь, внешние аппаратные средства или другая система.

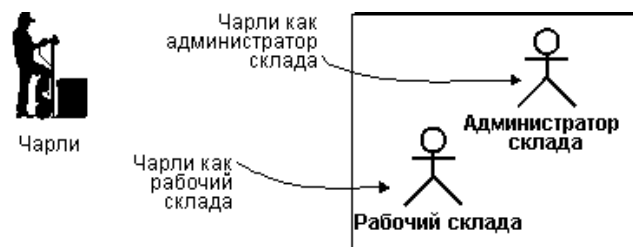
Разница между субъектом и индивидуальным пользователем системы состоит в том, что субъект представляет специфический класс пользователей, а не фактического пользователя. Несколько пользователей могут играть одну и ту же роль, что означает, что они могут быть одним и тем же субъектом. В этом случае каждый пользователь представляет экземпляр субъекта.



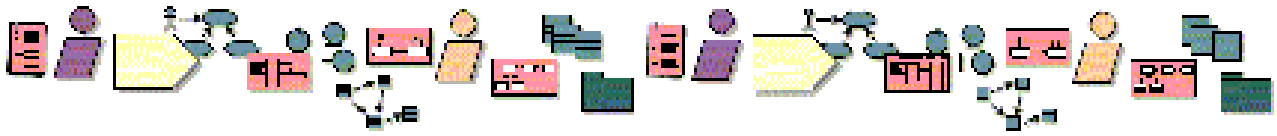
Ивар и Марк - операторы рециркуляторной машины. Когда они используют машину, каждый представлен экземпляром субъекта Оператор.

Однако, в некоторых случаях роль, смоделированную субъектом, играет только один человек. Например, для довольно маленькой системы может быть только один человек, играющий роль администратора системы.

Кроме того, один и тот же пользователь может действовать как несколько субъектов (то есть один и тот же человек может играть различные роли).



Чарли использует Систему складской обработки прежде всего как Администратора склада, но иногда он использует эту систему как обычный Работник склада.



Синхронизация

Субъекты отыскиваются и связываются с прецедентами в стадии Начало при определении границ системы. Характеристики субъектов описываются до того, как моделируются и реализуются интерфейсы пользователя.

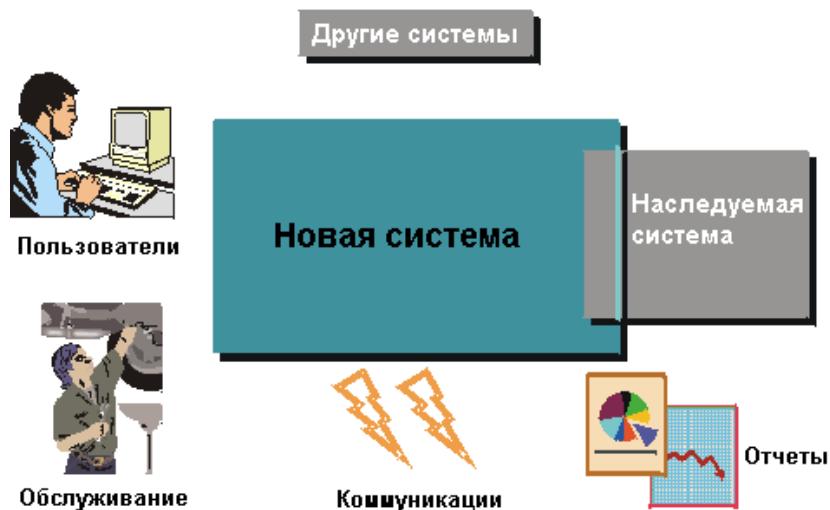
Создание и уточнение артефактов Субъект происходит в результате следующих действий Rational Unified Process:

- Поиск субъектов и прецедентов
- Моделирование интерфейса пользователя

Артефакт предоставляет информацию для следующих действий:

- Моделирование интерфейса пользователя
- Прототипирование интерфейса пользователя

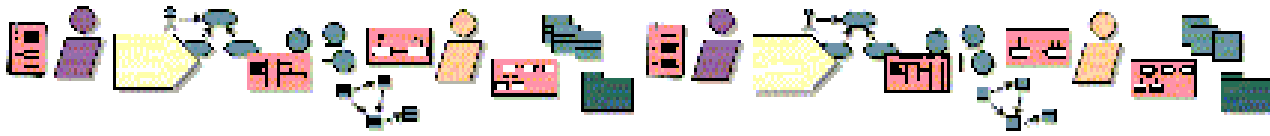
Как искать субъекты



Что из окружения системы станет субъектами системы?

Сначала нужно подумать о личностях, которые будут использовать систему. Как Вы можете их классифицировать? Часто бывает полезно представить в памяти несколько личностей (две или три) и удостовериться, что субъекты, которых Вы идентифицировали, покрывают их потребности. Когда Вы идентифицируете субъекты, полезно иметь в виду следующий набор вопросов:

- Кто поставляет, использует или удаляет информацию?
- Кто будет использовать эти функциональные возможности?



- Кто заинтересован в каких-либо требованиях?
- Где в организации используется система?
- Кто поддерживает и обслуживает систему?
- Каковы внешние ресурсы системы?
- Какие другие системы должны будут взаимодействовать с этой?

Имеется несколько различных аспектов окружения системы, которые Вы представите как отдельные субъекты:

- Пользователи, которые используют главные функции системы.

Пример:

У Системы складской обработки, которая поддерживает работу склада, есть несколько категорий пользователей: Рабочий склада, Служащий учета, Администратор склада. Все эти категории имеют в системе определенные роли, и поэтому Вы должны представить каждую из них отдельным субъектом.

- Пользователи, которые выполняют вторичные функции системы, типа системного администрирования.

Пример:

В рециркуляторной машине, используемой для приема банок, баллонов и контейнеров, главный субъект – это Заказчик, для которого прежде всего и сформирована система. Однако, кто-то должен управлять машиной. Эта роль отводится субъекту – Оператору.

- Внешние аппаратные средства, используемые системой.

Пример:

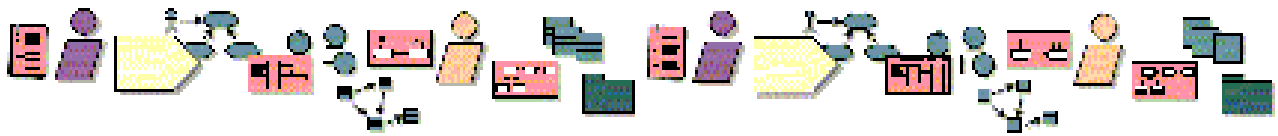
Система вентиляции, которая управляет температурой в помещении, непрерывно контролирует данные датчиков температуры. Поэтому Датчик – это субъект.

- Другие системы, взаимодействующие с системой.

Пример:

Банкомат должен связываться с центральной системой, которая проводит банковские счета. Центральная система – вероятно, внешняя для банкомата, и поэтому должна быть субъектом.

Если Вы строите прикладную программу для интернета, ваши основные субъекты будут в некотором смысле анонимными. Вы действительно не знаете, кем они являются, и Вы не можете сделать никаких предположений относительно их навыков и квалификации. Но Вы все-таки можете описать роль, которую, как Вы ожидаете, они будут играть в вашей системе.



Пример:

Системы, которые предоставляют информацию (типа механизмов поиска), будут иметь вполне анонимных субъектов, которые обращаются к приложению только для того, чтобы найти информацию относительно конкретной темы.

Пример:

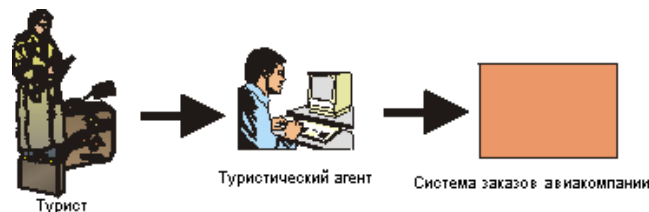
Правительственный информационный сайт, чей регламент состоит в предоставлении любому гражданину или абоненту сети информации относительно законов и правил, их применения, форм и так далее. Например, Налоговое управление США имеет страницу, которая предоставляет информацию о правилах заполнения налоговой декларации. Она предоставляет все формы, доступные в электронном виде, а также позволяет абонентам зарегистрировать свою налоговую декларацию с помощью интернета. Роль основного субъекта в этом случае играет любой, заинтересовавшийся тем, как регистрируется налоговая декларация в США. Конечно, человек, однажды зарегистрировавший свое посещение этой страницы, больше не является анонимным.

Субъекты помогают определять границы системы

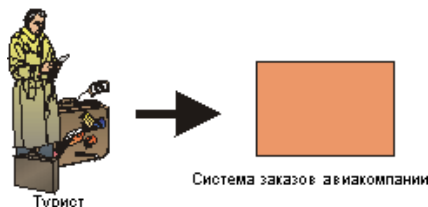
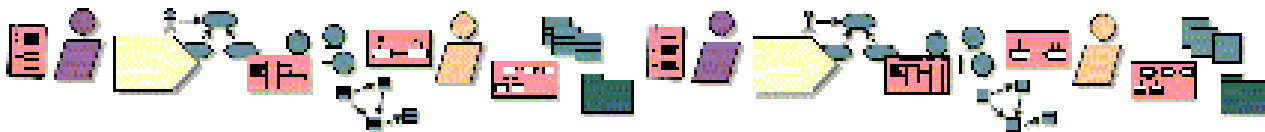
Обнаружение субъектов также означает, что Вы установили границу, которая помогает в понимании цели и контекста системы. Как субъекты должны рассматриваться только те, кто непосредственно связывает свои потребности с системой. Если Вы включаете большее количество ролей, чем их существует в окружении системы, Вы будете моделировать деловую сферу, в которой система будет использоваться опосредованно.

Пример:

Кто является субъектом в Системе заказов авиакомпании? Это зависит от того, строите ли Вы систему, которую будет использовать туристический агент, или Вы строите систему, с которой пассажир может соединяться непосредственно через интернет.



Если Вы строите Систему заказов, которая будет использоваться в туристическом агентстве, субъектом будет Туристический агент. Турист не взаимодействует непосредственно с системой, и поэтому он не субъект



Если Вы строите систему, которая позволит пользователям соединяться с ней через интернет, Турист взаимодействует непосредственно с системой, и поэтому он является субъектом системы.

Отчет

Отчет об артефакте Субъект используют:

- **системные аналитики** при фиксации границ системы;
- **авторы прецедентов** при описании прецедентов и их взаимодействия с субъектами;
- **проектировщики интерфейса пользователя** при фиксации характеристик субъектов – людей
- **аналитики объектов** при реализации прецедентов и их взаимодействия с субъектами.

Интерактивная версия Rational Unified Process содержит предопределенный шаблон и инструкции по созданию отчета о субъекте с использованием SoDA. SoDA автоматизирует генерацию отчета на основе информации модели прецедентов Rational Rose так, чтобы он был создан быстро и точно. Отчет о субъекте может быть сгенерирован SoDA с версиями программ Word или Frame.

Отчет состоит из следующих разделов

1. Краткое описание

Краткое описание субъекта должно включать следующую информацию:

- Кого или что представляет субъект.
- Почему необходим субъект.
- Какие интересы субъект имеет в системе.

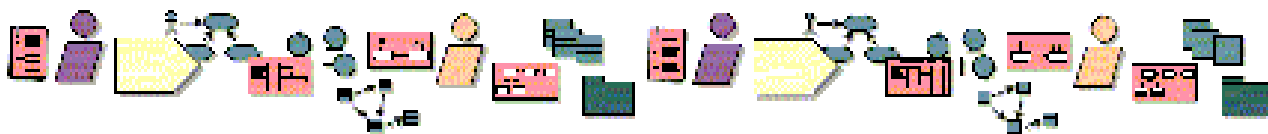
Краткое описание должно состоять, самое большее, из нескольких предложений.

Пример:

В модели прецедентов рециркуляторной машины три субъекта кратко описаны следующим образом:

Заказчик: Заказчик собирает дома баллоны, банки и контейнеры и возвращает их в магазин, чтобы получить деньги обратно.

Оператор: Оператор отвечает за эксплуатацию рециркуляторной машины.



Администратор: Администратор отвечает за вопросы обеспечения деньгами и за обслуживание заказчика в магазине.

2. Характеристики

Характеристики субъекта могут влиять на разрабатываемую систему и, в особенности, на оптимально подходящую форму интерфейса пользователя. Обратите внимание, что, если деловые работники, соответствующие субъектам, уже описаны в модели деловых объектов, некоторые из следующих характеристик, возможно, уже были зафиксированы. Характеристики субъекта включают:

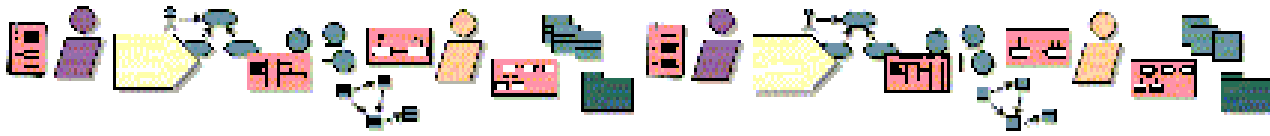
- Область ответственности субъекта.
- Физическая среда, в которой субъект будет использовать систему. Отклонения от идеального случая (когда пользователь находится в тихом офисе, не отвлекается), которые могут затрагивать использование таких вещей как звук, выбор шрифта и надлежащего применения комбинаций устройства ввода данных (например, клавиатуры, сенсорного экрана, мыши и быстрых клавиш).
- Число пользователей, представленных этим субъектом. Это число – полезный фактор при определении значения субъекта и значения частей интерфейса пользователя, который субъект использует.
- Частота, с которой субъект будет использовать систему. Эта частота определит, как долго (интерфейсу пользователя) нужно ожидать субъекта и сохранять его информацию между сеансами.

В большинстве случаев достаточно приблизительная оценка числа пользователей и частоты использования. Разница между 30 и 40 не будет затрагивать форму интерфейса пользователя, но разница между 3 и 30 уже может влиять на нее.

Другие характеристики субъекта включают:

- Уровень знаний субъектом проблемной области. Этот уровень поможет определять необходимые размеры проблемной справки, допустимость использования в интерфейсе пользователя проблемной терминологии.
- Уровень компьютерного опыта субъекта. Этот уровень поможет правильно определить соотношение сложных и упрощенных методов взаимодействия в интерфейсе пользователя.
- Другие прикладные программы, используемые субъектом. Заимствование концепций интерфейса пользователя от этих приложений сократит время обучения субъекта и уменьшит нагрузку на его память, так как субъект уже знаком с этими концепциями.
- Общие характеристики субъектов, типа уровня знаний (образования), знания языков, возраста и т.д. Эти характеристики могут влиять на подробности интерфейса пользователя, тип шрифта и языка.

Эти характеристики используются, прежде всего, при идентификации граничных классов и



прототипа, чтобы гарантировать лучшее соответствие применимости между семейством пользователя и проектом интерфейса пользователя.

Пример:

Ниже приводится пример характеристик субъекта Пользователь почты. Этот субъект, среди прочего, взаимодействует с прецедентом Управление входящим сообщением почты.

- Пользователь почты - опытный пользователь ПК.
- Рабочая среда Пользователя почты - обычно тихий офис.
- Целевое число пользователей почты – ок. 500,000.

3. Связи

Перечисляются связи, включающие субъекта.

- Для связи-ассоциации включаются ее краткое описание и связанные прецеденты. Включаются также их кратность и навигация из роли, включающей рассматриваемый субъект.
- Для обобщения включается его краткое описание и связанный субъект-предок.

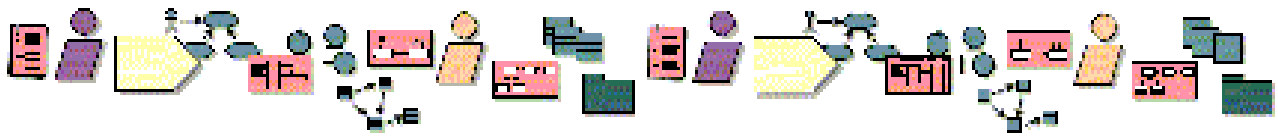
4. Диаграммы

Диаграммы субъекта и его связей.

Контрольные точки

Качество артефактов Субъект необходимо оценить на их соответствие следующим контрольным точкам:

- Все ли субъекты найдены? То есть, все ли роли в среде системы Вы объяснили и смоделировали? Хотя Вы и должны проверить это, Вы не можете дать окончательного ответа, пока Вы не нашли и не описали все прецеденты.
- Каждый ли субъект связан, по крайней мере, с одним прецедентом? Удалите все субъекты, не упомянутые в описаниях прецедентов, не имеющие связей-ассоциаций с прецедентами. Однако, субъект, упомянутый в описании прецедента, вероятно, будет иметь связь-ассоциацию с этим прецедентом.
- Можете ли Вы назвать, по крайней мере, двух людей, которые могли бы играть роль конкретного субъекта? Если нет, проверьте, не является ли роль субъекта частью другой роли. Если да, то объедините субъект с другим субъектом.
- Имеются ли субъекты, которые играют относительно системы подобные роли? Если да, Вы должны объединить их в один субъект. Связи-ассоциации и описания прецедента показывают, как взаимодействуют субъекты и система.
- Имеются ли два субъекта, играющие относительно прецедента одну и ту же роль? Если да, Вы должны использовать обобщение субъектов, чтобы моделировать их одинаковое



поведение.

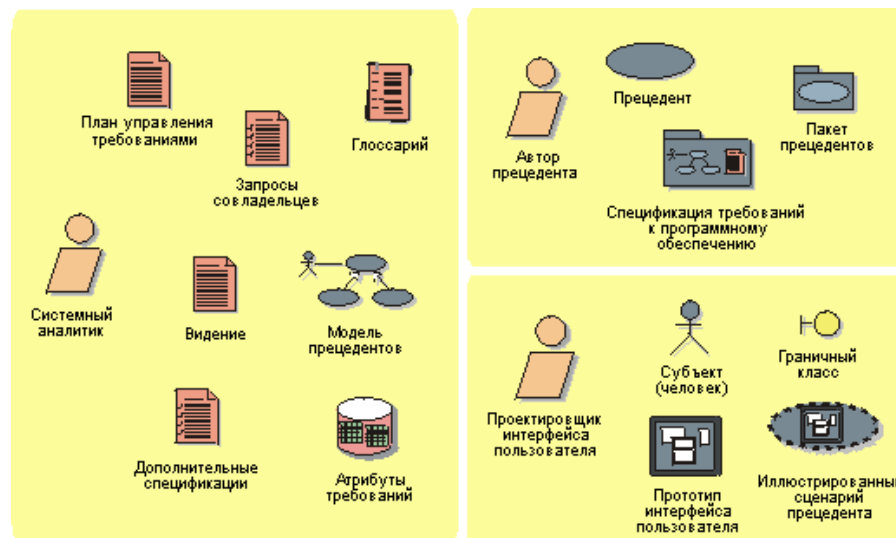
- Будет ли конкретный субъект использовать систему несколькими (полностью различными) способами, или он имеет несколько (полностью различных) целей в использовании прецедента? Если да, то Вы, вероятно, имеете больше одного субъекта.
- Имеют ли субъекты интуитивно понятные и подробные названия? Пользователи и заказчики понимают эти названия? Важно, чтобы название субъекта соответствовало его роли. Если нет, измените название.

Артефакт: План управления требованиями

Краткий обзор

Артефакт План управления требованиями описывает документацию требований, типы требований и соответствующие им атрибуты требований, определяет информацию и механизмы управления, которые будут использоваться для измерения, составления отчетов и управления изменением требований к изделию.

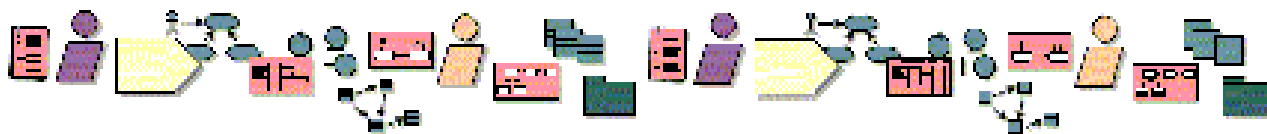
Этот очень важный артефакт, с которого, собственно, должна начинаться работа по управлению требованиями, является вложением в составной артефакт План разработки программного обеспечения (см. иллюстрацию на стр. 6-11). В предыдущих версиях План управления требованиями не выделялся в самостоятельный артефакт. Теперь эта несправедливость устранена:



Артефакты потока работ «Требования» Rational Unified Process 2000. Сравните с иллюстрацией на стр. 12-1.

Синхронизация

Артефакт План управления требованиями разрабатывается в стадии Начало и модифицируется в каждой главной вехе.



Создание и модификации артефакта происходят в результате следующих действий Rational Unified Process:

- Разработка плана управления требованиями
- Управление зависимостями

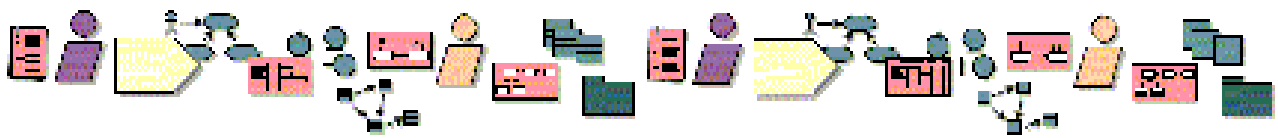
Артефакт предоставляет информацию для следующих действий:

- Составление плана разработки программного обеспечения
- Детализация прецедента
- Детализация требований к программному обеспечению
- Внедрение адаптированного процесса
- Управление зависимостями

Отчет

Интерактивная версия Rational Unified Process содержит predetermined шаблон и инструкции по его использованию для артефакта План управления требованиями. Ниже приводится его краткая схема:

1. **Введение**
 - 1.1 **Цель**
 - 1.2 **Контекст**
 - 1.3 **Определения, акронимы и сокращения**
 - 1.4 **Ссылки**
 - 1.5 **Краткий обзор**
2. **Артефакты требований и типы требований**
3. **Атрибуты требований**
 - 3.1 **Атрибуты для <тип требования>**
 - 3.1.1 **Статус**
 - 3.1.2 **Выгода**
 - 3.1.3 **Объем работ**
 - 3.1.4 **Риск**
 - 3.1.5 **Стабильность**



3.1.6 Целевой выпуск

3.1.7 Кому назначен

3.1.8 Причина

4. Критерии трассируемости

4.1 Критерий для <тип требования>

Работник: Автор прецедента

Человек, действующий как автор прецедента, должен иметь хорошие навыки общения с людьми и навыки записи результатов бесед. Знание прикладной области желательно, но не необходимо для всех. Автор прецедента должен быть знаком со всеми инструментальными средствами и иметь опыт фиксации результатов потока работ Требования

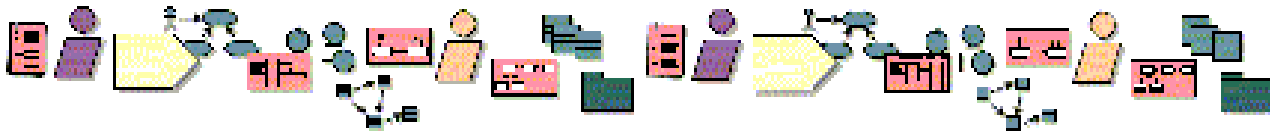
Артефакт: Прецедент

Краткий обзор

Прецедент определяет набор экземпляров прецедента, где каждый экземпляр – это последовательность действий, выполняемых системой, которая выдает наблюдаемый результат, ценный для конкретного субъекта.

Прецеденты используют следующие категории совладельцев и участников работ:

- **Заказчики** используют прецеденты, чтобы понять поведение системы, и, так как они должны согласовывать поток событий прецедента, заказчики используют прецеденты при рассмотрении результатов моделирования.
- **Потенциальные пользователи** используют прецеденты, чтобы понять поведение системы.
- **Архитекторы** используют прецеденты, чтобы идентифицировать архитектурно существенные функциональные возможности.
- Те, кто **анализирует, проектирует и выполняет** систему, используют прецеденты, чтобы понять требуемое поведение системы и уточнять систему.
- **Проектировщики прецедентов** используют потоки событий прецедентов для поиска классов. (Для проектировщиков прецедентов они являются наиболее важными артефактами.)
- **Тестировщики** используют прецеденты как базу для идентификации вариантов тестирования.
- **Администраторы** используют прецеденты для планирования и отслеживания моделирования прецедентов.
- **Авторы документов** используют прецеденты, чтобы понять, как описывать в документации



последовательность использования системы (типа руководства пользователя).

Толкование

Определение понятия прецедента и подробное его толкование уже приводилось в главе 4, при обсуждении основных концепций Rational Unified Process (см. стр. 4-5).

Синхронизация

Прецеденты идентифицируются и по возможности кратко описываются как можно раньше в стадии Начало, чтобы помочь в определении возможностей системы. Прецеденты, необходимые для выполнения анализа или архитектурного проектирования системы, более подробно описываются в стадии Уточнение. Оставшиеся прецеденты подробно описываются в стадии Конструирование.

Поиск, создание и модификации прецедентов происходят в результате следующих действий Rational Unified Process:

- Поиск субъектов и прецедентов
- Структурирование модели прецедентов
- Детализация прецедента
- Обзор требований

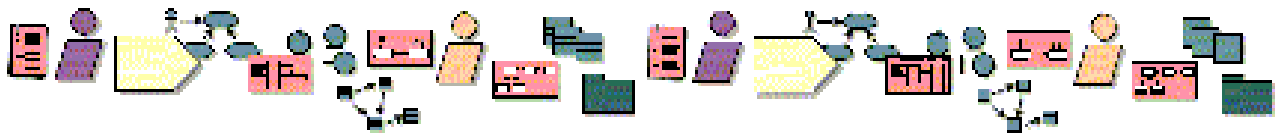
Прецедент предоставляет информацию для следующих действий:

- Фиксация общего словаря
- Детализация прецедента
- Детализация требований к программному обеспечению
- Моделирование интерфейса пользователя
- Обзор требований
- Структурирование модели прецедентов
- Анализ прецедента
- Проектирование прецедента

Как искать прецеденты

Ниже представлен набор вопросов, на которые полезно ответить при идентификации прецедентов:

- Для каждого ли субъекта определены все его задачи, для решения которых должна использоваться система?
- Нужно ли информировать субъекта относительно наступления некоторых событий в системе?



- Должен ли субъект сообщать системе о внезапных внешних изменениях?
- Обеспечивает ли система правильное поведение в деловой сфере?
- Все ли возможности могут быть выполнены идентифицированными Вами прецедентами?
- Какие прецеденты поддерживают и обслуживают систему?
- Какая информация должна изменяться или создаваться в системе?

Прецеденты, которые часто пропускаются, так как они не отражают первичные функции системы, могут быть следующего вида:

- Запуск и останов системы.
- Эксплуатация системы. Например, добавление новых пользователей и установка параметров пользователя.
- Эксплуатация данных, хранящихся в системе. Например, система создана для работы параллельно с наследуемой системой, и информация должна быть синхронизирована между ними.
- Функциональные возможности, необходимые для изменения поведения системы. Примером может служить функциональная возможность создания новых отчетов, а не только жесткого набора определенных отчетов о данных, хранящихся в системе.

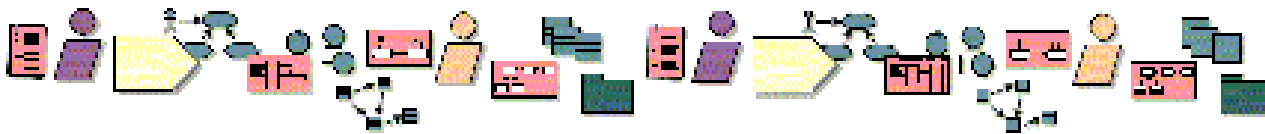
Если Вы разработали модели деловых прецедентов и деловых объектов, используйте имеющуюся в них информацию (см. стр. 10-17).

Как эволюционируют прецеденты

В ранних итерациях стадии Уточнение детально описывают только несколько прецедентов, которые рассматриваются как архитектурно существенные. Всегда сначала нужно разрабатывать схему прецедента (в пошаговом формате), прежде, чем углубляться в подробности. Эта пошаговая схема должна быть вашей первой попыткой определения структуры потока событий прецедента (см. раздел Поток событий - структура на стр. 14-17). Всегда начинайте с основного потока прецедента. Когда достигается некоторое соглашение по схеме основного потока, Вы можете добавлять то, что вводят альтернативные потоки относительно основного потока.

К концу уточнения все прецеденты, которые Вы планируете описать в деталях, должны быть закончены.

В вашей модели будут встречаться прецеденты, которые настолько просты, что они не нуждаются в детальном описании потока событий, вполне достаточно и пошаговой схемы. Критерием для принятия решения является то, что Вы не видите непонимания среди читателей, и что проектировщики и тестировщики согласны с уровнем детализации, предоставляемым пошаговым форматом. Примеры – прецеденты, которые описывают простой ввод или поиск некоторых данных в системе.



Контекст прецедента

Часто трудно решить, рассматривать ли некоторый набор взаимодействий системы и пользователя, как один или как несколько прецедентов. Рассмотрите использование рециркуляторной машины. Заказчик ставит пустую тару, типа банок, баллонов и контейнеров, в рециркуляторную машину. Когда он поставил всю имеющуюся тару, он нажимает кнопку, после чего печатается чек. Теперь он может обменять этот чек на деньги.

Как быть? Считать одним прецедентом установку тары и другим – затребование чека? Или все это – один прецедент? Происходит два действия, но одно без другого имеет для заказчика небольшое значение. Скорее всего, это – один законченный диалог с установкой тары и получением чека, который является ценным для заказчика (и имеет для него смысл). Таким образом, законченный диалог, от установки первого предмета до нажатия кнопки и получения чека, является законченным прецедентом.

Кроме того, Вы захотите сохранить эти два действия вместе, иметь возможность делать их одновременный обзор, вместе их изменять, тестировать, составлять для них справочники и вообще управлять ими как модулем. Это становится очевидным в больших системах.

Как реализуются прецеденты

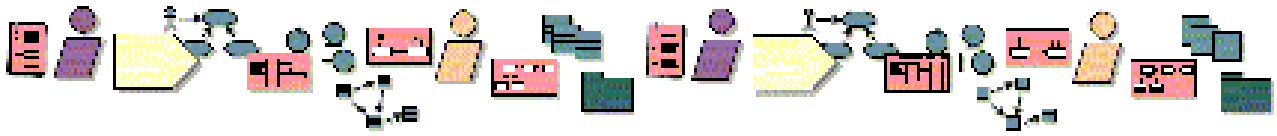
Прецедент описывает то, что происходит с системой, когда субъект взаимодействует с ней для выполнения прецедента. Прецедент не определяет то, как внутри системы выполняются ее задачи в терминах сотрудничества объектов. Это должна показывать реализации прецедента.

Пример:

В примере с телефоном, прецедент указал бы – среди прочего – что когда абонент поднимает трубку, система дает гудок, затем принимает набор цифр, находит сторону приема, там звонит телефон, при подъеме трубки происходит соединение, передается разговор и так далее.

В выполняемой системе образец прецедента не соответствует какому-либо конкретному объекту в модели выполнения (например, образцу класса в коде). Вместо этого, он соответствует определенному потоку событий, который вызван субъектом и выполнен как последовательность событий в наборе объектов. Другими словами, образцы прецедентов соответствуют поддерживающим связь образцам реализованных объектов. Мы называем это реализацией прецедента. Часто одни и те же объекты участвуют в реализации более, чем одного прецедента. Например, прецеденты Депонирование и Снятие со счета в банковской системе могут использовать в своих реализациях некоторый объект Счет. Но это не подразумевает, что два прецедента связаны, просто в своих реализациях они используют один и тот же объект.

Вы можете рассматривать поток событий как состоящий из нескольких подпотоков, которые вместе образуют суммарный поток событий. Вы можете многократно использовать описание подпотока в потоках событий других прецедентов. Подпотоки в описании потока событий одного прецедента могут быть общими с таковым других прецедентов. При проектировании Вы должны



иметь одни и те же объекты, выполняющие это общее поведение для всех соответствующих прецедентов; то есть это поведение должен выполнить только один набор объектов, независимо от того, который из прецедентов выполняется.

Пример:

В прецедентах Снятие денег и Проверка баланса системы банкомата начальный подпоток один и тот же. Поток событий обоих прецедентов начинается с проверки допустимости кредитной карты и персонального кода пользователя.

Экземпляр прецедента может следовать почти неограниченным, но счетным числом путей. Эти пути определяют выборы, обозначенные для экземпляра прецедента в описании потока событий. Выбранный путь зависит от событий. Типы событий включают:

- Ввод от субъекта. Например, субъект может выбрать из нескольких предложенных вариантов, что делать дальше.

Пример:

В прецеденте Возврат предметов в системе рециркуляторной машины Заказчик всегда имеет две возможности: его рука все еще ставит в машину очередной предмет или нажимает кнопку и получает чек.

- Проверка значения или типа внутреннего объекта или атрибута. Например, поток событий может отличаться, если значение больше или меньше, чем некоторое значение.

Пример:

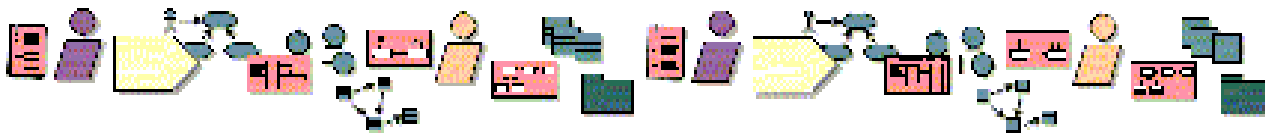
В прецеденте Снятие денег в системе банкомата поток событий будет иным, если Пользователь просит выдать денег больше, чем он имеет на своем счете. Таким образом, экземпляр прецедента будет следовать различными путями.

Параллельное выполнение экземпляров прецедента

Экземпляры нескольких прецедентов и несколько экземпляров одного и того же прецедента работают одновременно, если система это позволяет. При моделировании прецедента Вы можете предполагать, что экземпляры прецедентов могут без конфликта быть одновременно активными. Проектная модель, как ожидается, решит эту проблему, потому что модель прецедента не описывает, как система работает. Один из способов рассмотрения состоит в том, чтобы предполагать, что одновременно активен только один экземпляр прецедента и что выполнение этого экземпляра является элементарным действием. При моделировании прецедента предполагается, что машина интерпретации работает бесконечно быстро, так, чтобы преобразование экземпляров прецедента в последовательную цепочку не было проблемой.

Название и краткое описание

Каждый прецедент должен иметь название, которое указывает на то, что достигается его взаимодействием с субъектом (субъектами). Название, вероятно, будет состоять из нескольких слов, которые будут понятны. Никакие два прецедента не могут иметь одно и то же название.



Пример:

Здесь представлены варианты названия для прецедента Возврат предметов в примере с рециркуляторной машиной:

- Прием предметов залога
- Возврат предметов залога
- Предметы залога

Краткое описание прецедента должно отражать его роль и назначение. Когда Вы записываете описание, обратитесь к субъектам, включенным в прецедент, к глоссарию и, если необходимо, определите новые понятия.

Пример:

Ниже представлены примеры краткого описания прецедентов Возврат предметов и Добавление нового типа баллона в системе рециркуляторной машины:

Возврат предметов: Пользователь использует машину для автоматического подсчета всех сданных предметов залога (баллонов, банок и контейнеров) и получения чека. Чек должен быть оплачен в кассе.

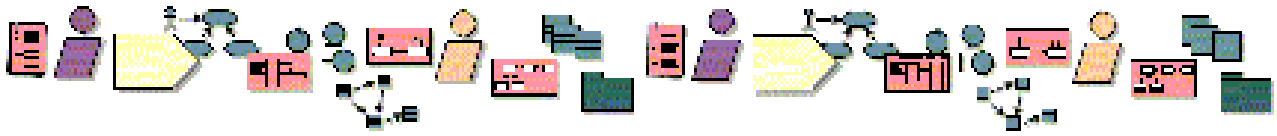
Добавление нового типа баллона: Новые типы баллонов могут быть добавлены к системе запуском «режима обучения» и вставкой пяти баллонов этого типа точно так же, как при обычном возвращении предметов. Машина может измерить баллоны и «научиться» их идентифицировать. Администратор определяет сумму возмещения для нового типа баллонов.

Поток событий – содержание

Поток событий прецедента содержит наиболее важную информацию, производимую при моделировании прецедента. Он должен описывать поток событий прецедента достаточно ясно и просто для постороннего. Помните, что поток событий должен представлять то, что делает система, а не то, как сконструирована система, чтобы выполнять требуемое поведение.

Рекомендации по содержанию потока событий:

- Опишите, как прецедент начинается и заканчивается
- Опишите, какими данными обмениваются субъект и прецедент
- Не описывайте подробности интерфейса пользователя, если это не необходимо для понимания поведения системы. Например, когда известно заранее, что приложение будет базироваться на сетевой технологии, полезно использовать ограниченный набор сетевой терминологии,. Иначе текст прецедента может показаться слишком абстрактным. При изложении используйте слова «переходит», «просматривает», «гиперсвязь», «страница», «подтверждает» и «окно просмотра». Однако, не желательно включать в описание слова «фрейм» или «web-страница», т.к. этим Вы высказываете предположения относительно проектного решения.

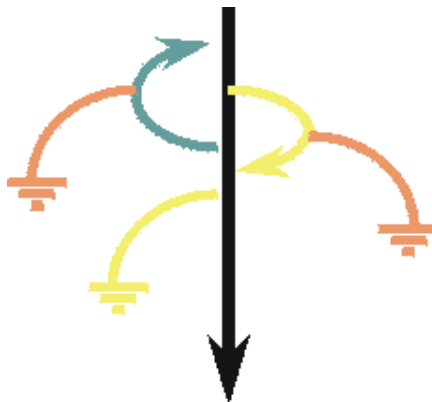


- Описывайте поток событий, а не только функциональные возможности. Чтобы выполнить это, начинайте описание каждого действия со слов «Когда субъект ...»
- Описывайте только те события, которые принадлежат прецеденту, а не то, что случается в других прецедентах или вне системы
- Избегайте неопределенной терминологии типа «например», «и т.д.» или «информация о ...»
- Детализируйте поток событий. Нужно ответить на все «когда» и «если». Помните о тех проектировщиках испытаний, которые будут использовать этот текст для идентификации вариантов тестирования.

Если Вы использовали некоторые термины в других прецедентах, убедитесь, что были использованы точно те же термины, и что их значение является точно тем же. Чтобы управлять общими терминами, поместите их в глоссарий.

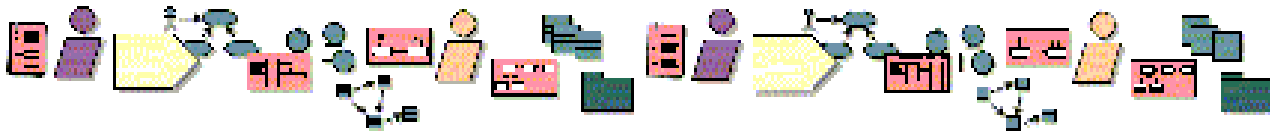
Поток событий – структура

Поток событий может содержать **основной поток событий** и **альтернативные потоки событий**. Основной поток событий должен отражать то, что «обычно» случается при выполнении прецедента. Альтернативные потоки событий отображают поведение необязательного или исключительного характера относительно нормального поведения, а также вариации нормального поведения. Вы можете рассматривать альтернативные потоки событий как «окольные пути» относительно основного потока событий, некоторые из которых возвратятся к основному потоку событий, а некоторые – закончат выполнение прецедента.



Типичная структура потока событий. Прямая стрелка представляет основной поток событий, а кривые – альтернативные пути относительно основного. Некоторые альтернативные пути возвращаются к основному потоку событий, в то время как другие заканчивают прецедент.

И основной поток событий, и альтернативные потоки должны быть далее структурированы в шаги или подпотоки. При этом ваша главная цель должна состоять в составлении понятного текста (см. также раздел Поток событий - стиль ниже). Практическое правило состоит в том, что подпоток должен быть сегментом поведения в пределах прецедента, который имеет ясную цель



и атомарность в том смысле, что Вы выполняете или все или ни одного из описанных действий. Вы можете иметь несколько уровней подпотоков, но если это возможно, лучше стараться избегать этого, так как усложняется понимание текста. Вы можете иллюстрировать структуру потока событий с помощью диаграммы действий (см. стр. 14-26).

Этот способ написания текста, структурированного в последовательные подразделы, рассчитан на читателя, который знает порядок следования подпотоков. Чтобы избежать недоразумений, Вы должны всегда указывать, установлен ли порядок подпотоков или нет. Соображения этого вида часто имеют связь с:

- Деловыми правилами. Например, права пользователя должны быть проверены прежде, чем система сделает некоторые данные доступными.
- Проектом пользовательского интерфейса. Например, система не должна приводить в исполнение некоторую последовательность поведения, которая может быть интуитивной для одних, но не для других пользователей.

Чтобы показать в структуре место альтернативного потока событий, Вы должны представить следующую информацию для каждого «окольного пути» относительно основного потока событий:

- Где в основном потоке событий может быть вставлено альтернативное поведение.
- Условие, которое должно быть выполнено, чтобы началось альтернативное поведение.
- Как и где продолжается основной поток событий, или как заканчивается прецедент.

Пример:

Здесь описан альтернативный подпоток прецедента Возврат предмета в системе рециркуляторной машины.

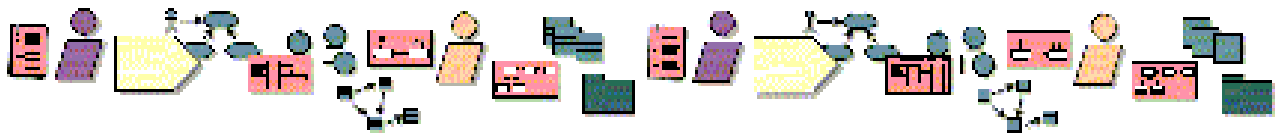
2.1. Баллон застрял

Если на шаге 1.5, Вставить возвращаемый предмет, баллон застревает в гнезде, датчики гнезда обнаружат эту проблему. Конвейерная лента останавливается и машина дает сигнал тревоги для вызова оператора. После устранения проблемы машина продолжает выполнение основного потока с шага 1.9.

В этом примере альтернативный поток событий вставлен в определенном месте основного потока событий. Но бывают такие альтернативные потоки событий, которые могут быть вставлены больше, чем в одном месте, а некоторые могут быть вставлены даже в любом месте основного потока событий.

Пример:

Ниже описан альтернативный подпоток прецедента Возврат предмета в системе рециркуляторной машины:



2.2. Передняя панель снята

Если кто-то снимет переднюю панель рециркуляторной машины, машина перестает работать и дает сигнал тревоги оператору. Когда передняя панель снова закрыта, машина продолжает работу с того места в основном потоке событий, где она была остановлена.

Если альтернативный поток событий достаточно прост, то может возникнуть соблазн описывать его в разделе основного потока событий, используя некоторую неофициальную конструкцию типа «условного оператора». Этого нужно избегать. Слишком много альтернатив сделают плохо различимым нормальное поведение. Кроме того, включение в разделы основного потока событий альтернативных путей может сделать текст «псевдокодоподобным» и трудно читаемым.

В общем случае, извлекая части потока событий и описывая эти части отдельно, можно увеличивать разборчивость основного потока событий и улучшать структуру прецедента и модели прецедентов. Вы можете моделировать извлеченные части:

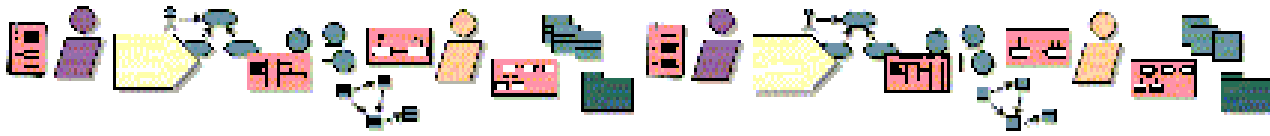
- Как альтернативный поток событий в пределах основного прецедента, если это – просто вариант, выбор или исключительная ситуация в основном потоке событий.
- Как явное включение в основной прецедент, если это – что-то, что Вы хотите сформировать так, чтобы его могли многократно использовать другие прецеденты.
- Как неявное включение в основной прецедент, если основной поток событий основного прецедента закончен, то есть имеет определенные начало и конец. Расширяющий поток делается тогда, когда Вы предпочитаете не показывать его в описании основного прецедента, чтобы не делать этот прецедент слишком сложным.
- Как подпоток в основном потоке событий, возможно как другой выбор, если не применяется ни одна из вышеупомянутых альтернатив. Например, в прецеденте Поддержка информации служащего могут быть отдельные подпоток для добавления, удаления и изменения информации о служащем.

Поток событий – стиль

Для описания прецедентов существует множество стилей. Для примера мы показываем основной поток событий прецедента Администрирование регламента, описанный тремя различными стилями, отличающимися, прежде всего, степенью формальности. Первый стиль, показанный в примере 1, рекомендуется потому, что он прост для понимания, и порядок, в котором случаются различные вещи, очевиден. Текст разделен на пронумерованные и имеющие названия подразделы. Нумерация упрощает ссылки на подразделы. Названия позволяют читателю быстро делать краткий обзор потока событий, просматривая текст и читая только заголовки.

Описание потока событий в примере 2 не в состоянии указать порядок, в котором случаются различные вещи. Если Вы запишете в этом стиле, Вы и другие можете упустить некоторые важные вещи, которые касаются системы.

Другой стиль, показанный в примере 3, может быть полезен, если Вам трудно выразить



последовательность событий. Этот стиль псевдокода более точен, но текст получается сложным для чтения и понимания человеком, не обладающим навыками программирования, особенно, если Вы хотите добиться быстрого понимания потока событий.

Пример 1:

1.1. Начало прецедента

Прецедент начинается, когда субъект Оператор требует от системы создать регламент измерения. Система ищет все элементы сети субъекта, их измеряемые объекты и соответствующие функции измерения, которые доступны этому конкретному Оператору. Доступные элементы сети – те, которые находятся в работе, и на обращение к которым Оператор имеет полномочия. Доступность функций измерения зависит от того, что было установлено для конкретного типа измеряемого объекта.

1.2. Конфигурирование регламента измерения

Система позволяет субъекту Оператор выбирать элементы сети для измерения, а затем показывает, какие измеряемые объекты доступны для выбранных элементов сети. Система позволяет Оператору выбрать измеряемые объекты, а затем выбрать, какие функции измерения устанавливать для каждого измеряемого объекта.

Система позволяет Оператору вводить текстовый комментарий относительно регламента измерения.

Оператор сообщает системе о завершении формирования регламента измерения. Система отвечает, генерируя уникальное название для регламента измерения и устанавливая значения по умолчанию для частоты и продолжительности измерения. Значения по умолчанию уникальны для каждого Оператора. Затем система позволяет Оператору редактировать эти значения по умолчанию.

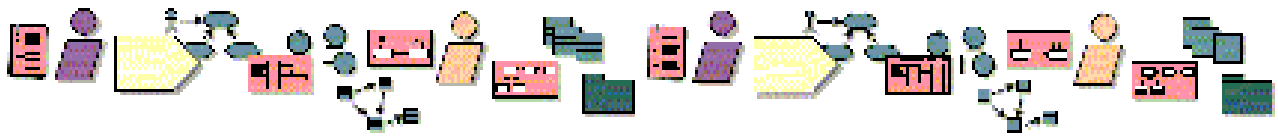
1.3. Инициализация регламента

Оператор требует от системы инициализировать регламент измерения. Система делает запись регламента измерения с указанием идентификатора создавшего Оператора, даты создания и статуса «Scheduled» (Намечен).

1.4. Завершение прецедента

Система подтверждает Оператору инициализацию регламента измерения и делает регламент измерения доступным для рассмотрения другими субъектами.

Описание прецедента. В этом стиле текст прост для чтения и поток событий легко проследить. Стремитесь к этому стилю в ваших описаниях.



Пример 2:

Операторы могут создавать Регламент сбора данных измерения элементов сети.

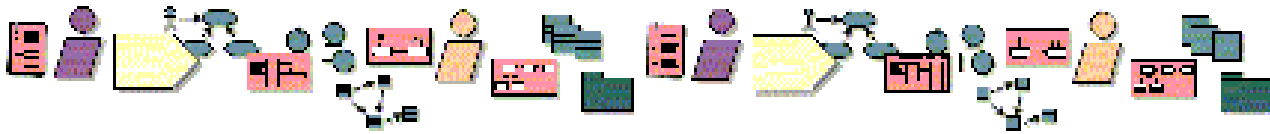
Система назначает Регламенту уникальное название и устанавливает значения по умолчанию, которые указывают длину и продолжительность измерения, а также как часто оно должно повторяться. Оператор должен иметь возможность редактировать эти значения.

Оператор должен далее определить, как соответствуют друг другу функции измерения, элементы сети и измеряемые объекты. Оператор может также добавлять к регламенту персональный комментарий.

Когда необходимая информация определена, создается и инициализируется новый Регламент с определенными атрибутами, идентификатором создателя и датой создания. Статусу регламента будет установлено значение «scheduled» (Намечен). Возможные значения статусов: Scheduled (Намечен), Executing (Выполняется), Completed (Закончен), Canceled (Отменен) и Erroneous (Ошибочный)

После этого интерфейс пользователя сообщает, что новый Регламент был создан и может быть отображен по запросу.

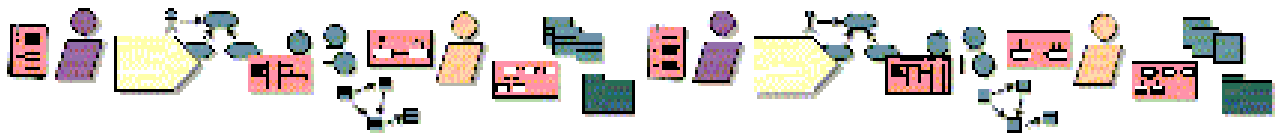
Описание прецедента: Этот стиль читаем, но не содержит ясного потока событий.



Пример 3:

```
"Администрирование регламента" (Идентификатор пользователя)
REPEAT
    <='Отображается меню администрирования регламента'
    IF (=> "Создание регламента" (Функция измерения,
        элемент сети, измеряемый объект)) THEN
        Система находит уникальное название и устанавливает по
        умолчанию когда и как долго должно выполняться измерение.
        <= "Отображается регламент" (Атрибуты по умолчанию)
        REPEAT
            =>'Редактирование регламента' (Назначение атрибутам
                новых значений)
            <= "Обновление экрана" (Новые атрибуты)
        UNTIL (Все атрибуты определены)
        REPEAT
            IF (=>'Редактирование регламента' (Назначение
                Атрибутам новых значений)
            THEN <= "Обновление экрана" (Новые атрибуты)
            ELSIF (=> "Сохранение регламента" (Идентификатор
                регламента, Атрибуты)) THEN
                Регламент создан и инициализирован в системе
                с определенными атрибутами, именем создателя,
                датой создания и статусом "scheduled" (Намечен).
                <= "Новый регламент создан" (Регламент)
            ENDIF
        UNTIL (=> "Завершение")
    ENDIF
UNTIL "Завершение администрирования регламента "
```

Описание прецедента: Здесь автор выбрал формальный стиль, используя псевдокод. Этот стиль делает понятными шаги процесса, и может быть полезен, если поток событий трудно описать словами.



Поток событий - пример

Законченное описание потока событий прецедента Управление регламентом, включая его альтернативные потоки, может выглядеть следующим образом:

1. Основной поток событий

1.1. Начало прецедента

Прецедент начинается, когда субъект Оператор требует от системы создать регламент измерения. Система ищет все элементы сети субъекта, их измеряемые объекты и соответствующие функции измерения, которые доступны этому конкретному Оператору. Доступные элементы сети – те, которые находятся в работе, и на обращение к которым Оператор имеет полномочия. Доступность функций измерения зависит от того, что было установлено для конкретного типа измеряемого объекта.

1.2. Конфигурирование регламента измерения

Система позволяет субъекту Оператор выбирать элементы сети для измерения, а затем показывает, какие измеряемые объекты доступны для выбранных элементов сети. Система позволяет Оператору выбрать измеряемые объекты, а затем выбрать, какие функции измерения устанавливать для каждого измеряемого объекта.

Система позволяет Оператору вводить текстовый комментарий относительно регламента измерения.

Оператор сообщает системе о завершении формирования регламента измерения. Система отвечает, генерируя уникальное название для регламента измерения и устанавливая значения по умолчанию для частоты и продолжительности измерения. Значения по умолчанию уникальны для каждого Оператора. Затем система позволяет Оператору редактировать эти значения по умолчанию.

1.3. Инициализация регламента

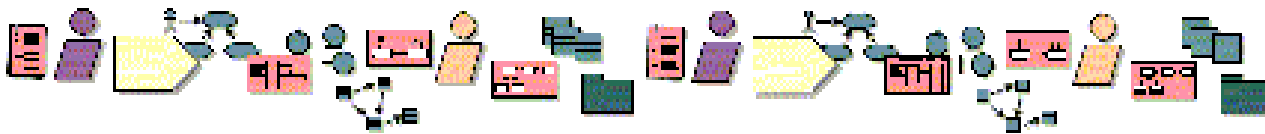
Оператор требует от системы инициализировать регламент измерения. Система делает запись регламента измерения с указанием идентификатора создавшего Оператора, даты создания и статуса «Scheduled» (Намечен).

1.4. Завершение прецедента

Система подтверждает Оператору инициализацию регламента измерения и делает регламент измерения доступным для рассмотрения другими субъектами.

2. Альтернативные потоки событий

2.1. Нет доступных элементов сети



Если на шаге 1.1, Начало прецедента, оказалось, что для этого Оператора сейчас нет доступных для измерения элементов сети, система сообщает об этом Оператору. После этого прецедент заканчивается.

2.2. Нет доступных функций измерения

Если на шаге 1.2, Конфигурирование регламента измерения, для выбранных элементов сети нет доступных функций измерения, система сообщит об этом Оператору и позволит Оператору выбирать другие элементы сети.

2.3. Отмена регламента измерения

Система позволит Оператору отменить все действия в любом пункте в процессе выполнения прецедента. В таком случае система возвратится в состояние, в котором она была до начала прецедента, и прецедент заканчивается.

Специальные требования

В специальных требованиях прецедента Вы описываете все требования прецедента, которые не отражаются потоком событий. Это требования, которые будут влиять на проектную модель, но не содержащие функциональности. Обсуждение требований, не содержащих функциональности, см. также на стр. 13-6. Вы можете организовывать эти требования в категориях типа Применимости, Надежности, Эффективности и Заменяемости, но обычно их так немного, что такая группировка не имеет особого значения.

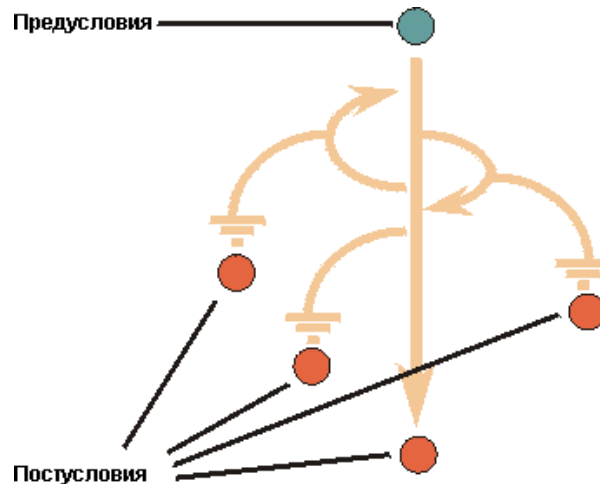
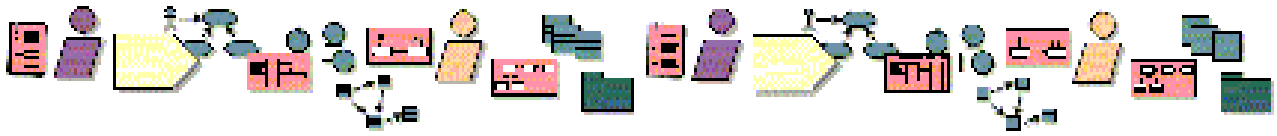
Пример:

В системе рециркуляторной машины специальным требованием прецедента Возврат предметов могло бы быть:

Машина должна быть способна опознавать предметы вложения с надежностью больше чем 95 процентов.

Предусловия и постусловия

Иногда для разъяснения того, как поток событий начинается и заканчивается, бывает полезно использовать понятия **предусловий** и **постусловий**. Однако, используйте их только, если это воспринимается аудиторией прецедента как полезная информация.



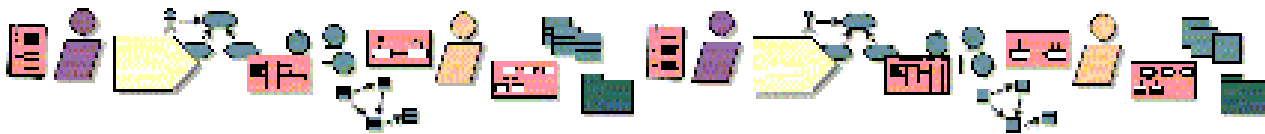
Предусловие – это состояние системы и ее окрестности, которое требуется прежде, чем прецедент может быть начат. Постусловие – это состояние, в котором будет система после того, как прецедент закончится.

Рассмотрите следующее:

- Состояния, описанные предусловиями или постусловиями, должны быть состояниями, которые пользователь может наблюдать. Состояния «Пользователь вошел в систему» или «Пользователь открыл документ» являются примерами наблюдаемых состояний.
- Предусловие – это ограничение для возможности начала прецедента. Прецедент начинается вовсе не событием.
- Предусловие для прецедента – это не предусловие только для одного подпотока, хотя Вы можете определять предусловия и постусловия и на уровне подпотоков.
- Постусловие для прецедента должно быть истинно независимо от того, который из альтернативных потоков был выполнен; оно не должно быть истинно только для главного потока. Если что-либо может потерпеть неудачу, Вы можете описать это в постусловии, говоря: «Действие закончено, или, если что-либо потерпело неудачу, действие не выполнено», а не только: «Действие закончено».
- Когда Вы используете постусловия вместе со связями расширения, Вы должны позаботиться, чтобы прецедент расширения не вводил подпоток, который нарушает постусловие в основном прецеденте.
- Постусловия могут быть мощным инструментом для описания прецедентов. Сначала Вы определяете то, чего прецедент, как предполагается, достигает, т.е. постусловия. Далее Вы можете описывать, как достичь этого состояния (необходимый поток событий).

Пример:

Предусловия для прецедента Снятие денег в системе банкомата: заказчик имеет



персонифицированную карту, которая соответствует считывающему устройству и зарегистрирована банковской системой.

Постусловие для прецедента Снятие денег в системе банкомата: в конце прецедента счет и данные транзакции сбалансированы, связь с банковской системой повторно инициализирована и заказчику возвращена его карта.

Точки расширения

Точка расширения открывает прецеденту возможность расширения. Она имеет название и список ссылок к одной или более точкам в пределах потока событий прецедента. Точка расширения может ссылаться на одно место между двумя шагами поведения в пределах прецедента. Или она может ссылаться на набор дискретных мест.

Использование названий точек расширения поможет Вам отделить технические требования поведения прецедента расширения от внутренних подробностей основного прецедента. Основной прецедент может изменяться или перестраиваться, но пока названия точек расширения остаются теми же, это не будет затрагивать прецедент расширения. В то же время, Вы не загружаете текст, описывающий поток событий основного прецедента, подробностями того, где его поведение могло бы быть расширено.

Пример:

В телефонной системе прецедент Телефонный вызов может быть расширен абстрактным прецедентом Показ вызывающего абонента. Это – необязательный сервис, часто упоминаемый как «Caller ID» (Идентификация номера телефона вызывающего абонента), который, возможно, потребуется стороной приема. Описание точки расширения в прецеденте Телефонный вызов может выглядеть следующим образом:

Название: Показ вызывающего абонента

Расположение: После раздела 1.9 Звонок телефона стороны приема.

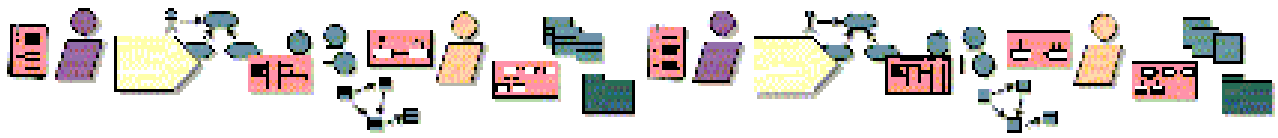
Диаграммы прецедентов

Вы можете проиллюстрировать связи прецедента с субъектами и с другими прецедентами с помощью диаграммы прецедентов (иногда – больше, чем одной диаграммы), принадлежащей прецеденту. Это полезно, если прецедент используется многими субъектами или имеет связи со многими другими прецедентами. Диаграмма этого вида имеет «локальный» характер, так как она показывает модель прецедентов с точки зрения только одного прецедента и не предназначена для объяснения общих фактов относительно всей модели прецедентов.

Диаграммы действий

Поток событий прецедента может быть описан графически с помощью диаграммы действий.

Мы подробно познакомились с диаграммами действий при описании основного потока работ Деловое моделирование (см. стр. 8-16, 9-11). Эти диаграммы показывают:



- Состояния действия, которые представляют выполняемое действие или шаг в пределах потока событий.
- Переходы, которые показывают последовательность действий. Этот тип перехода иногда называют переходом завершения, так как он отличается от перехода, в котором не требуется явного запуска события. Здесь событие запускается завершением действия, которое представляется состоянием действия.
- Решения, для которых определен набор условий защиты. Эта защита создает условия для контроля набора альтернативных переходов, которые следуют после окончания действия. Решения и условия защиты позволяют Вам показывать альтернативные потоки в потоке событий прецедента.
- Полосы синхронизации, которые Вы можете использовать, чтобы показать параллельные подпотоки. Полосы синхронизации позволяют Вам показывать конкурентные пути в потоке событий прецедента.

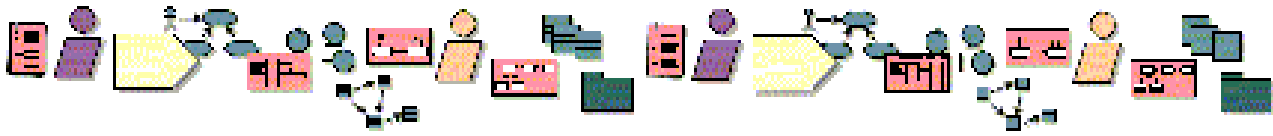
Артефакт: Пакет прецедентов

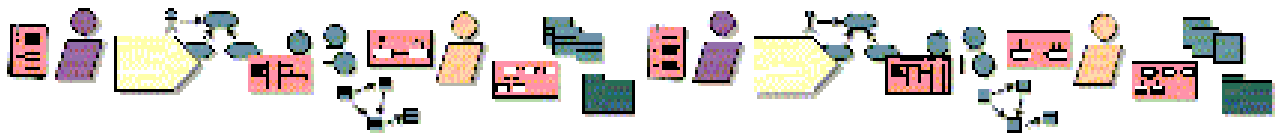
Пакет прецедентов – это совокупность прецедентов, субъектов, связей, диаграмм и других пакетов; он используется для структурирования модели прецедентов путем разделения ее на меньшие части.

Вы можете выделять разделы модели прецедентов в пакеты прецедентов по многим причинам:

- Вы можете использовать пакеты прецедентов для отражения очередности создания, конфигурации или поставки частей в законченной системе.
- Распределение ресурсов и компетентности различных групп разработки может потребовать, чтобы проект был разделен между ними. Некоторые пакеты прецедентов подходят для группы, а некоторые – для одного человека. Правильное разбиение позволяет разрабатывать пакеты наиболее эффективным способом. Вы должны убедиться, однако, что задачи каждого пакета определены так, что разработка может выполняться параллельно.
- Вы можете использовать пакеты прецедентов для структурирования модели прецедентов в соответствии с типами пользователей. Многие изменения требований происходят от пользователей. Пакеты прецедентов гарантируют, что изменения, поступившие от специфического типа пользователей, затронут только те части системы, которые соответствуют этому типу пользователей.
- В некоторых приложениях определенная часть информация должна иметь ограниченное распространение. Пакеты прецедентов позволяют Вам сохранять тайну в тех областях, где это необходимо.

В следующем, заключительном, выпуске, посвященном потоку работ Требования, мы обсудим артефакты, которые формирует Проектировщик интерфейса пользователя.





Поток работ «Требования» (окончание)

Артефакты интерфейса пользователя

Артефакты интерфейса пользователя занимают некоторое промежуточное положение между анализом и проектированием и управлением требованиями.

С одной стороны, эти артефакты содержат некоторые элементы проекта: подробное описание порядка взаимодействия пользователя с будущей системой, включающее рекомендации по применимости (иллюстрированные сценарии), наметки архитектуры интерфейса (граничные классы и связи между ними) и даже внешний вид диалоговых окон (прототипы).

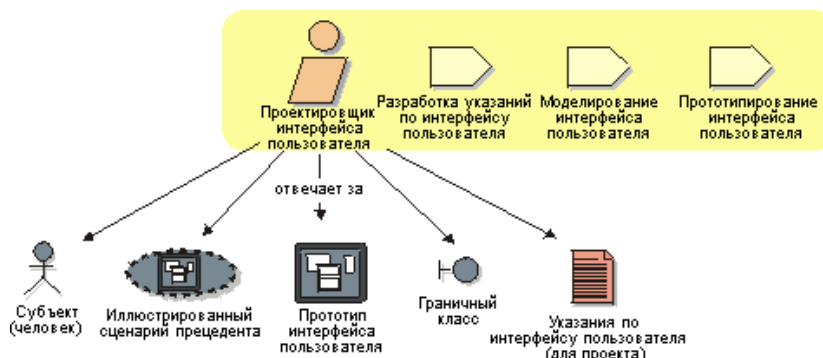
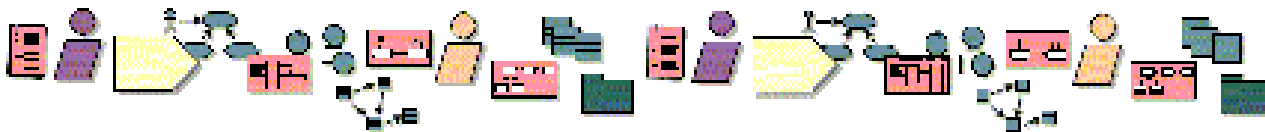
С другой стороны, именно при моделировании и прототипировании интерфейса важно возможно более тесно взаимодействовать с потенциальными пользователями системы. При этом применимость системы может исследоваться адресно, помогая раскрыть все предварительно неоткрытые требования и уточнить определения уже идентифицированных ранее.

Создание и поддержка артефактов интерфейса пользователя выполняются в потоке работ Требования (см. иллюстрацию на стр. 11-6), часть Уточнение определения системы (см. стр. 11-16). Ответственность за эту работу возлагается на Проектировщика интерфейса пользователя.

Работник: Проектировщик интерфейса пользователя

Проектировщик интерфейса пользователя выполняет и координирует прототипирование и проектирование интерфейса пользователя. Сюда входит:

- фиксация требований к интерфейсу пользователя, включая требования применимости;
- формирование прототипов интерфейса пользователя;
- включение других совладельцев интерфейса пользователя, типа конечных пользователей, в обзор практичности и в сеансы тестирования удобства использования;
- рассмотрение и обеспечение соответствующей обратной связи при окончательном выполнении интерфейса пользователя (созданного другими разработчиками, то есть проектировщиками и конструкторами).



В этой главе мы будем рассматривать действия Моделирование и Прототипирование интерфейса пользователя. Действие Разработка указаний по интерфейсу пользователя мы сейчас рассматривать не будем, т.к. оно выполняется в потоке работ Среда и будет рассмотрено в свое время.

Обратите внимание, что проектировщик интерфейса пользователя не должен сам выполнять интерфейс. Вместо этого, он должен сосредоточиться и заниматься только проектированием и «формированием видения» интерфейса, потому что:

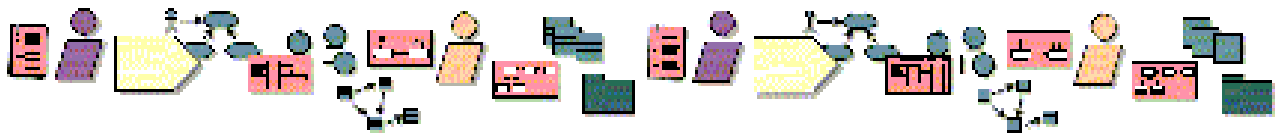
- Навыки, необходимые проектировщику интерфейса пользователя, часто должны быть усовершенствованы и оптимизированы для текущего проекта и вида приложения (с потенциально уникальными требованиями применимости), а это требует времени и сосредоточенности.
- Должен быть исключен риск «смещения зависимости», то есть проектировщик интерфейса пользователя не должен находиться под влиянием соображений выполнимости (в противоположность соображениям применимости).

Моделирование интерфейса пользователя

Целью этого действия является создание модели интерфейса пользователя, которая поддерживает обоснование его применимости.

Для каждого прецедента, приоритетного с точки зрения применения в текущей итерации, выполняются следующие шаги:

- Описание характеристик связанных субъектов
- Создание иллюстрированного сценария прецедента
- Описание потока событий иллюстрированного сценария
- Формулирование требования применимости к иллюстрированному сценарию прецедента
- Поиск необходимых граничных классов иллюстрированного сценария прецедента
- Описание взаимодействий между граничными объектами и субъектами



- Дополнение диаграммы иллюстрированного сценария прецедента
- Ссылка на прототип интерфейса пользователя из иллюстрированного сценария прецедента

Для идентифицированных граничных классов выполняются следующие шаги:

- Описание ответственности граничных классов
- Описание атрибутов граничных классов
- Описание связей между граничными классами
- Описание требований применимости к граничным классам
- Представление граничных классов на общих диаграммах классов
- Оценка результатов

Обратите внимание, что эти шаги представлены в логическом порядке, но Вам, вероятно, придется переставлять их или исполнять некоторых из них параллельно. Кроме того, некоторые шаги необязательны - это зависит от сложности конкретного интерфейса пользователя.

Прототипирование интерфейса пользователя

Целью этого действия является создание прототипа интерфейса пользователя.

Для каждого иллюстрированного сценария прецедента, который будет смоделирован в текущей итерации, выполняются следующие шаги:

- Проектирование прототипа интерфейса пользователя
- Выполнение прототипа интерфейса пользователя
- Установка обратной связи с прототипом интерфейса пользователя

Так же, как и в предыдущем действии, эти шаги перечислены в некотором логическом порядке, который может меняться.

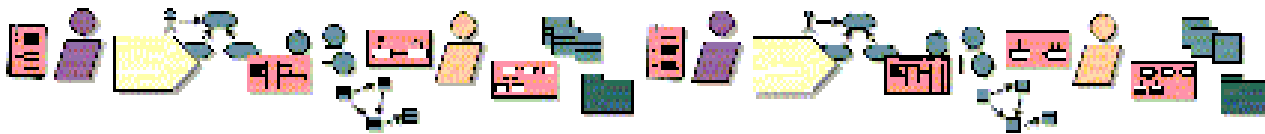
Сначала у Вас будет уходить много времени на проектирование прототипа; потом, когда Вы больше узнаете о нем, основное время будет уходить на выполнение и поддержку обратной связи.

Артефакт: Иллюстрированный сценарий прецедента

Краткий обзор

Иллюстрированный сценарий прецедента – это логическое и концептуальное описание того, как прецедент обеспечивается интерфейсом пользователя, включая взаимодействия между субъектом(ами) и системой.

Иллюстрированные сценарии прецедентов используют:



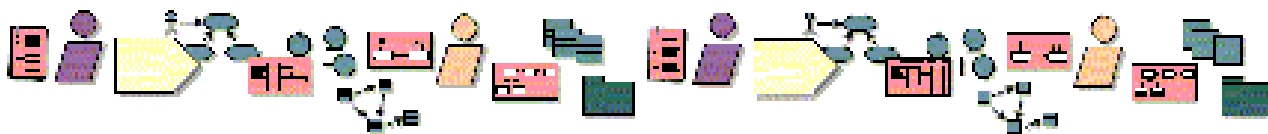
- **проектировщики интерфейса пользователя** при формировании модели интерфейса пользователя;
- **проектировщики** граничных объектов, участвующих в иллюстрированном сценарии прецедента, чтобы понять роли объектов в прецедентах и то, как эти объекты взаимодействуют. Они используют эту информацию при проектировании и выполнении граничных объектов (то есть при создании интерфейса пользователя);
- **те, кто проектируют следующую версию системы**, чтобы понять, как система выполняет поток событий в терминах граничных объектов. Например, изменение может затрагивать ограниченное число прецедентов, в этом случае проектировщики должны видеть реализацию их потоков событий;
- **тестировщики**, которые проверяют прецеденты системы;
- **руководитель** при планировании и отслеживании работ анализа и проектирования.

Толкование

Иллюстрированные сценарии прецедентов используются для выяснения и обсуждения требований интерфейса пользователя, включая требования применимости. Они обеспечивают высокоуровневое представление интерфейса пользователя, и разрабатываются намного быстрее, чем фактический интерфейс пользователя. Таким образом, иллюстрированные сценарии прецедентов могут использоваться для создания и обсуждения нескольких версий интерфейса пользователя прежде, чем он будет прототипироваться, разрабатываться и реализовываться.

Иллюстрированный сценарий прецедента описывается в терминах граничных классов и их статических и динамических связей, типа объединений, ассоциаций и ссылок. Каждый граничный класс — это, в свою очередь, высокоуровневое представление окна или подобной конструкции в интерфейсе пользователя. Преимущества такого подхода состоят в следующем:

- Он обеспечивает высокоуровневое представление статических связей между окнами, типа иерархии вместилищ окон, и других зависимостей между объектами в интерфейсе пользователя.
- Он обеспечивает высокоуровневое представление динамических связей между окнами, типа путей передвижения по окнам и других навигационных путей между объектами в интерфейсе пользователя.
- Он обеспечивает способ фиксации требований к каждому окну или подобной конструкции в интерфейсе пользователя, описывая соответствующий граничный класс. Описание каждого граничного класса определяет ответственности, атрибуты, связи и т.д., которые имеют прямое отображение в соответствующей конструкции интерфейса пользователя.
- Он обеспечивает трассировку к определенному прецеденту, и, как следствие, безшовную интеграцию с управляемым прецедентами подходом в разработке программного обеспечения. В результате интерфейс пользователя будет управляться прецедентами, которые должны быть



обеспечены системой, и ролями субъектов (пользователей) и их ожиданиями от этих прецедентов.



Иллюстрированный сценарий прецедента в модели анализа трассируется (взаимно-однозначно) к прецеденту в модели прецедентов.

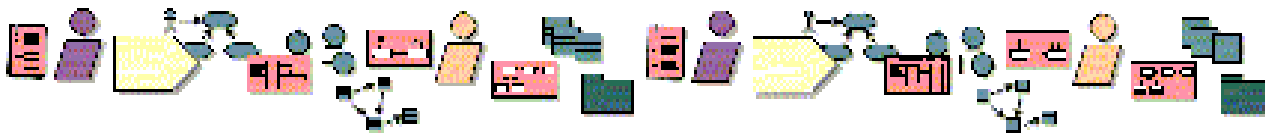
Работа над иллюстрированным сценарием прецедента обычно выполняется в следующей последовательности:

- Начальное описание потока событий иллюстрированного сценария
- Расширение описания аспектами применимости
- Создание диаграмм граничных классов
- Создание диаграмм взаимодействия граничных объектов
- Дополнение описания иллюстрированного сценария объектами из диаграмм
- Фиксация требований применимости
- Поддержка обратной связи с прототипом интерфейса пользователя

Начальное описание потока событий иллюстрированного сценария

Ниже представлены рекомендации того, как описывать поток событий иллюстрированного сценария:

- **Начните с разьяснения самого прецедента, а не его интерфейса пользователя.** Сохраните описание независимым от интерфейса пользователя, особенно, если прецедент неисследован. Впоследствии, когда прецедент будет понят, поток событий иллюстрированного сценария может быть расширен аспектами применимости интерфейса.
- **Сохраните описания действий краткими.** Описание не должно быть очень большим, потому что оно предназначено только для проектировщиков интерфейса пользователя. Описания действий как последовательности шагов обеспечивают лучший обзор, так как это делает описание короче. Например, когда Вы описываете, как прецедент обменивается данными с субъектом, постарайтесь сделать это кратко, но полно; передаваемые данные Вы можете перечислять в конце предложения в круглых скобках: «создать запись о человеке (имя, адрес, телефон)».
- **Избегайте последовательностей и режимов.** Субъекты (люди) часто могут выполнять взаимодействия с прецедентом в различных последовательностях, особенно в системах с интенсивным интерфейсом, где пользователь управляет системой. Последовательности часто



подразумевают режимы интерфейса пользователя. Вы должны избегать режимов, где это возможно. Вы должны определять последовательности только тогда, когда это обязательно в прецеденте. Например, что пользователь должен идентифицировать себя прежде, чем он может забрать деньги из банкомата, или что система должна показать счет пользователю прежде, чем пользователь сможет принять или отказаться от него.

- **Сохраняйте совместимость с прецедентом.** Так как иллюстрированный сценарий может описываться более или менее параллельно с соответствующим прецедентом, эти два артефакта должны сохранять совместимость и предоставлять обратную связь друг другу. В частности, поток событий иллюстрированного сценария должен быть совместим с потоком событий соответствующего прецедента. Обратите внимание, что это часто требует отлаженной связи между автором прецедента, ответственным за прецедент, и проектировщиком интерфейса пользователя, ответственным за иллюстрированный сценарий прецедента.

Пример:

Ниже приводится пример начального описания потока событий иллюстрированного сценария для прецедента Управление входящими сообщениями почты, прежде, чем это описание будет расширено аспектами применимости.

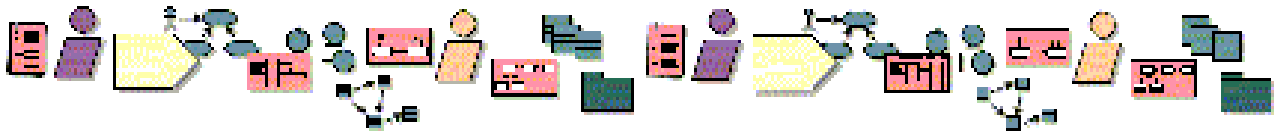
- а) Прецедент начинается, когда пользователь почты требует отобразить сообщения почты и система отображает сообщения.
- б) Пользователь почты может выполнить один или более из следующих шагов:
 - с) Упорядочить почтовые сообщения по отправителям или по темам.
 - д) Прочитать текст почтового сообщения.
 - е) Сохранить почтовое сообщение как файл.
 - ф) Сохранить приложение к сообщению как файл.
- г) Прецедент заканчивается, когда пользователь почты подает команду о завершении работы с почтовыми сообщениями.

По возможности используйте текст пошагового описания соответствующего прецедента как шаблон для создания начального описания потока событий иллюстрированного сценария.

Расширение описания аспектами применимости

Ориентиры

Действительно пригодная для использования система не только помогает пользователю, автоматизируя простые и повторяющиеся задачи, но также показывает ориентиры, обычно (неявно) предоставляя необходимую информацию для задач, которые не могут быть автоматизированы. Такие ориентиры могут обеспечиваться, например, справочными надписями или контекстной экранной справкой.



Эта важная для проектирования интерфейса пользователя информация может быть предоставлена путем указания на потребность в таком ориентировании в определенных точках потока событий прецедента.

Проектировщик интерфейса пользователя должен просмотреть поток событий и рассмотреть на каждом шаге следующие проблемы:

- В каких ориентирах может нуждаться пользователь?
- Какие ориентиры могла бы обеспечить система?
- Что может быть представлено как ориентиры и что не должно?

Описание потока событий может идентифицировать основные функциональные возможности системы. Пометки о полезных ориентирах помогают идентифицировать «необязательные» функциональные возможности, которые не критичны для пользователя в смысле ограничения его способности выполнить свою работу, но они могут помочь ему выполнить эту работу, снабжая его полезной информацией. Таким образом, что-либо, что может помочь найти такие необязательные функциональные возможности, должно быть представлено как полезные ориентиры. Однако, Вы не должны представлять как полезные ориентиры то, что только идентифицирует функциональные возможности, которые мы все равно нашли бы, используя хорошую практику формирования интерфейса пользователя (например, не представляйте как полезные ориентиры то, что система должна осуществлять обратную связь с пользователем при его действиях или показывать пользователю различные опции, которые он может выбрать, и т.д.).

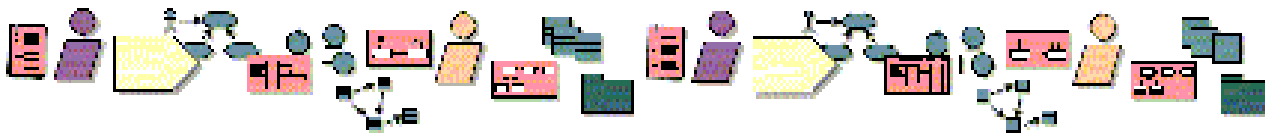
Полезные ориентиры можно использовать также и для того, чтобы сообщить о том, что не нужно показывать при формировании интерфейса, чтобы не загружать пользователя ненужной информацией.

Полезные ориентиры – это не «требования» в таком же смысле, как поток событий, это – скорее «пожелание» или «рекомендация». Когда Вы идентифицируете и описываете полезные ориентиры, Вы должны думать не в терминах того, что должна делать система, а в терминах того, что может понадобиться пользователю в дополнение к этому. Помните, что ориентиры не абсолютно необходимы, это просто средство улучшения применимости.

Пример:

Ниже приводится фрагмент описания потока событий иллюстрированного сценария для прецедента Управление входящими сообщениями почты, расширенного ориентирами (текст в пределах []).

- а) Прецедент начинается, когда пользователь почты требует отобразить сообщения почты и система отображает сообщения. [Пользователь должен иметь способ различать новые, прочитанные и непрочитанные сообщения; кроме того, пользователь должен видеть отправителя, тему и приоритет каждого сообщения.]



Ориентиры часто необходимы в действиях, когда пользователь должен принять решение. В следующих ситуациях часто используют полезные ориентиры:

- Они нетривиальны для пользователя.
- Они затрагивают человеческий ресурс или деловое окружение системы (люди или деловое окружение системы обычно означают деловую сферу, пользователя и задачу, которую пытается выполнить пользователь).
- Они понадобятся после того, как прецедент закончится.

Средние значения атрибутов и объемы объектов

Часто важно фиксировать средние значения атрибутов и объемы объектов, которые должны управляться или представляться пользователю. Тогда интерфейс пользователя может быть оптимизирован для этих средних значений и объемов.

Пример:

Ниже приводится фрагмент описания потока событий иллюстрированного сценария для прецедента Управление входящими сообщениями почты, расширенное средними значениями атрибутов и объемами (текст в пределах {}).

d) Прочитать текст почтового сообщения. {Текст сообщения в среднем содержит 100 символов.}

Средняя интенсивность использования

Средняя интенсивность использования фиксируется, чтобы найти действия, которые используются интенсивно в противоположность действиям, которые используются редко. В результате мы найдем в прецеденте и обычно используемые и редкие последовательности (потоки). Это – важная информация, которая поможет отсортировать по приоритетам, определить интенсивно используемые части интерфейса пользователя и сосредоточиться на его навигационных иерархиях (например, сформировать закладки или дополнительные инструментальные панели для выполнения наиболее частых действий).

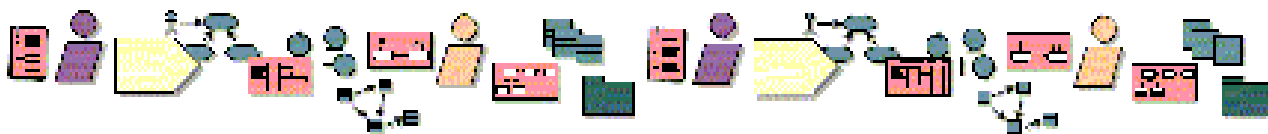
Пример:

Ниже приводится фрагмент описания потока событий иллюстрированного сценария для прецедента Управление входящими сообщениями почты, расширенное средними значениями частоты использования (текст в пределах ()).

d) Прочитать текст почтового сообщения. (Выполняется больше, чем в 75 % случаев.)

e) Сохранить почтовое сообщение как файл. (Выполняется меньше, чем в 5 % случаев.)

Вывод из этого описания – это то, что шаг d) нуждается в особой поддержке интерфейса



пользователя.

Результирующее описание потока событий иллюстрированного сценария

Пример:

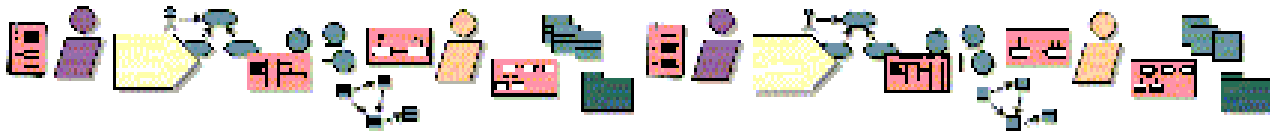
Ниже приводится пример финального описания потока событий иллюстрированного сценария для прецедента Управление входящими сообщениями почты, расширенное различными аспектами применимости.

- а) Прецедент начинается, когда пользователь почты требует отобразить сообщения почты и система отображает сообщения. [Пользователь должен иметь способ различать новые, прочитанные и непрочитанные сообщения; кроме того, пользователь должен видеть отправителя, тему и приоритет каждого сообщения.] {Среднее число одновременно показываемых непрочитанных сообщений почты – около 100; в 90 % случаев строка описания содержания сообщения – меньше 40 символов.}
- б) Пользователь почты может выполнить один или более из следующих шагов:
 - с) Упорядочить почтовые сообщения по отправителям или по темам. (Выполняется больше, чем в 60 % случаев.)
 - д) Прочитать текст почтового сообщения. {Текст сообщения в среднем содержит 100 символов.} (Выполняется больше, чем в 75 % случаев.)
 - е) Сохранить почтовое сообщение как файл. (Выполняется меньше, чем в 5 % случаев.)
 - ф) Сохранить приложение к сообщению как файл. [Пользователь должен быть способен видеть типы файлов приложений.] {В 95 % случаев сообщения имеют меньше двух приложений.}
- г) Прецедент заканчивается, когда пользователь почты подает команду о завершении работы с почтовыми сообщениями.

Таким образом, основная идея потока событий иллюстрированного сценария состоит в том, чтобы расширить описание потока событий различными аспектами применимости; затем эта информация используется при проектировании пригодного для использования интерфейса пользователя. Заметьте также, что аспекты применимости, которые показаны в примере выше, могут заменяться или расширяться другими аспектами, в зависимости от потребностей специфического типа приложения или технологии использования интерфейса пользователя.

Создание диаграмм граничных классов

Иллюстрированный сценарий прецедента реализуется граничными классами и их взаимодействующими объектами. Для отображения граничных классов (вместе с их связями), участвующих в иллюстрированном сценарии прецедента, мы создаем диаграмму классов и рассматриваем ее как часть описания иллюстрированного сценария прецедента.



Пример:

Ниже приводится диаграмма классов, принадлежащая иллюстрированному сценарию прецедента Управление входящими сообщениями почты:

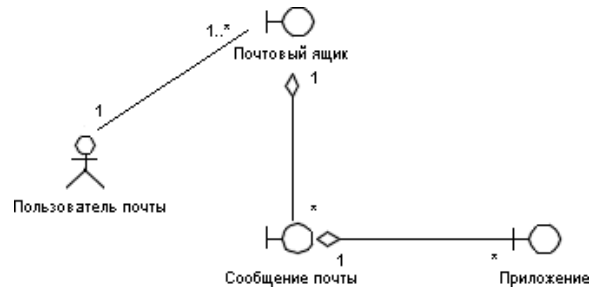


Диаграмма классов включает субъект Пользователь почты и граничные классы Почтовый ящик, Сообщение почты и Приложение. Мы объясняем здесь, что Пользователь почты взаимодействует со всеми объектами, содержащимися в иерархии объединений под Почтовым ящиком.

Создание диаграмм взаимодействия граничных объектов

Для отображения граничных объектов, участвующих в иллюстрированном сценарии прецедента, и их взаимодействия с пользователем мы используем диаграммы последовательности или сотрудничества. Это полезно для прецедентов со сложными последовательностями или потоками событий.

Пример:

Ниже приводится диаграмма сотрудничества, принадлежащая иллюстрированному сценарию прецедента Управление входящими сообщениями почты:

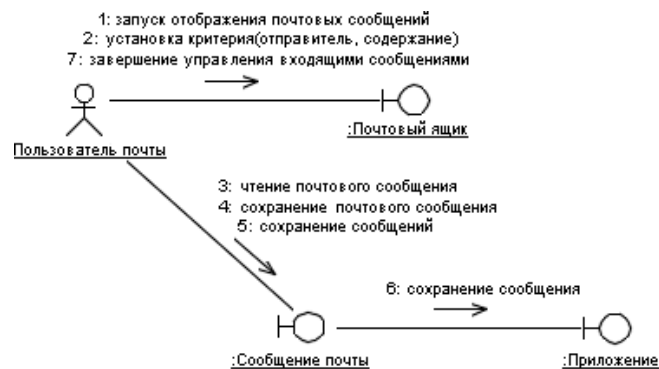
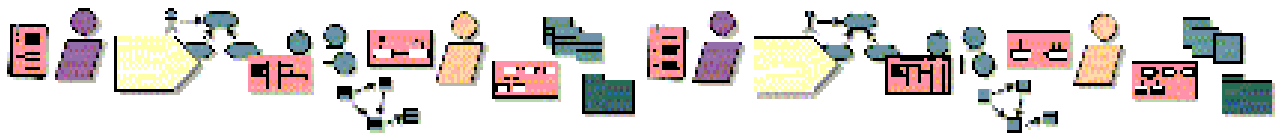


Диаграмма сотрудничества содержит субъект Пользователь почты и граничные объекты Почтовый ящик, Сообщение почты и Приложение, участвующие в иллюстрированном сценарии прецедента, реализующем прецедент Управление входящими сообщениями почты.



Дополнение описания иллюстрированного сценария объектами диаграмм

В случае необходимости, диаграмма иллюстрированного сценария прецедента может разъясняться с использованием потока событий иллюстрированного сценария как дополнительного текстового описания. Это можно сделать расширением описания за счет включения в иллюстрированный сценарий используемых граничных классов.

Пример:

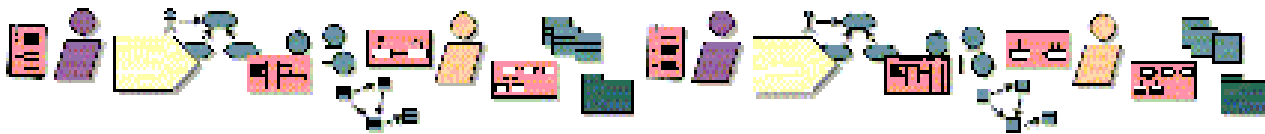
Ниже приводится пример описания потока событий иллюстрированного сценария для прецедента Управление входящими сообщениями почты, расширенное граничными классами (текст в пределах «>>»).

- а) Прецедент начинается, когда пользователь почты требует отобразить сообщения почты и система отображает сообщения. [Пользователь должен иметь способ различать новые, прочитанные и непрочитанные сообщения; кроме того, пользователь должен видеть отправителя, тему и приоритет каждого сообщения.] {Среднее число одновременно показываемых непрочитанных сообщений почты – около 100; в 90 % случаев строка описания содержания сообщения – меньше 40 символов.} «Почтовый ящик»
- б) Пользователь почты может выполнить один или более из следующих шагов:
 - с) Упорядочить почтовые сообщения по отправителям или по темам. (Выполняется больше, чем в 60 % случаев.) «Почтовый ящик»
 - д) Прочитать текст почтового сообщения. {Текст сообщения в среднем содержит 100 символов.} (Выполняется больше, чем в 75 % случаев.) «Сообщение почты»
 - е) Сохранить почтовое сообщение как файл. (Выполняется меньше, чем в 5 % случаев.) «Сообщение почты»
 - ф) Сохранить приложение к сообщению как файл. [Пользователь должен быть способен видеть типы файлов приложений.] {В 95 % случаев сообщения имеют меньше двух приложений.} «Сообщение почты» «Приложение»
- г) Прецедент заканчивается, когда пользователь почты подает команду о завершении работы с почтовыми сообщениями. «Почтовый ящик»

Фиксация требований применимости

Требования применимости могут определять уровень применимости интерфейса пользователя. Эти требования могут быть найдены, например, в артефакте Дополнительная спецификация. Требования применимости должны устанавливать не то, что, как Вы полагаете, должна делать система, а самый низкий уровень применимости, которого должна достичь система, чтобы использоваться.

То, чего должна достичь система, чтобы использоваться, зависит главным образом от того, что является альтернативой к использованию системы. Разумно требовать, чтобы система была



существенно более пригодна для использования, чем альтернативы. В качестве альтернатив можно предложить использовать:

- Существующие ручные процедуры.
- Наследуемую систему(ы).
- Конкурирующее изделие.
- Более раннюю версию(и) системы.

Требования применимости могут исходить также из потребности экономно внедрять новую систему: если заказчик должен заплатить \$3 000 000 за новую систему, он может предложить наложить требования применимости, которые подразумевают, что он сохранит \$1 000 000 ежегодно из-за уменьшенной рабочей нагрузки на свой персонал.

Требования применимости в иллюстрированном сценарии прецедента обычно определяют:

- Максимальное время выполнения – как много времени должно потребоваться обученному пользователю, чтобы выполнить обычный сценарий прецедента.
- Максимальная норма ошибки – сколько ошибок в среднем совершит обученный пользователь при выполнении обычного сценария прецедента.

Единственные допустимые ошибки – это те, которые не являются неисправимыми, и не будут оказывать отрицательного воздействия на организацию, типа потерь в бизнесе или нанесения ущерба аппаратным средствам. Если единственное последствие ошибки – то, что она требует времени для ее идентификации, это затронет измеренное время выполнения.

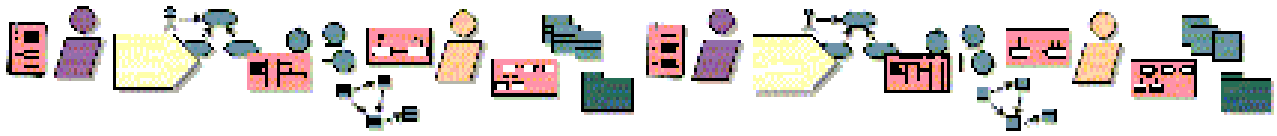
Время обучения должно быть измерено как время, которое нужно затратить прежде, чем пользователь сможет выполнять сценарий быстрее, чем указанное максимальное время выполнения.

Обратите внимание, что требования применимости не должны работать как цель, то есть, верхний предел. Требования применимости должны определять самую низкую приемлемую абсолютную применимость системы. Таким образом, Вы не обязательно должны прекратить улучшать применимость, когда требования применимости выполнены.

Пример:

Ниже приводится пример описания требований применимости для прецедента Управление входящими сообщениями почты, как оно представлено в соответствующем иллюстрированном сценарии прецедента.

- Пользователь почты должен быть способен упорядочить почтовые сообщения одним щелчком мыши.
- Пользователь почты должен быть способен скроллить тексты почтовых сообщений нажатием одной кнопки клавиатуры.



- Пользователь почты не должен получать сигнал о входящем сообщении почты при чтении существующего сообщения.

Поддержка обратной связи с прототипом интерфейса пользователя

Для дальнейшего разъяснения иллюстрированного сценария прецедента последний может ссылаться на элементы (например, окна) прототипа интерфейса пользователя, соответствующего участвующим в нем граничным классам. Это может быть полезно, если части прототипа интерфейса пользователя уже существуют и иллюстрированный сценарий описан.

Синхронизация

Иллюстрированные сценарии прецедентов производятся сразу, как только соответствующие прецеденты будут отсортированы по приоритетам, чтобы иметь возможность рассмотреть их с позиций применимости. Иллюстрированный сценарий прецедента создается до моделирования и реализации интерфейса пользователя, то есть и в потоке работ Требования и в потоке работ Анализ и проектирование.

Создание и уточнение артефакта Иллюстрированный сценарий прецедента происходит в результате действия Моделирование интерфейса пользователя.

Этот артефакт предоставляет исходную информацию для действия Прототипирование интерфейса пользователя.

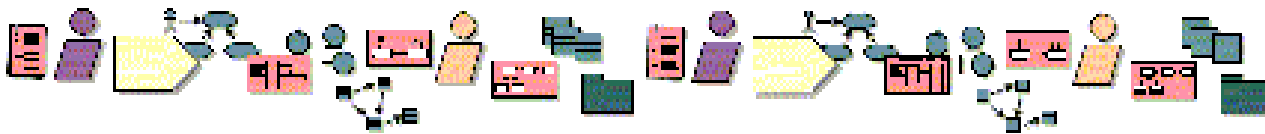
Проектировщик интерфейса пользователя отвечает за целостность иллюстрированного сценария прецедента и гарантирует, что:

- описание иллюстрированного сценария потока событий читабельно и соответствует цели;
- диаграммы, описывающие иллюстрированный сценарий прецедента, понятны и соответствуют цели;
- требования применимости читабельны и соответствуют цели; они правильно фиксируют требования применимости соответствующего прецедента в модели прецедентов;
- трассировка зависимости к соответствующему прецеденту в модели прецедентов правильна;
- связи (связи-ассоциации, связи включения и расширения) соответствующего прецедента в модели прецедентов в пределах иллюстрированного сценария прецедента обработаны правильно.

Очень полезно, чтобы проектировщик интерфейса, ответственный за иллюстрированный сценарий прецедента, был также ответственным за граничные классы и связи, которые используются в иллюстрированном сценарии.

Использование

Иллюстрированные сценарии прецедентов используются, прежде всего, в начальных стадиях разработки, прежде, чем интерфейс пользователя будет смоделирован, как инструмент



«рассуждения», чтобы зафиксировать требования к интерфейсу пользователя.

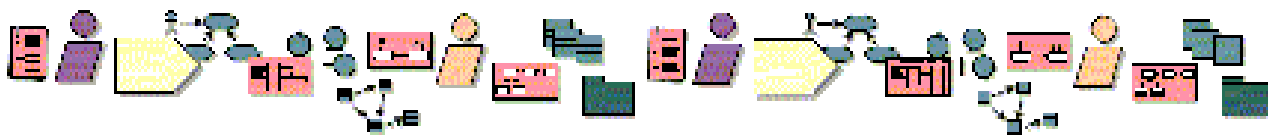
Иллюстрированные сценарии прецедента часто рассматриваются как переходные артефакты, только как проект, и могут не поддерживаться после прототипирования быстрого действия или выполнения интерфейса пользователя. Однако в некоторых случаях может быть полезно поддерживать иллюстрированные сценарии прецедентов в течение ряда итераций, например, если имеются сложные требования, представленные в интерфейсе пользователя, которые требуют времени (более, чем несколько итераций), чтобы стать понятными.

Иллюстрированные сценарии прецедентов должны описываться с учетом того, что читателями этих описаний будут только проектировщики прецедента, так как иллюстрированные сценарии прецедента концептуальны и высокоуровневы, и могут оказаться непонятными другим читателям. Вместо этого, рассматривать и обсуждать, в том числе с другими совладельцами типа конечных пользователей, нужно их конкретные проявления (то есть непосредственно интерфейс пользователя или его прототип). Однако, иллюстрированные сценарии прецедентов могут использоваться проектировщиком интерфейса пользователя для ссылок при испытаниях использования, чтобы сфокусировать внимание на некоторых аспектах тестирования (например, на сложных последовательностях взаимодействий).

Компромисс в вышеуказанной рекомендации: если разработать иллюстрированные сценарии существенно дешевле, чем фактический прототип интерфейса пользователя, может быть разумно воспользоваться их наличием и рассмотреть иллюстрированные сценарии прежде, чем реализовывать прототип. Для этого важно, чтобы описания иллюстрированных сценариев были ясными и самодостаточными так, чтобы они могли быть поняты пользователями; создание таких описаний может потребовать существенных ресурсов разработки.

В случаях, когда в потоке работ требований разработаны иллюстрированные сценарии прецедентов и создан прототип интерфейса пользователя, соответствующие граничные классы – это хорошая исходная информация для действий анализа. Однако иногда возникает потребность создавать иллюстрированные сценарии прецедента в течение анализа. Это делается для увязывания элементов проекта интерфейса пользователя с выполняемыми действиями, потому что:

- Некоторые проекты не создают прототип интерфейса пользователя; вместо этого они проектируют и выполняют непосредственно интерфейс пользователя без прототипа.
- Некоторые проекты создают прототип интерфейса пользователя, но только для небольшого числа прецедентов. Для остальной части прецедентов интерфейс пользователя не прототипируется.



Артефакт: Граничный класс

Краткий обзор

Граничный класс моделирует взаимодействие между одним или несколькими субъектами и системой.

Граничный класс представляет интерфейс между системой и некоторой сущностью вне системы: человеком или другой системой. Его назначение – обеспечить обмен информацией с окружающим миром и защитить систему от изменений в ее среде.

Толкование

Граничные классы используются для моделирования взаимодействия между системой и ее средой, то есть, ее субъектами. Граничные классы фиксируют следующие аспекты взаимодействия:

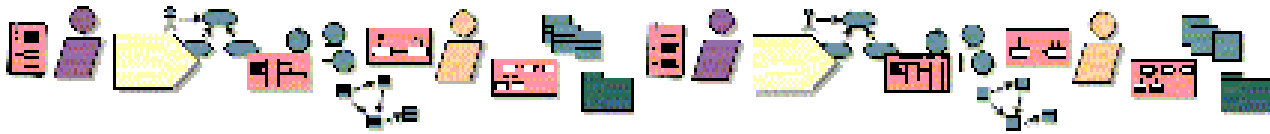
- согласование поведения субъекта с «внутренней организацией» системы;
- прием сообщений от субъекта системе, например, информации или запросов;
- вывод сообщений от системы субъекту, например, накопленную информацию или произведенные результаты.

Таким образом, граничные классы могут использоваться для фиксации требований к интерфейсу пользователя. Создание объектной модели пользовательского интерфейса заслуживает большого внимания, так как многие из создаваемых сегодня интерфейсов пользователя объектно-ориентированы, и наблюдается тенденция к еще большей объектной ориентации из-за ее выгод в части естественного и эффективного использования. Хотя многие из лучших проектных решений интерфейса пользователя были приняты непосредственно в ходе прототипирования и графической разработки, рассуждение относительно структуры и требований применимости к интерфейсу пользователя естественно делать в терминах объектной модели.

Использование граничных классов при моделировании интерфейса пользователя

Интерфейс пользователя, базирующийся на работе с окнами, может быть смоделирован граничными классами; для его описания на высоком уровне используйте следующие рекомендации:

- Используйте один граничный класс, чтобы представить одно главное окно.
- Используйте один граничный класс для представления каждого необходимого логического объекта (типа), управляемого субъектом в пределах главного окна; обратите внимание, что каждый такой объект также часто имеет свои собственные вторичные окна, типа окон свойств.
- Используйте ответственности граничных классов, чтобы описать необходимые операции для соответствующих окон и объектов.
- Используйте атрибуты граничных классов, чтобы описать свойства соответствующих окон и объектов.



- Используйте связи между граничными классами, чтобы представить связи типа иерархии объединений и навигационных путей соответствующих окон и объектов.
- Используйте специальные требования к граничным классам, чтобы зафиксировать требования применимости и другие типы требований, как требования проектирования и выполнения к соответствующим окнам и объектам.

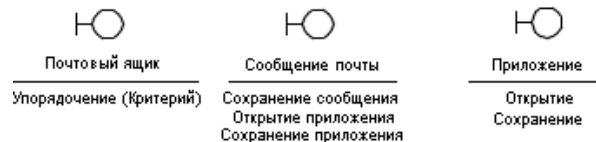
Примеры граничных классов

Следующие примеры можно привести для почтовой прикладной программы:

- Граничные классы:

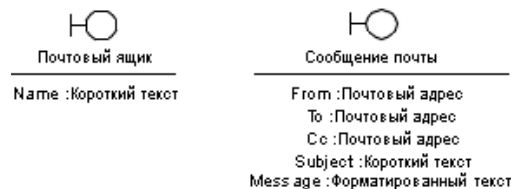


- Ответственности граничных классов:



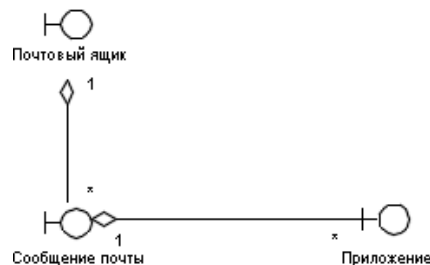
Ответственность Упорядочение (Критерий) состоит в том, чтобы упорядочить почтовые сообщения в окне почты согласно различным критериям, типа отправителя, содержания, времени получения и т.д.

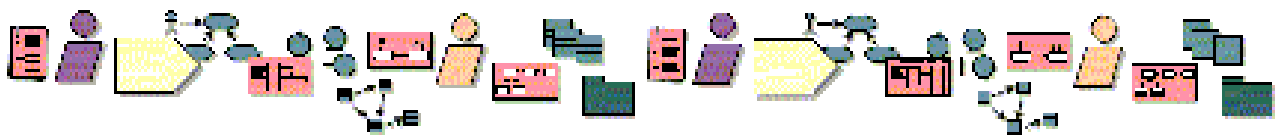
- Атрибуты классов:



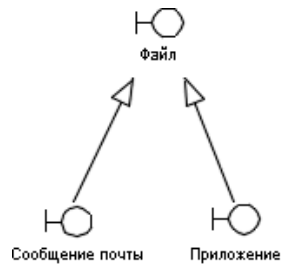
Обратите внимание на то, что типы атрибутов представлены здесь на концептуальном уровне; в фактическом интерфейсе пользователя они могут быть реализованы различными единицами измерения, типа сантиметров, количества символов и т.д..

- Связи объединения:





■ Связи обобщения:



Примеры специальных требований к граничным классам

Специальными требованиями к граничным классам могут быть:

- Требования применимости, такие как квалификация пользователя, время обучения и вероятность ошибки, связанные с граничным классом.
- Не функциональные требования, которые установлены для класса, но которые будут выполнены, когда соответствующий интерфейс пользователя будет разработан и реализован. Примером такого требования является то, что граничный класс должен быть реализован с использованием определенного компонента, типа управления ActiveX.

Требованием применимости для класса Сообщение почты могло бы быть:

- Пользователь почты не должен получать сигнал о входящем сообщении почты при чтении существующего сообщения.

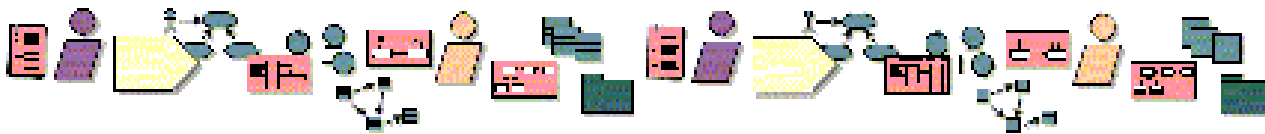
Требованием применимости для класса Почтовое окно могло бы быть:

- Пользователь почты должен иметь возможность упорядочить сообщения почты одиночным щелчком мыши.

Другие требования применимости имеют отношение к значениям атрибутов и размерам объектов. Они выставляются потому, что интерфейс пользователя часто должен обеспечивать читаемость определенного диапазона значений атрибута или пригодное для использования управление объектом, размеры которого находятся в определенном диапазоне. Тогда этот диапазон должен представляться как нормальный случай и обеспечиваться на 90 %. Интерфейс пользователя, в свою очередь, должен покрывать все возможные значения, включая превышающие диапазон, но может быть оптимизирован для значений в нормальном диапазоне.

Требованием применимости относительно значений атрибутов класса Сообщение почты могло бы быть:

- Тема сообщения должна быть способна содержать 40 символов без дефисов.



Связь субъектов и граничных классов

Так как граничные классы моделируют взаимодействия между субъектами и системой, мы поясняем это, сопоставляя каждый субъект с граничными классами, обрабатывающими взаимодействие с субъектом.

Для прикладной программы почты мы идентифицировали субъект Пользователь почты, который взаимодействует с Почтовыми ящиками.



В этом случае мы также обеспечиваем понимание из контекста того, что Пользователь почты взаимодействует со всеми объектами, содержащимися в иерархии объединений Почтового ящика.

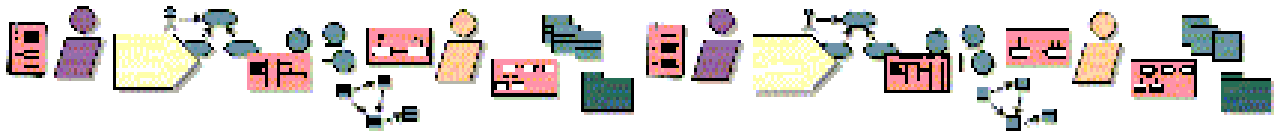
Критерии хорошего качества граничных классов и их связей

Проблема многих интерфейсов пользователя состоит в том, что они имеют слишком много окон и путей перемещения между окнами, которые становятся слишком длинными. Помимо самих дополнительных бесполезных действий, слишком длинные пути перемещения между окнами увеличивают вероятность того, что пользователь «потеряется» в системе. Идеально, если все окна можно открыть из одного первичного окна, часто называемого главным (main) окном. Тогда Вы имеете максимальную длину пути, равную двум шагам. Избегайте путей, имеющих длину более трех шагов.

Таким образом, при формировании интерфейса пользователя важно иметь для субъекта только одно главное окно, если возможно, содержащее все объекты, к которым должен обращаться субъект. Если это невозможно, окно должно обеспечить переход ко всем объектам, к которым должен обращаться субъект. Важно, что субъект проводит значительную часть своего «полезного времени», взаимодействуя с этим главным окном.

В результате эти соображения имеют следующее влияние на объектные модели интерфейса пользователя:

- Каждый субъект должен, если возможно, быть связан с только одним граничным классом (объединением), представляющим главное окно.
- Все граничные классы, используемые одним и тем же субъектом, должны быть связаны некоторым способом, часто через иерархию объединений, начинающуюся с граничного класса, представляющего главное окно.
- Иерархии объединений должны быть широкими и неглубокими, насколько это возможно.
- Граничные классы должны, как правило, быть частями одной (и только одной) иерархии объединений, так, чтобы субъект знал, где их найти.



- Нужно стремиться уменьшить насколько возможно количество иерархий объединения, потому что они усложняют субъектам понимание модели интерфейса.
- Если Вы имеете ассоциацию с множественностью один, и связанные классы – это части различных иерархий объединения, Вы должны рассмотреть замену ассоциации на объединение, если эти две иерархии могут быть объединены. Заметьте, однако, что это может привести к выбору между одной глубокой иерархией объединения и двумя более мелкими.

Критерии хорошего качества, представленные выше, находятся на весьма детальном уровне. Для более высокого уровня можно перечислить следующие критерии, которые не показывают так много деталей, но, по крайней мере, говорят о роли граничных классов:

- Граничные классы должны способствовать хорошей применимости системы.
- Граничные классы должны сохранять высоко концептуальный уровень, насколько это возможно.
- Граничные классы должны обеспечить логичную модульность взаимодействий между системой и субъектом.
- Граничные классы должны быть единственными объектами, которые могут измениться, если субъекты изменяют способ, которым они воздействуют на систему.
- Граничные классы должны быть единственными объектами, которые могут измениться, если система изменяет способ, которым она обеспечивает вывод для субъектов.
- Граничные классы должны «знать» требования от других типов объекта (типа объектов контроллеров и сущностей) так, чтобы они могли быть реализованы и были пригодными для использования и эффективными «внутри системы».

Синхронизация

Граничные классы, относящиеся к применимости системы, идентифицируются и описываются в течение стадий Начало и/или Уточнение прежде, чем будет смоделирован, спроектирован и реализован интерфейс пользователя.

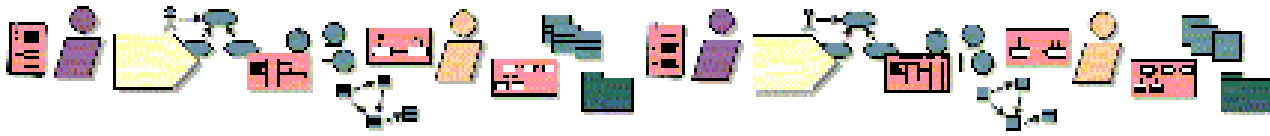
Создание и сопровождение артефактов Граничный класс происходит в действии Моделирование интерфейса пользователя.

Артефакт Граничный класс предоставляет исходную информацию для следующих действий:

- Прототипирование интерфейса пользователя
- Анализ прецедента

Проектировщик интерфейса пользователя или объектный аналитик ответственны за целостность граничного класса, гарантируя что:

- Класс отвечает требованиям, предъявленным в артефактах Реализация прецедента и



Иллюстрированный сценарий прецедента, в которых этот класс участвует.

- Класс настолько независим от других классов, насколько возможно.
- Свойства класса, включая его ответственности, однонаправленные связи и атрибуты, обоснованы и совместимы с друг другом.
- Роль класса в двусторонних связях, в которые он включен, ясна и интуитивно понятна.
- Видимость его элементов, прежде всего, атрибутов, правильна. Видимость может быть «public», «private» и так далее.
- Контекст его элементов, прежде всего операций и атрибутов, является правильным. Контекст – это «true» для контекста тип/класс и «false» для контекста объект/образец.
- Специальные требования читабельны и отвечают своим целям.
- Диаграммы, описывающие класс, читаемы и совместимы с другими свойствами.

Артефакт: Прототип интерфейса пользователя

Краткий обзор

Прототип интерфейса пользователя используют:

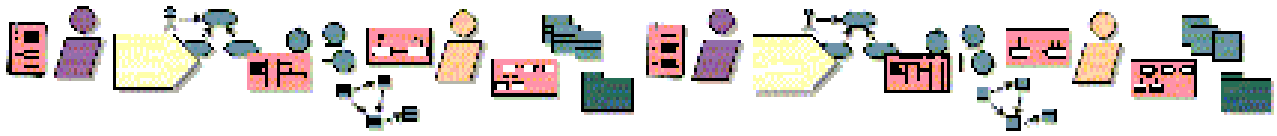
- **описатели прецедентов**, чтобы понять интерфейс пользователя для прецедента;
- **системные аналитики**, чтобы понять, как влияют друг на друга интерфейс пользователя и анализ системы;
- **проектировщики**, чтобы понять, как влияет интерфейс пользователя на «внутреннее устройство» системы;
- **те, кто тестируют классы**, для планирования действий испытания.

Прототип интерфейса пользователя может быть реализован как:

- Эскизы или рисунки на бумаге;
- Точечные изображения, выполненные с использованием инструментов рисования;
- Интерактивный выполнимый прототип (например, реализованный с использованием Microsoft® Visual Basic®).

В большинстве проектов Вы должны использовать все три способа в порядке, перечисленном выше. Так как обычно время их выполнения весьма различно (создавать и изменять рисунки намного быстрее, чем выполнимые программы), этот порядок позволяет вначале просто систематизировать сведения. Однако рисунки не отражают должным образом ограниченную площадь экрана; в рисунке проще поместить больше, чем вмещает площадь экрана.

Лучше всего окончательно определить интерфейс пользователя путем совместного



использования точечных рисунков и выполнимых программ. Это нужно сделать, так как Вы должны будете демонстрировать прототип не только проектировщикам прецедента, но и другим людям. Точечный рисунок может определить точный вид первичных окон, учитывая, что выполнимые программы могут аппроксимировать вид первичных окон и поддерживать их действия, также как вид и поведение вторичных окон. Естественно, если Вы можете с разумным усилием выполнить точный вид первичных окон в выполнимой программе, это лучше, чем объединение выполнимой программы с точечным рисунком. Если Вы не имеете достаточно ресурсов для создания выполнимой программы, Вы можете использовать точечные рисунки как конечное выполнение прототипа. В этом случае может быть полезно дополнить их иллюстрированными сценариями прецедента, описывающими их динамику; иначе велика вероятность того, что конструкторы интерфейса пользователя получат неправильную динамику.

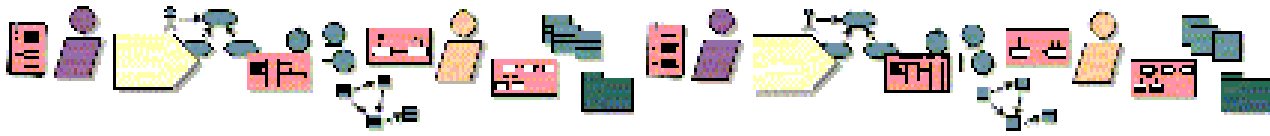
Заметьте, что не нужно стремиться достигнуть хорошей структуры и модуляризации исходного кода для выполнимого прототипа; вместо этого следует сосредоточиться на создании временного прототипа, который визуализирует существенные аспекты интерфейса пользователя и этим обеспечивает некоторые из его существенных действий. Кроме того, прототип, вероятно, несколько раз изменится при разработке и демонстрации его другим, и эти изменения часто выполняются как дешевые заплатки. В результате исходный код прототипа часто имеет очень ограниченное значение, и не «эволюционирует» в реальный код интерфейса пользователя.

Вообще, выполнимый прототип осуществить намного дешевле, чем выполнить реальный интерфейс пользователя. Ниже приведены некоторые различия между прототипом и реальным выполнением интерфейса пользователя:

- Прототипу нет необходимости поддерживать все прецеденты и сценарии. Вместо этого, прототип может поддерживать только небольшое число приоритетных прецедентов и/или сценариев.
- Первичные окна часто достаточно сложны в осуществлении; если Вы делаете продвинутый интерфейс пользователя, который действительно использует преимущества потенциальной визуализации, то может быть трудно найти готовые компоненты. Прежде, чем создавать новые компоненты, обычно Вы можете использовать примитивы, типа нажимной кнопки, кнопок переключения или выбора, как аппроксимацию того, как интерфейс пользователя будет отыскивать некоторый набор данных. Если возможно, сделайте несколько прототипов, показывая различные наборы данных, которые охватывают объекты средней величины.
- Имитируйте или игнорируйте все нетривиально осуществляемые оконные операции.
- Имитируйте или игнорируйте внутреннюю организацию системы, типа деловой логики, внешней памяти, множественных процессов и взаимодействия с другими системами.

Синхронизация

Прототип интерфейса пользователя формируется достаточно рано, в стадии Начало или в начале стадии Уточнение, и прежде, чем вся система (включая ее «реальный» пользовательский



интерфейс) будет проанализирована, спроектирована и реализована.

Обратите внимание, что главная цель создания прототипа интерфейса пользователя состоит в том, чтобы сформулировать и проверить функциональные возможности и применимость системы перед реальным проектированием и началом разработки. Таким способом Вы сможете гарантировать, что Вы строите правильную систему прежде, чем Вы затратите очень много времени и ресурсов на разработку.

Чтобы достичь этой цели, разработка прототипа должна быть существенно более дешевой, чем разработка реальной системы, при наличии достаточной возможности обеспечить значимые эксплуатационные испытания.

Создание и уточнение прототипа интерфейса пользователя происходит в результате действия Прототипирование интерфейса пользователя.

Артефакт Прототип интерфейса пользователя предоставляет исходную информацию для следующих действий:

- Детализация требований к программному обеспечению
- Разработка материалов поддержки
- Разработка учебных материалов

Проектировщик интерфейса пользователя отвечает за целостность прототипа интерфейса пользователя, гарантируя, что прототип способствует созданию пригодного для использования интерфейса пользователя, соответствует требованиям иллюстрированных сценариев прецедента и граничным объектам.

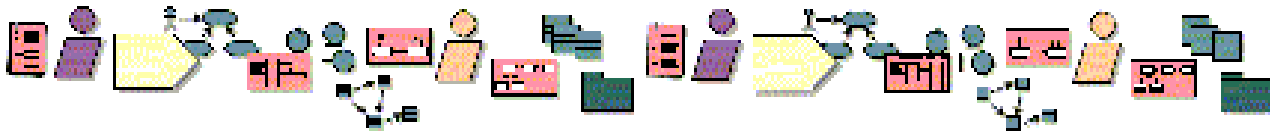
Проектирование прототипа интерфейса пользователя

Ниже перечислены различные аспекты проектирования прототипа интерфейса пользователя:

- Идентификация первичных окон
- Проектирование изображения первичных окон
- Проектирование операций первичных окон
- Проектирование окон свойств
- Проектирование операций, содержащих множественные объекты
- Проектирование различных особенностей

Не обязательно рассматривать все эти аспекты в прототипе. Часто имеет смысл оставить некоторые из них разработчикам, и не иметь с ними дело в прототипе. В этом случае мы рекомендуем Вам не трогать нижние аспекты, а сосредоточиться на верхней части списка.

Интерактивная версия Rational Unified Process содержит подробные рекомендации и много



Как демонстрировать прототип

Часто лучший способ демонстрации прототипа – просмотреть его сидя перед экраном вместе с человеком, которому Вы его демонстрируете. Пройдите через общий сценарий, например, нормальный поток прецедента с нормальными значениями, как описано в иллюстрированном сценарии прецедента. Поощряйте человека задавать вопросы и давать комментарии. Делайте себе заметки.

Другой, довольно дорогостоящий, путь демонстрации состоит в том, чтобы выполнить эксплуатационные испытания. В эксплуатационных испытаниях реальные пользователи выполняют реальные задачи с прототипом. Проблема состоит в том, чтобы получить надежные результаты:

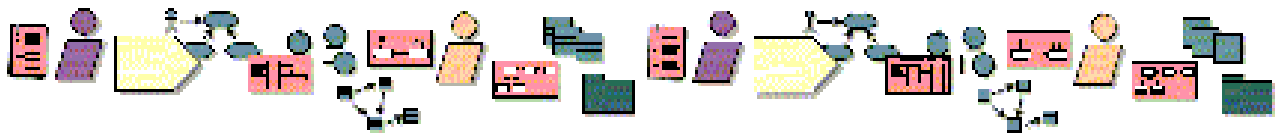
- Вы должны иметь очень «готовый» прототип, функционирующий почти как законченный интерфейс пользователя. Когда Вы достигнете этого в своей разработке, обычно будет слишком дорого делать существенные изменения.
- Вы должны будете обучить пользователя. Большинство пользователей будет «улучшать промежуточные звенья»; если Вы хотите проверить, насколько хорош прототип для этих пользователей, Вы должны будете обучать их, в худшем случае, в течение нескольких недель.

По этим причинам, и если Вы не имеете очень щедрого бюджета для создания интерфейса пользователя, эксплуатационные испытания по максимуму должны быть выполнены в конце итерации (или ее части), в которой создан интерфейс пользователя.

На этом мы заканчиваем знакомство с потоком работ Требования. Пожалуй, единственная тема, которую мы не обсудили достаточно подробно – это управление изменением требований (см. стр. 11-18). Я думаю, что мы исправим это упущение при рассмотрении потока работ Управление конфигурацией и изменением.

Выводы

- Требование – это условие или возможность, которой должна соответствовать система.
- В типичном проекте Вы должны рассмотреть несколько типов требований, включая функциональные и не содержащие функциональности.
- Rational Unified Process определяет проект интерфейса пользователя как описание и визуальное отображение, которое позволяет отработать различные требования применимости.
- В основном потоке работ Требования различные работники анализируют проблему, выявляют потребности совладельцев, определяют систему, управляют контекстом проекта, детализируют систему и управляют изменяющимися требованиями.
- Инструментальные средства Rational поддерживают сбор данных и управление требованиями.



Поток работ «Среда»

Введение

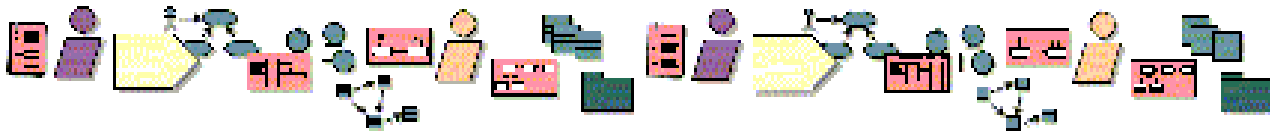
Цели потока работ

Цель действий основного потока работ поддержки Среда состоит в том, чтобы **обеспечить разработчиков программных проектов соответствующей средой** - процессами и инструментальными средствами, которые будут поддерживать группы разработки.

Среда разработки проекта программного обеспечения – это термин для обозначения всего того, что поддерживает разработку и развертывание системы, типа инструментальных средств, рекомендаций, процессов, шаблонов и инфраструктуры. В Rational Unified Process все они представлены как артефакты, которые перечислены ниже:

- Артефакт: Инструментальные средства
- Артефакт: Инструментальная поддержка
- Артефакт: Проектно-определенные шаблоны
- Артефакт: Адаптированный процесс
- Артефакты – руководящие указания, типа:
 - Артефакт: Указания по деловому моделированию
 - Артефакт: План управления конфигурацией
 - Артефакт: Указания по проектированию
 - Артефакт: Руководство по стилям
 - Артефакт: Указания по программированию
 - Артефакт: Указания по испытаниям
 - Артефакт: Указания по моделированию прецедентов
 - Артефакт: Указания по интерфейсу пользователя
- Артефакт: Инфраструктура разработки

Подробно артефакты среды будут обсуждаться в одном из следующих выпусков. Здесь же определим понятие одного из артефактов, который будет упоминаться достаточно часто:



Артефакт **Адаптированный процесс** описывает процесс разработки, который Вы выбрали, и которому Вы будете следовать в вашем проекте.

Если Вы работаете в организации, которая ведет несколько проектов программного обеспечения, то Вы, чаще всего, увидите, что в разработке этих проектов есть много общего. Проекты используют одни и те же инструментальные средства сходными способами. Процессы в различных проектах схожи, и некоторые рекомендации, вероятно, идентичны. Поэтому организация может извлечь пользу от разработки и поддержки общей среды разработки. Эта **среда разработки уровня организации** включает общие части процесса, общие инструменты и общие инфраструктуры.

Действия внедрения среды

Ниже перечислены некоторые ключевые действия, выполняемые при внедрении новой среды разработки, состоящей из Rational Unified Process и поддерживающих его инструментальных средств. Это:

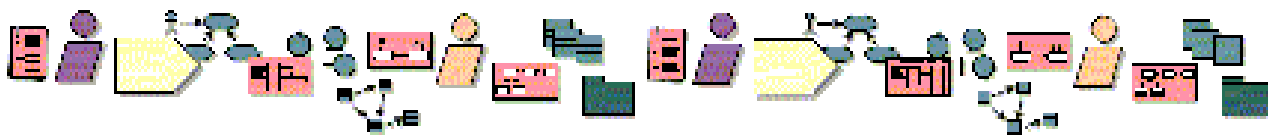
- Оценка проекта и организации
- Внедрение с приращением
- Управление и планирование
- Использование наставников
- Распределение принадлежности частей процесса
- Учет необходимости возврата инвестиций
- Информирование и вовлечение людей
- Обучение персонала

Оценка проекта и организации

Оцените текущее состояние проекта и организации, чтобы лучше понять то, какие части среды Вы должны улучшить. При этом Вы лучше поймете и то, как внедрять новую среду.

Следующие страницы Rational Unified Process объясняют, как оценить состояние проекта и поддерживающей его организации:

- Роль: Инженер-технолог
- Действие: Оценка текущей организации
- Артефакт: Оценка организации разработки
- Рабочие рекомендации: Семинар по оценке



- Роль: Специалист по инструментарию

Внедрение с приращением

Внедряйте среду, процесс и инструментальные средства с приращением так, чтобы люди, работающие в проектах, не были задавлены множеством внезапных нововведений. Внедряя среду с приращением, можно фокусироваться на подмножестве среды, что увеличивает вероятность успеха.

Вводите среду постепенно, по одному элементу одновременно. Результат оценки организации разработки поможет решить, какую часть процесса и какое инструментальное средство Вам следует вводить сначала. Обычно следует сосредоточиться на тех областях, где организация разработки имеет самые большие проблемы.

Внедрение среды с приращением может подразумевать, например, что в первой или начальной итерации, Вы сосредотачиваетесь на введении потока работ Требования вместе с инструментальными средствами управления требованиями. Во второй итерации фокус переходит на введение потока работ Анализ и проектирование вместе с его инструментальными средствами моделирования. В последующих итерациях вводятся все остальные части среды.

Управление и планирование

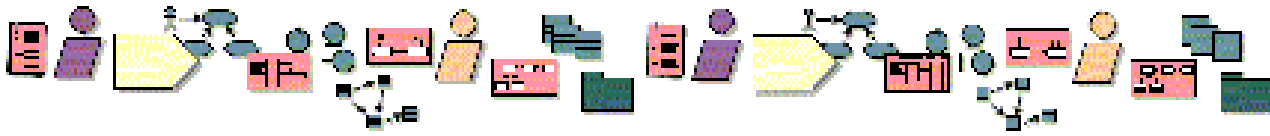
Внедрение нового процесса и новых инструментальных средств в проекте – это сложная задача. При изменении способа работы может подвергаться опасности успех проекта. К сожалению, опыт показывает, что по сравнению с другими действиями разработки, действия над средой иногда пропускаются руководителем проекта.

Действия над средой должны управляться и планироваться подобно всем другим действиям в проекте разработки программного обеспечения. Поэтому важно, чтобы руководитель проекта имел хорошее понимание нового процесса и инструментальных средств. Иногда для руководителя проекта трудно выделить необходимое время, чтобы подробнее узнать и о новом процессе и о потенциальных возможностях новых инструментальных средств. В этом случае руководитель проекта нуждается в поддержке от кого-то, кто знает о том, как внедрить среду, и кто предварительно уже участвовал в этом процессе. Даже если руководитель проекта имеет нужный набор навыков и опыта, мы рекомендуем, чтобы Вы включали в проект «экспертизу внедрения среды», поскольку это значительно увеличивает вероятность успешного выполнения проекта.

Использование наставников

При введении в проект нового процесса используйте наставников. Опыт показывает, что использование наставников критично, если Вы хотите успешно внедрить новый процесс. Если Вы не имеете наставников, существует явный риск того, что люди в проекте будут возвращаться назад к своим старым привычкам. Наставник действует как проводник изменений.

Проект нуждается в ресурсах и в бюджете для наставничества. События некоторых



наставнических действий должны быть запланированы, например, проведение семинаров. Важно, чтобы наставник процесса понимал свою роль проводника изменений, и проверял, что работа прогрессирует. Важно также, чтобы наставник постепенно становился ненужным. Поэтому наставник должен передавать знания и обязанности членам проекта.

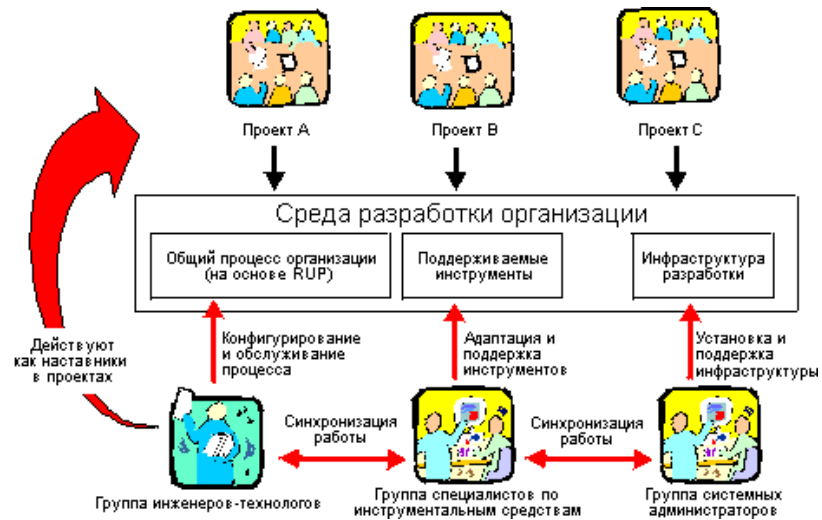
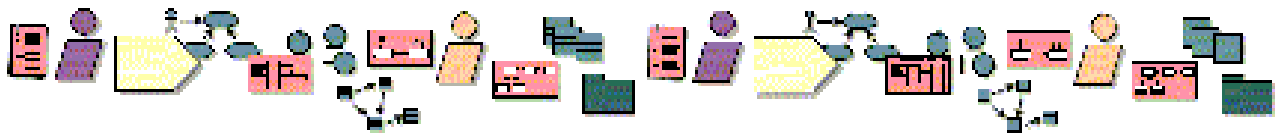
Распределение принадлежности частей процесса

Распределите части процесса среди членов проектной группы, чтобы они как можно быстрее восприняли и изучили новый процесс. Результат разработки лучше, когда «реальными экспертами» являются те участники проекта, которые непосредственно его разрабатывают. Если Вы распределите принадлежность частей процесса, это также увеличит вероятность успеха проекта, который становится слишком тяжелым, если зависит от внешних консультантов.

Как можно скорее назначьте участников проекта ответственными за каждый основной поток работ процесса. Этот человек несет первостепенную ответственность за компоновку соответствующей части процесса и соответствующей части адаптированного процесса. Например, ответственный за основной поток работ Требования является ответственным и за эту часть адаптированного процесса. Каждый человек должен быть ответственным за один или несколько основных потоков работ процесса, а кто-то, кто хорошо знает соответствующую область, может быть наставником для других разработчиков.

Если Вы имеете или собираетесь иметь общую среду уровня организации, полезно иметь специальную группу из инженеров-технологов, которые разрабатывают и поддерживают общий для организации процесс. При наличии в организации общего процесса, отдельные проекты разработки программного обеспечения требуют меньшего количества настроек процесса, потому что многое из того, что является общим для организации, уже сделано. Инженеры-технологи действуют как наставники в отдельных проектах разработки программного обеспечения.

Группа среды может иметь также специалистов по инструментальным средствам, которые устанавливают и обслуживают инструментальные средства поддержки. Специалисты по инструментальным средствам из этой группы могут помогать проектным группам в установке инструментальных средств. В эту группу могут входить и системные администраторы.



Инженеры-технологи, специалисты по инструментальным средствам и системные администраторы готовят среду разработки для организации.

Учет необходимости возврата инвестиций

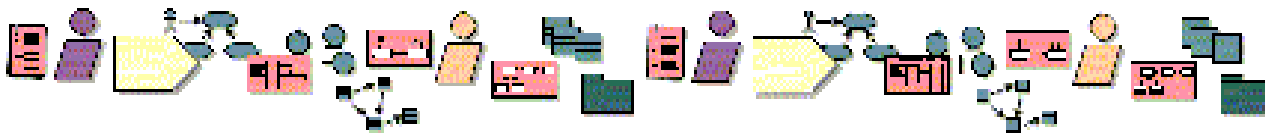
Думайте о возврате инвестиций, когда Вы конфигурируете процесс: сконцентрируйтесь на тех вещах, которые возвратят денег больше, чем объем инвестиций.

Опыт показывает, что некоторые проекты имеют тенденцию тратить слишком много времени и ресурсов на разработку обширных рекомендаций, большого адаптированного процесса и дополнительных материалов, связанных с процессом. С этим связаны три больших проблемы:

- Люди не читают обширные описания
- Выполнить все с самого начала очень трудно. Лучше сделать что-то менее объемное, попробовать его, а затем скорректировать
- Это изменяет роль наставничества. Людей со знанием процесса (наставников) нужно иметь не для того, чтобы написание обширных описаний стало их основной задачей.

Старайтесь многократно использовать существующие рекомендации. Например, хорошее описание существующего прецедента, используемое в качестве Указаний по моделированию прецедентов, может быть стоимостно-эффективной альтернативой разработке законченного руководства по моделированию прецедентов.

На практике Вы не сможете измерять инвестиции и окупаемость, а затем сравнивать варианты. Для инженера-технолога наиболее важное соображение состоит в том, что независимо от того, что он делает, это должно иметь окупаемость для разработчиков.



Информирование и вовлечение людей

Люди должны быть информированы относительно нового процесса и инструментальных средств. Включайте их в работу, потому что самая большая угроза любому изменению в организации – это ориентация на замену людей. Введение нового процесса и инструментальных средств в любой организации означает, что люди должны изменить способы, которыми они работают. Люди имеют естественную сопротивляемость к изменениям. Всегда имеется риск получить отрицательную спираль, когда отрицательный настрой людей приводит к отрицательным результатам, которые, в свою очередь, приводят к еще большему числу отрицательных настроений.

Ниже перечислены некоторые воздействия, которые могут помочь в предотвращении формирования среди сотрудников организации отрицательных ориентаций:

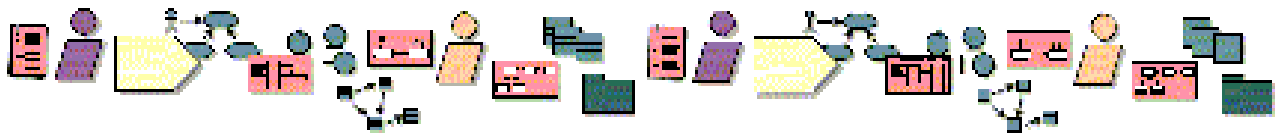
- Устанавливайте реалистичные ожидания. Не преувеличивайте возможности нового процесса или новых инструментальных средств
- Включите ключевых сотрудников в работу над изменениями. Позвольте им участвовать в пилотном проекте и поручите им отвечать за части процесса.
- Объясните, почему должны быть сделаны изменения. Какие проблемы организации будут решены? Какие изменения в технологии требуют нового процесса и новых инструментальных средств? Какую выгоду Вы извлечете из использования этих новых инструментальных средств и процесса?
- Информируйте всех сотрудников обо всех новостях, информируйте все подразделения о ходе внедрения процесса. Эта информация не должна быть очень детальной; важно то, что они получают некоторую информацию.
- Не забудьте о совладельцах, типа заказчиков или спонсоров. Например, если Вы переходите в разработке от водопада-подобного подхода к итерационному, совладельцы должны понять, как управляется итерационный проект и как измеряется его продвижение. В итерационном рабочем проекте, например, они не могут ожидать полностью законченного проекта в раннем промежуточном отчете. Кроме того, они должны понять, что в итерационном проекте совершенно другой способ фиксации требований.

Обучение персонала

Обучите людей новому процессу и новым инструментальным средствам, потому что они должны понять и новый процесс, и то, как используются новые инструментальные средства.

Имеются несколько способов обучения людей, включая использование следующих методов:

- Стандартные учебные курсы
- «Лагерь новобранцев», обеспечивающие от 1 до 5 недель сконцентрированного практического обучения. Немногие организации могут позволить себе устраивать лагерь новобранцев, но



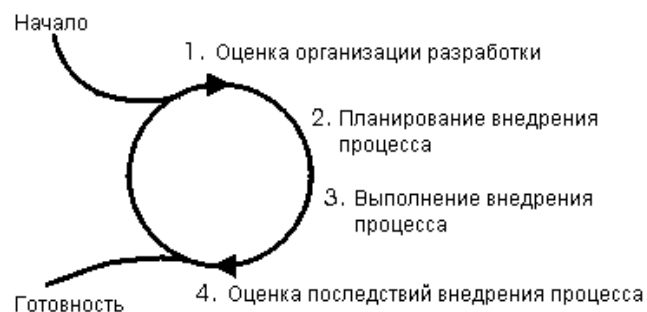
этот метод доказал свою эффективность, если в новом процессе и инструментальных средствах имеется много нового для участников проекта.

- Наставничество в процессе работы, когда наставники делают обзор результатов, ведут семинары и отвечают на вопросы. Наставничество может быть очень эффективным способом передачи знаний.
- Семинары «старт с ходу» – эффективный способ обучения людей почти одновременно с вводом новой части среды. На семинаре такого типа люди работают над реальными проектными материалами и следуют новым частям адаптированного процесса, используя новые шаблоны, рекомендации и инструментальные средства. Как правило, за этот семинар отвечают инженер-технолог и специалист по инструментарию. Не тратьте много времени на разработку учебного материала для семинара «старт с ходу». Главная цель состоит в том, чтобы получить практический опыт, используя новые разделы адаптированного процесса вместе с шаблонами, рекомендациями и инструментальными средствами. Семинар «старт с ходу» – это также хорошая возможность проверить адаптированный процесс, шаблоны, рекомендации и инструментальные средства.

Внедрение процесса в организации

Пошаговая процедура

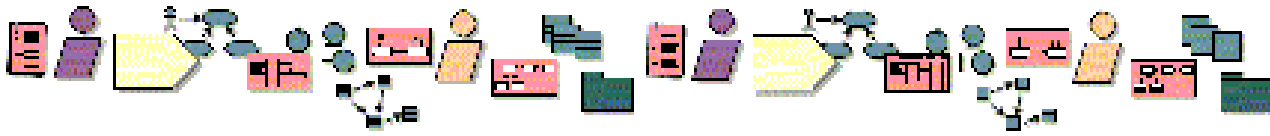
Внедрение нового процесса в организации разработки программного обеспечения может быть описано четырьмя шагами.



Шаги внедрения процесса и инструментальных средств в организации.

Шаг 1: Оценка организации разработки

Вы должны понять текущее состояние организации разработки программного обеспечения в терминах ее людей, ее процесса и ее инструментальных средств поддержки. Вы должны идентифицировать проблемы и потенциальные области для уточнения, а также собрать информацию относительно внешней среды, типа сведений о конкурентах и рыночных тенденциях. Когда этот шаг будет закончен, Вы должны знать:



-
- Текущее состояние организации разработки программного обеспечения.
 - Персонал, включая уровень компетентности людей, их навыки и побуждения.
 - Инструментальные средства, в настоящее время используемые организацией.
 - Текущий процесс разработки программного обеспечения и как он описан.
 - Деловые цели организации.

Как Вы должны использовать эти знания:

- Используйте эти данные для создания плана перехода от текущего состояния к целевому.
- Идентифицируйте те области, которые должны быть улучшены в первую очередь. Вы не сможете ввести весь процесс и все инструментальные средства сразу. Вы сможете делать это в приращениях, начиная с областей, которые имеют самую большую необходимость и лучший потенциал для уточнения.
- Объясните спонсорам, почему Вы должны изменить процесс, инструментальные средства и переквалифицировать людей.
- Создайте мотивы и общее понимание среди тех людей в организации, кто непосредственно или косвенно на это влияет.

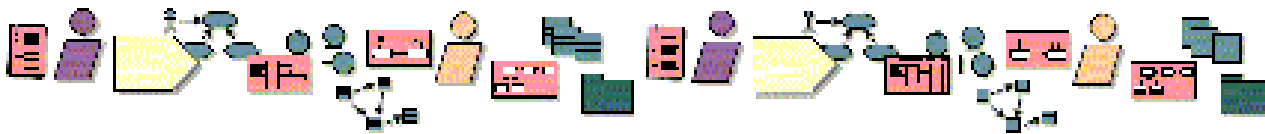
В течение этого шага Вы выполните действие Rational Unified Process «Оценка текущей организации», результатом которого будет артефакт «Оценка организации разработки».

Шаг 2: Планирование внедрения процесса

Разработайте план внедрения процесса и инструментальных средств в организации. Этот план описывает, как эффективно двигаться из текущего состояния организации к ее новому видению. Чтобы разработать этот план, Вам нужно:

- Определить набор целей
- Идентифицировать риски
- Выбрать проекты для первоочередного внедрения среды
- Спланировать обучение
- Спланировать наставничество
- Решить, начинать ли внедрять среду разработки для организации

Некоторые дополнительные рекомендации по планированию внедрения среды будут представлены в конце этого выпуска (см. стр. 16-19).



Выберите в качестве пилотного проекта:

- Законченный проект разработки программного обеспечения, рассматриваемый как имеющий низкие риски с технической и финансовой точек зрения.
- Первую законченную итерацию реального проекта разработки программного обеспечения, с учетом того, что главный фокус находится на изучении и улучшении процесса, а не на разработке программного обеспечения.

Альтернативой может быть выбор нового проекта разработки программного обеспечения, который, как предполагается, имеет низкие риски с технической и финансовой точек зрения. В этом случае, Вы ограничиваете контекст проекта и делаете его вашим пилотным проектом.

Чтобы пилотный проект был эффективен, важно контролировать, что он использует реальную среду разработки и что имеется необходимость реального проектирования.

Пилотный проект должен быть укомплектован некоторыми ключевыми специалистами, которые:

- Умеют слушать людей
- Могут действовать как наставники
- Являются экспертами в некоторых областях

Трудно дать точные рекомендации относительно времени и ресурсов. Однако, следующие данные обычны для многих пилотных проектов:

- Число участников менее десяти
- Продолжительность меньше четырех месяцев
- Один наставник - 50% рабочего времени

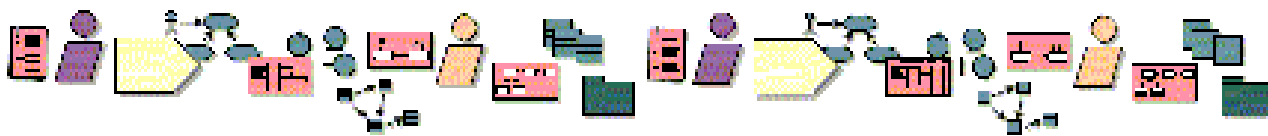
Планирование обучения

Планирование обучения выполняется на основе результатов изучения уровня компетентности персонала, которые были определены на шаге 1: Оценка организации разработки.

Рассмотрите части процесса, которые Вы планируете внедрять и то, какие инструментальные средства будут введены в каждом проекте. Идентифицируйте те области, где персоналу необходимо повысить уровень компетентности, и до какого уровня это должно быть сделано.

Решите, какое обучение необходимо для каждого проекта (см. параграф «Обучение персонала» на стр. 16-6, где перечислены различные формы обучения).

Изменение процесса и инструментальных средств воздействует на всю организацию, поэтому рекомендуется обучать и людей, не участвующих в проектах. Это даст им понимание того, что означают планируемые изменения. Такое обучение может состоять из обзорного курса,



совмещенного с семинарами при введении нового процесса и инструментальных средств.

Планирование наставничества

Опыт свидетельствует, что наличие помощи наставника при внедрении процесса – это ключевой фактор успеха. Поэтому рекомендуется, чтобы каждый проект разработки программного обеспечения имел наставника, помогающего в запуске использования процесса.

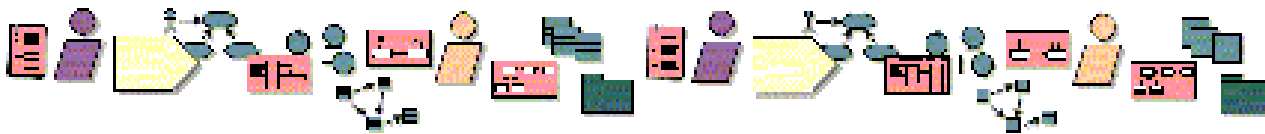
Наставник – это кто-то, кто наставляет и учит проектные группы тому, в чем они нуждаются, и тогда, когда они в этом нуждаются. Типичные способы наставничества:

- **Ведение семинаров.** Некоторые действия лучше выполнять в группе; например, искать субъекты и прецеденты в ходе моделирования прецедентов. В процессе таких действий полезно иметь лидера моделирования, который является экспертом процесса. Опытный лидер моделирования может сэкономить много времени и усилий.
- **Экспертиза процесса.** Эксперт процесса – это человек, который поддерживает проект непосредственно на месте. Задача эксперта процесса состоит в том, чтобы помочь разработчикам использовать все возможности процесса и моделирования. Если эксперт процесса обнаруживает какие-либо проблемы, возникшие из-за недостатка знаний среди разработчиков, он отвечает за восполнение этого пробела знаний в проектной группе.
- **Поддержка руководителя проекта.** Эксперт процесса может помогать руководителю проекта планировать и управлять проектом. Иногда руководитель проекта имеет маленький или не имеет никакого опыта относительно рассматриваемого процесса.
- **Рецензирование.** Недорогой способ передачи знаний состоит в том, чтобы иметь эксперта процесса, делающего обзор результатов каждой стадии. Обзор часто показывает возникшие проблемы, связанные с тем, как члены проектной группы используют процесс. Конечно, рецензент процесса не заменяет никакого из постоянных рецензентов с их прикладной областью и техническими знаниями.

Хороший наставник становится необязательным, потому что цель состоит в том, чтобы проект стал независимым от наставника. Однако, далеко не всегда к этому стремится сам наставник. Кроме того, имеется риск того, что проект станет зависящим от наставника, проявляющего большую заботу обо всех проблемах и трудных задачах, чем решающая их проектная группа. Люди из проектной группы должны быть назначены ответственными за прием задач от наставника, особенно в области монопольного использования процесса.

Помните, что имеется внутренний конфликт между той целью, что наставник должен стать необязательным, и целью руководителя конкретного проекта успешно завершить проект.

Однако для долгосрочного успеха важно не полагаться на внешнего наставника. Поэтому Вы должны планировать передачу роли наставника людям в пределах организации. Поручите кому-то в организации стать экспертом и учиться у наставника.



Одна из потенциальных проблем в этом случае состоит в том, что будущий наставник уже является ресурсом проекта и не может сосредоточиться на задачах наставничества. Поэтому должно быть ясно, что внутренний наставник выполняет только не отнимающие много времени функции:

- Поддержку руководителя проекта
- Рецензирование
- Ведение семинаров

Проекты нуждаются также в помощи при установке инструментальных средств, так что нужно планировать ресурсы для наставничества по инструментальным средствам и для поддержки инструментальных средств.

Решение о начале внедрения среды разработки в организации

Решите, собираетесь ли вы внедрять во всей организации среду разработки, которую с необходимой адаптацией сможет использовать каждый проект разработки программного обеспечения.

В большинстве случаев рекомендуется подождать, пока несколько проектов разработки программного обеспечения начнут использовать процесс и инструментальные средства прежде, чем Вы выберете этот шаг. В таком случае будет проще идентифицировать те части процесса и инструментальных средств, которые являются переналаживаемыми и которые принесут выгоду от внедрения в организации.

Если Вы решили разрабатывать среду для всей организации, Вы должны инициализировать проект создания и внедрения среды разработки организации.

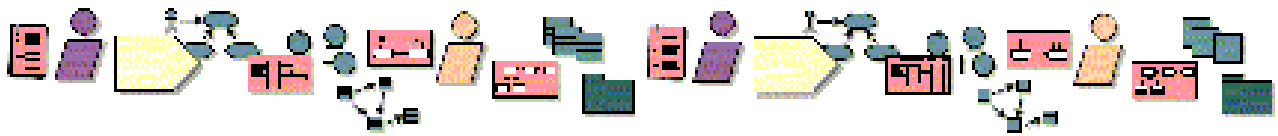
Понятие внедрения процесса как обычного проекта разработки программного обеспечения будет предложено в одном из следующих параграфов этого выпуска (см. стр. 16-17).

Если Вы решаете инициализировать проект создания и внедрения среды для организации, то всем должно быть ясно, что эта проектная группа будет очень тесно работать с группами проектов разработки программного обеспечения. Также должно быть ясно, что группа, которая разрабатывает общую для организации среду, является *сервисной организацией*, и результаты ее деятельности измеряются успехом проектов разработки программного обеспечения, которые она поддерживает.

Шаг 3: Выполнение внедрения среды

Внедрение среды в организации начинается с ее внедрения в проекты разработки программного обеспечения, в которых Вы настраиваете процесс и инструментальные средства. С организационной точки зрения этот шаг означает, что Вы:

- Курируете проекты разработки программного обеспечения.



- Руководите запуском процесса и использованием инструментальных средств в проектах.
- n Курируете развитие среды во всей организации.

Особенностям внедрения среды в проекте разработки программного обеспечения будет посвящен следующий выпуск цикла.

Шаг 4: Оценка внедрения среды

После реализации процесса и инструментальных средств в реальном или пилотном проекте разработки программного обеспечения Вы должны оценить результаты своих усилий. Достигли ли Вы установленных целей? Оцените людей, процесс и инструментальные средства, чтобы понять области, на которых нужно сосредоточить внимание на следующей стадии внедрения процесса.

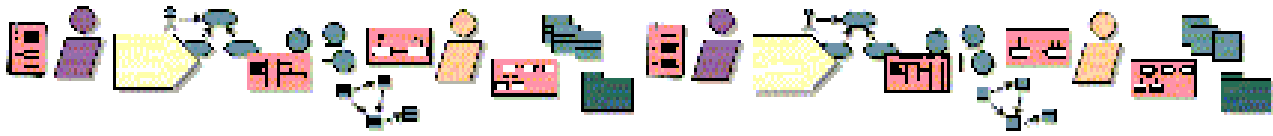
Артефакты

Когда Вы внедряете процесс и инструментальные средства в организации **параллельно** в нескольких индивидуальных проектах разработки программного обеспечения, может быть полезной разработка некоторых дополнительных документов. Эти документы должны разрабатываться как дополнения к артефактам «Оценка организации разработки» и «Адаптированный процесс» для каждого проекта.

- Прежде всего, Вам понадобится *План внедрения*, в котором нужно описать генеральный план того, как внедрять процесс и инструментальные средства **параллельно** в нескольких индивидуальных проектах. Этот план охватывает процесс, инструментальные средства и обучение, и обычно обслуживает несколько проектов разработки программного обеспечения.
- Во-вторых, Вам может понадобиться документ *Рекомендации по развертыванию*, который поможет внедрению процесса и инструментальных средств в отдельных проектах. Рекомендации по развертыванию содержат советы и указания о том, как планировать выполнение процесса и инструментальных средств в индивидуальном проекте разработки программного обеспечения.

Различные подходы к внедрению

Существует несколько подходов к внедрению процесса и инструментальных средств в организации. Некоторые примеры перечислены ниже. Они описывают то, что Вы делаете для всей организации. Что делается при этом в проекте разработки программного обеспечения, Вы прочтете в следующем выпуске.



Типичный подход

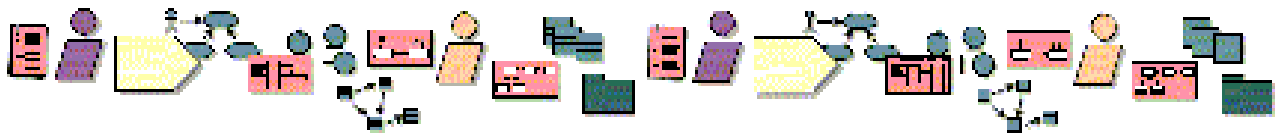
Типичный подход, проиллюстрированный на рисунке ниже, означает, что Вы на первом шаге внедряете процесс и инструментальные средства в пилотном проекте.

После пилотного проекта Вы оцениваете использование процесса и инструментальных средств, а затем готовите процесс и инструментальные средства, которые будут доступны для более широкой аудитории.

Типичный подход – это часто наиболее эффективный способ внедрения процесса и инструментальных средств.



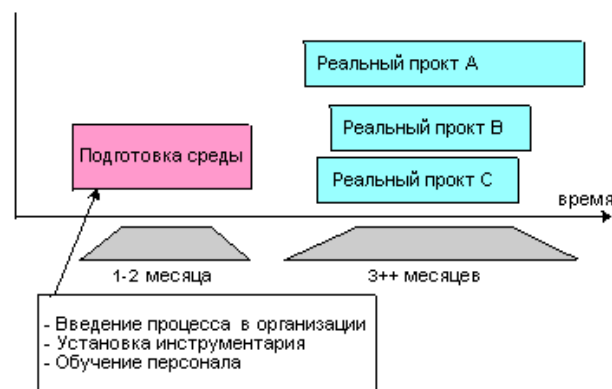
Типичный подход к внедрению процесса и инструментальных средств



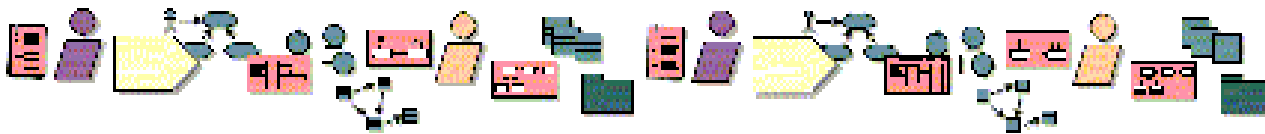
Быстрый подход

Быстрое внедрение, проиллюстрированное следующим рисунком, использует процесс и инструментальные средства непосредственно в реальных проектах без проверки их работы в пилотном проекте. Этот подход вводит больший риск неудачи, и для принятия этого риска должны быть должные основания. Например, если текущий процесс очень похож на Rational Unified Process и если инструментальные средства уже используются в организации, новый процесс и инструментальные средства могут быть внедрены относительно просто и с низким риском.

Другой случай, оправдывающий быстрый подход – это когда организация страдает от таких острых проблем, что любое изменение воспринимается как улучшение. Это предполагает, что потенциал для улучшения больший, чем проблемы, с которыми организация неизбежно столкнется.



Быстрый подход



Осторожный подход

Более осторожный подход состоит в том, чтобы выполнить больше, чем один пилотный проект прежде, чем реальные проекты начнут использовать новый процесс и инструментальные средства. Используйте осторожный подход, когда риски высоки и когда имеется много новых факторов. Вы можете захотеть проверить процесс и инструментальные средства на нескольких проектах прежде, чем Вы запустите его во всей организации.



Осторожный подход

Рассмотрите возможность использования этого подхода, если истинны одно или более из следующих предположений:

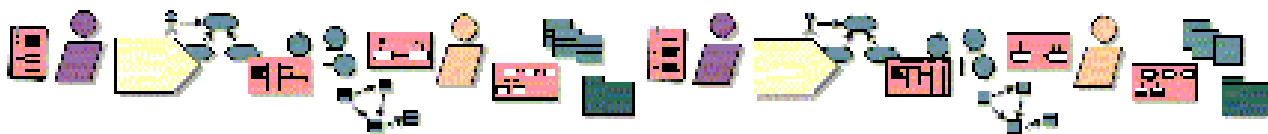
- В процессе и инструментальных средствах есть много изменений, которым нужно научить людей.
- Существует много рисков.
- Способность к изменениям низка.

Рассредоточенный подход

Рассредоточенный подход означает, что Вы делаете Rational Unified Process доступным для всей организации разработки. Каждый проект разработки программного обеспечения может самостоятельно решать, как использовать процесс. Между проектами разработки программного обеспечения нет никакой координации или многократного использования компонентов процесса или инструментальных средств.

Рассредоточенный подход все же может дать некоторые преимущества организации по следующим причинам:

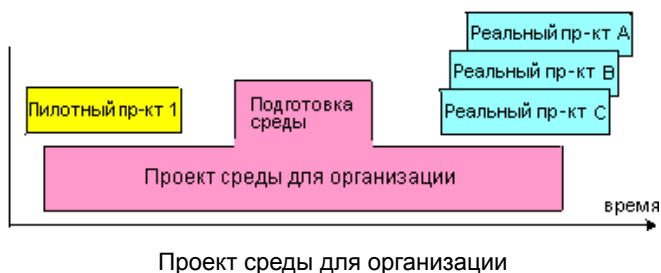
- Проекты получают общий словарь.
- Люди привыкают к Rational Unified Process как к общему процессу.
- Рассредоточенный подход может стать первым шагом к действительному использованию процесса и инструментальных средств.



Разработка среды для организации

Если решено разрабатывать и поддерживать среду разработки для всей организации, эта работа должна быть хорошо спланирована. Необходимо иметь специальную группу, которая разработает и будет поддерживать эту общую для организации среду, состоящую из процесса, инструментальных средств и инфраструктуры.

Планирование проекта среды для организации должно быть синхронизировано с проектами разработки программного обеспечения, которые она поддерживает. Цель проекта среды для организации состоит в том, чтобы разработать среду, которую смогут использовать проекты разработки программного обеспечения.



Рекомендуется, чтобы Вы запустили проект среды для организации, как Вы запустили бы любой проект разработки программного обеспечения. Следуйте за потоком работ Управление проектом Rational Unified Process.

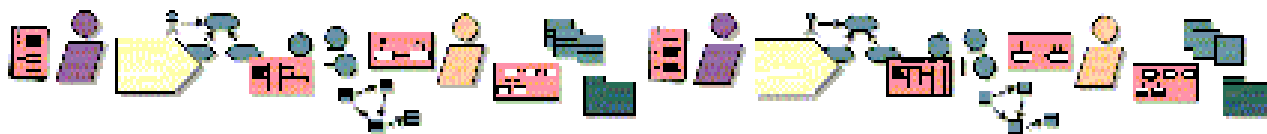
Организация работы

На кого-то в организации должна быть возложена вся полнота ответственности за внедрение процесса и инструментальных средств для всей организации. Эта ответственность предполагает планирование, управление и составление бюджета внедрения процесса и инструментальных средств.

Трактовка внедрения процесса как проекта

Внедрение процесса разработки программного обеспечения в организации – это сложная задача, и она должна решаться управляемым способом. Рекомендуется выполнять ее как проект, который является внешним или подпроектом вашего проекта разработки программного обеспечения. Установите промежуточные вехи, распределите ресурсы и управляйте ими, как Вы делали бы это в любом другом проекте.

Проект внедрения процесса разделен на ряд стадий, где все четыре шага внедрения выполняются в каждой стадии, пока проект не будет завершен, и процесс и инструментальные средства не будут развернуты, и не будут успешно использоваться всей организацией, как показано на следующем рисунке.



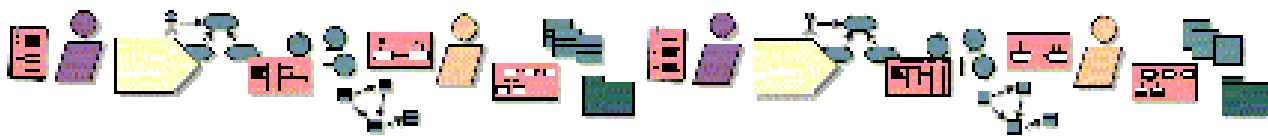
Проект внедрения процесса может быть разделен на стадии.

Следующая таблица дает общую идею относительно того, как может быть запланирован проект из четырех стадий.

	Цель	Важные результаты после завершения стадии
Стадия 1	Донести необходимость выполнения проекта внедрения процесса до спонсоров	Положительное или отрицательное решение спонсоров. Для поддержки решения могут демонстрироваться инструментальные средства и может иллюстрироваться Адаптированный процесс.
Стадия 2	Обработать главные риски	Готов демонстрационный прототип среды разработки программного обеспечения заказчика, включая инструментальные средства, шаблоны, рекомендации и примеры адаптированного процесса.
Стадия 3	Завершить все работы	Завершена заказанная среда разработки программного обеспечения, включая интеграцию, тестирование и демонстрацию ее использования. Все инструментальные средства готовы к использованию. Шаблоны, рекомендации и примеры адаптированного процесса готовы, обучающаяся программа готова и наставники готовы начать поддерживать реальные проекты, начиная со следующей стадии.
Стадия 4	Сделать результаты доступными для всей организации	Процесс и инструментальные средства доступны для использования во всей организации.

Четыре стадии проекта внедрения процесса.

Проект внедрения процесса и инструментальных средств для организации во многом похож на проект разработки программного обеспечения. Дело дошло до того, что были предложения дать этим стадиям названия Начало, Уточнение, Конструирование и Переход, как это предусмотрено для проекта разработки программного обеспечения, использующего Rational Unified Process. Однако, Rational рекомендует Вам не использовать те же названия стадий, чтобы избежать любых



недоразумений.

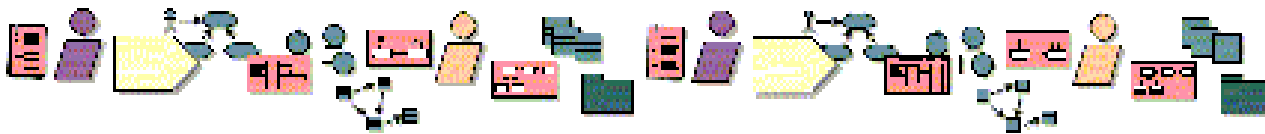
Рекомендации по планированию внедрения среды

Когда Вы определяете содержание и цели промежуточных вех, помните о следующих рекомендациях:

- Держите в памяти конечное видение
- Фокусируйтесь на главном, чтобы возможно раньше разрешить риски.
- Старайтесь как можно раньше разрешить большие проблемы.
- Выберите те области, где Вы можете возможно раньше получить некоторые простые выгоды.

Ниже перечислены некоторые типичные факторы и то, как они могут затрагивать планы.

- **Проблемы в текущем процессе разработки.** Фокусируйтесь на тех областях процесса разработки, где организация имеет сегодня проблемы. Фокусируйтесь на областях, где Вы можете быстро получить некоторые простые результаты и где люди возможно раньше смогут увидеть выгоды. В первых итерациях сосредоточьтесь на области, в которой, как Вы знаете, Вы можете разрешить одну из больших проблем в организации.
- **Потребность изменения в текущей организации.** Если в организации имеется большое количество проблем, например с инструментальными средствами или способами работы людей, уровень разочарований в организации будет высоким. В таком случае Вы можете быть более активны, и использовать новый процесс и инструментальные средства или их части в реальных проектах.
- **Готовность к изменению в текущей организации.** Если изменения являются непривычными или невозможными в настоящее время, цели первых нескольких итераций должны быть скромными. В этом случае первичной целью должно быть формирование доверия и уверенности в процессе, а более серьезные изменения должны быть зарезервированы для более поздних итераций, когда они могут быть проведены более легко.
- **Размер организации.** Если организация, использующая процесс и инструментальные средства большая, убедитесь, что адаптированный процесс и инструментальные средства достаточно устойчивы, чтобы использоваться многими разработчиками. В этом случае нужно быть более осторожными и внедрять процесс и инструментальные средства поддержки в течение нескольких итераций в одном или большем количестве проектов разработки программного обеспечения.
- **Включенные риски.** Если они маленькие, будьте более активны и запускайте использование процесса и инструментальных средств в новых проектах раньше. Если риски большие, будьте более осторожны и используйте пилотные проекты для проверки процесса и



инструментальных средств.

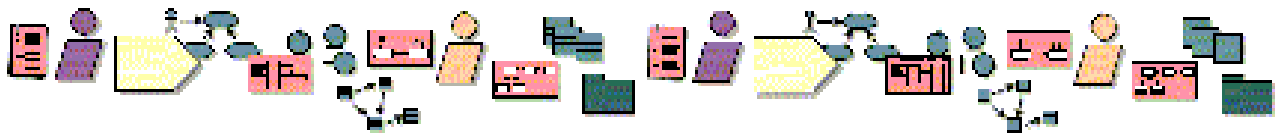
- **Ориентация среди людей в организации.** Назовите сегодняшние проблемы в организации и покажите, как она работает. Если персонал понимает сегодняшние проблемы, будет проще принять и понять потребности в изменении. Постарайтесь сразу привлечь людей из наиболее активной части персонала организации.

Не пытайтесь сделать все сразу. Вместо этого разделите внедрение на ряд приращений и в каждом внедрите часть нового процесса вместе с инструментальными средствами поддержки. Как правило, сосредоточьтесь на одной из областей, где, как Вы полагаете, изменение будет иметь большое влияние. Если организация разработки программного обеспечения имеет слабую сторону в тестировании, Вы можете начать, вводя поток работ Испытание Rational Unified Process вместе с инструментальными средствами, которые автоматизируют тестирование. Однако, если организация имеет слабую сторону в фиксации или в управлении требованиями, начните, вводя поток работ Требования вместе с его инструментальными средствами поддержки.

Внедряйте процесс и инструментальные средства в итерациях проектов разработки программного обеспечения (в пилотных или в реальных проектах). Цель состоит в том, чтобы протестировать процесс и инструментальные средства как реальную среду, насколько это возможно. Рассмотрите следующие факторы, когда Вы выбираете проекты разработки программного обеспечения и итерации:

- Если цель состоит в том, чтобы внедрить процесс и инструментальные средства в одиночном проекте разработки программного обеспечения, решите, внедрить ли процесс и инструментальные средства в этот один проект, а затем оценивать и улучшать процесс по мере продолжения проекта.
- Если цель состоит в том, чтобы внедрить процесс и инструментальные средства в большой организации во многих проектах, рассмотрите возможность внедрения и проверки процесса и инструментальных средств в итерациях нескольких стадий. В этом случае выберите относительно маленький проект, где Вы сможете применять процесс и инструментальные средства в течение всего жизненного цикла программного проекта.
- Если Rational Unified Process представляет существенное отклонение от вашего текущего процесса разработки программного обеспечения, или если Вы должны получить хорошее управление рисками и преимуществами введения нового процесса, или если Вы работаете в новой организации с небольшим процессом или при отсутствии процесса вообще, рассмотрите возможность проверки Вашего или упрощенного адаптированного процесса на мини-проекте или на пилотном проекте перед применением его к вашему главному, критическому проекту разработки.

Использование нового процесса, новых инструментальных средств и, возможно, новая технология в программном проекте делает график проекта более энергозависимым. Убедитесь, что распределили время и ресурсы на внедрение процесса, обучили людей, и так далее в итерациях



проекта разработки программного обеспечения, в котором Вы запускаете использование процесса и инструментальных средств.

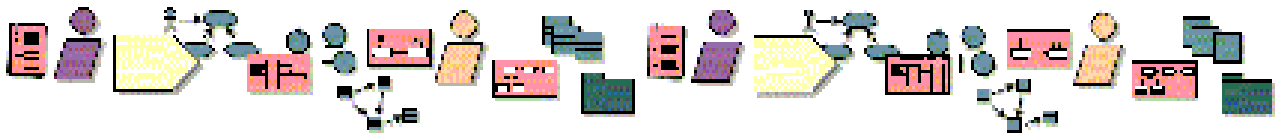
Основные причины для неудачи

Лучше учиться на чужих ошибках!

Ниже представлен краткий список типичных причин неудач внедрения процесса в организации:

- Неудача внедрения процесса и инструментальных средств с приращением.
- Недостаток поддержки руководства.
- Недостаток участия совладельцев — все совладельцы, на которых воздействует новый процесс и инструментальные средства, должны, каждый в своей мере, участвовать во внедрении, включая заказчиков, другие отделения организации и субподрядчиков.
- Недостаток готовности или способности организации к изменению.

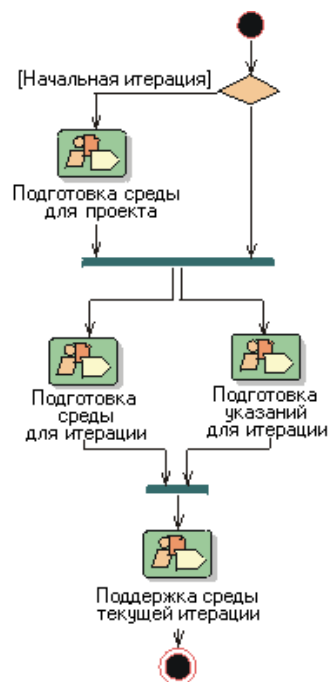
Как показывает опыт, эти причины, чаще всего, не являются техническими проблемами.



Поток работ «Среда» (Продолжение)

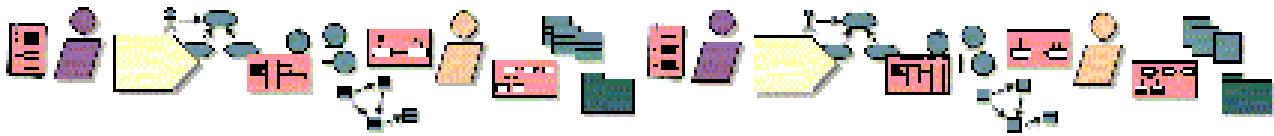
Внедрение процесса в проекте

Внедрение процесса и инструментальных средств в проекте осуществляется при выполнении потока работ дисциплины Среда в тесном взаимодействии с действиями дисциплины Управление проектом, которая имеет дело с планированием проекта, идентификацией рисков и текущим управлением проектом (см. стр. 6-1 – 7-22).



Поток работ дисциплины Среда

Важно понять, что могут быть различные способы внедрения процесса. Подход, который Вы выбираете, зависит от текущего состояния проекта и от окружающей его организации. Естественно, если Вы уже имеете общую среду разработки уровня организации, Вам придется значительно легче. В вашем распоряжении будет некоторый базовый процесс, шаблоны, стандарты. Вам помогут специалисты из группы среды (см. иллюстрацию на стр. 16-5). Но может быть, именно Вы окажетесь первопроходцем (или «подопытным кроликом»?). Дальнейшее изложение ориентировано на этот случай.



Приспосабливание процесса

Вашей первой задачей будет разработка Адаптированного процесса (см. стр. 16-2). Чаще всего это действие сводится к приспособливанию какого-то базового процесса к потребностям Вашего проекта. Этим базовым процессом будет или процесс уровня организации, о котором говорилось в предыдущем выпуске, или Rational Unified Process, если Вы являетесь первопроходцем в вашей организации.

Полные благих намерений «новообращенные» поклонники Rational Unified Process часто стремятся использовать все его рекомендации, возможности и действия. Ведь это «лучший опыт»! А что «говорит» по этому поводу сам Rational Unified Process?

Цель программного проекта состоит в том, чтобы **произвести изделие**. Хороший процесс должен позволить произвести изделие, которое отвечает потребностям его совладельцев, вовремя и в пределах бюджета. При этом **процесс является настолько простым, насколько это возможно**, и в то же время он все еще наследует **лучший опыт**, сконцентрированный в Rational Unified Process.

Когда Вы выбираете из множества артефактов, действий и ролей Rational Unified Process те, которые будут использоваться в проекте, спросите себя: “А нужен ли этот предмет в моем проекте?”, “По какому принципу я могу выбирать, какие из этих предметов нужны в моем проекте?» и ”Нужно ли в моем проекте выполнять выбранные предметы в полном объеме, как они описаны в Rational Unified Process?”. Ответы на эти вопросы являются исходной информацией для приспособливания.

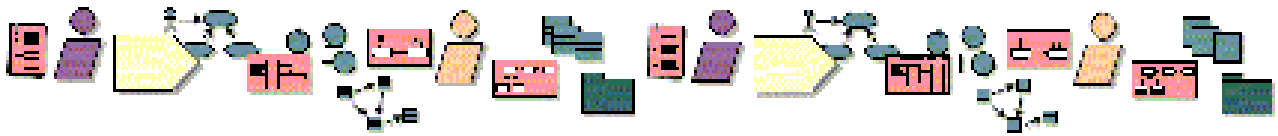
Сначала сформируйте каркас

Общая проблема многих проектов состоит в том, что они часто сосредотачиваются на одной специфической области и «утопают» в подробностях этой специфической области. Лучше сначала схематично указать все основные элементы процесса перед детальным раскрытием его специфических областей.

После того, как этот каркас процесса разработки программного обеспечения будет создан, проект может эффективно сосредоточиться на специфической области, которая идентифицирована как главная составляющая ваших проблем. Этот выбор должен базироваться на идентифицированных и расположенных по приоритетам рисках проекта и на ранее определенной стратегии уменьшения этих идентифицированных рисков.

Не включайте необоснованные действия и артефакты

Руководитель проекта или инженер-технолог могут иметь большой список пожеланий «хорошо бы иметь»: метрику, средство управления, отчет и т.д. Однако действия и артефакты стоят времени и денег. Некоторые из этих затрат, типа ежедневного взаимодействия с комплектом инструментальных средств среды, могут быть видимыми или не видимыми, но они проявляются в более низкой производительности основных задач.



Отличайте критические потребности процесса от тех, которые «хорошо бы иметь». Определяйте, перевешивают ли выгоды их стоимость.

Оградите ваших разработчиков от процесса

Вы, вероятно, хорошо обучили персонал навыкам проектирования, выполнения и тестирования. Нет смысла тратить их время на еженедельное заполнение форм, шлифовку документации или борьбу с громоздкими инструментальными средствами. Если эти действия действительно необходимы, то рассмотрите возможность возложить эти действия на персонал поддержки.

Минимизируйте формальные промежуточные артефакты

Формат промежуточного артефакта – артефакта, который не предназначен для конечного продукта – не столь важен, как действия и размышления, в результате которых он создан. Совершенно не имеет значения, как он выглядит или какие инструментальные средства Вы используете для его формирования, если он достигает цели.

Западная, в которую попадают многие, это стремление сразу формализовывать артефакты. Ранние версии артефактов часто быстро развиваются и в течение некоторого времени остаются изменчивыми, а их значения исследуются. Формализация документа может препятствовать этому процессу. Вы можете затратить много времени впустую, полируя артефакт, который является в значительной степени потребляемым. Нарисованные от руки диаграммы, простые описания на индексных карточках часто являются достаточными в ранних стадиях артефакта, а для некоторых проектов большего может и не требоваться.

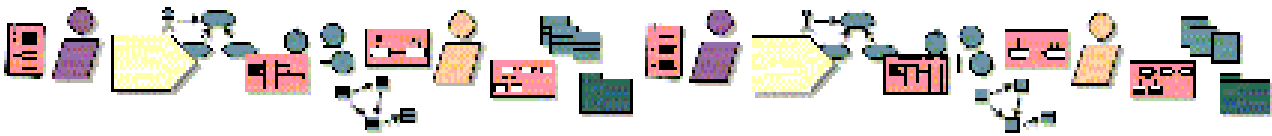
Используйте удобные форматы

Артефакт может быть приспособлен, чтобы поддерживать его в любой форме. Например, документ Видение может быть зафиксирован как веб страница, План проекта может быть зафиксирован как файл Microsoft Project, а Список рисков – как тип требования Rational RequisitePro.

Генерируйте, когда это возможно

В некоторых проектах затрачивается много времени на заполнение шаблонов официальных документов ручным копированием и вставлением информации. Рассмотрите возможность производства требуемых документов из источника с использованием инструментальных средств типа Rational Soda. Или примите более простые средства представления той же самой информации, например, использование Rational Rose Publisher для генерации и публикации модели проекта на веб.

Во многих случаях Вы можете вообще пропускать артефакт, потому что соответствующая информация неявно представлена в среде. Например, вместо генерации раздела Плана управления требованиями, перечисляющего атрибуты типов требований, Вы можете просто представить приспособленный проект Rational RequisitePro с соответствующими типами требований и трассируемостью, и допустить к нему все заинтересованные стороны. Как другой пример,



представьте всем заинтересованным сторонам версию только для чтения файлов Microsoft Project вместо дублирования графика в План разработки программного обеспечения.

Используйте веб

Полезный артефакт – тот, который сообщает ценную информацию. Эта информация должна быть на кончиках пальцев у тех, кто в ней нуждается. Веб-технология – превосходный механизм для этой цели. Если требования, проект и выполнение доступны на веб, нет никакой необходимости в производстве больших наборов быстро устаревающей бумажной документации.

Используйте интегрированные инструментальные средства

Выберите инструментальные средства, которые соответствуют процессу, и приспособьте процесс к имеющимся инструментальным средствам. Желательный результат – это простой в использовании процесс и комплект инструментальных средств. Интегрированные инструментальные средства способны обеспечить большую согласованность, они могут предоставить более информативные измерения и отчеты, чем инструментальные средства, которые не интегрированы.

Регулярно возвращайтесь к процессу

Регулярно повторно обращайтесь к процессу, чтобы совершенствовать его и снижать сложность. Если ваш персонал не убежден, что каждый шаг в процессе нужен для конечного продукта, то процесс, вероятно, излишне сложен.

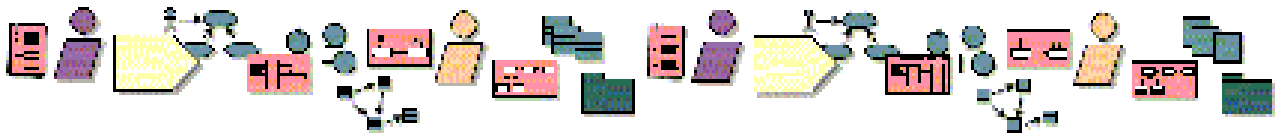
Приспосабливайте при сохранении лучших действий

Rational Unified Process поощряет приспособливание. Однако, приспособливание – это не разрешение искать обходные пути процесса в целом. Сущность Rational Unified Process воплощена в его **лучших действиях**. Следуйте духу этих лучших действий при приспособливании действий и артефактов, чтобы удовлетворить ваши потребности.

Общие рекомендации по планированию

Воспользуйтесь рекомендациями по внедрению среды в организации: не стремитесь внедрить новый процесс одномоментно. Следующие общие рекомендации применимы почти в каждом проекте:

- **Перед началом проекта:** Прежде, чем проект фактически начнется, люди, которые действуют как инженеры-технологи, специалисты по инструментарию и руководители проекта, должны изучить Rational Unified Process. Это критический фактор для успешности проекта. Если члены проектной группы не знают, что делать, они, вероятно, не преуспеют.
- **Стадия Начало:** В течение этой стадии Вы, обычно, фокусируетесь на понимании того, как улучшить способы управления требованиями (поток работ Требования) и управления проектом (поток работ Управление проектом).



-
- **Стадия Уточнение:** К концу стадии Уточнение весь процесс и инструментальные средства используются. Наиболее критичная часть этой стадии – как выполнять управление конфигурацией и изменениями, потому что в стадии Конструирования члены группы разработки будут параллельно выполнять совместные работы.
 - **Стадия Конструирование:** В этой стадии не вводятся никакие новые процессы или инструментальные средства. Работа здесь сосредоточена на производстве изделия, поэтому среда разработки должна быть устойчивой. В стадии Конструирования возникает побуждение подключить к проекту новых людей для увеличения скорости разработки.
 - **Стадия Переход:** Не вводятся никакие новые процессы или инструментальные средства. В стадии Переход фокус сдвигается на специфическое для проекта уточнение процесса, чтобы произвести «анатомическое вскрытие» проекта и собрать опыт, обобщить его и сохранить в форме, которую смогут использовать будущие проекты. Этот обобщенный опыт служит исходной информацией для усовершенствования процесса и инструментальных средств при разработке следующего поколения изделия.

Подходы к внедрению процесса и инструментальных средств

Rational Unified Process описывает два базовых метода внедрения процесса и инструментальных средств в проекте разработки программного обеспечения. Эти подходы:

- «Изменить все». Это означает, что проект принимает весь Rational Unified Process и весь комплект новых инструментальных средств.
- «Улучшить процесс». Это означает, что руководство проекта решает улучшить некоторые области процесса и инструментальных средств, принимая части Rational Unified Process и инструментальных средств поддержки.

Какую часть RUP Вы предполагаете принять и какие новые инструментальные средства Вы решаете внедрить в определенном проекте, зависит от следующих факторов:

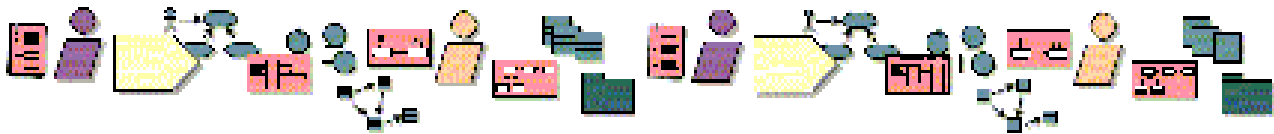
- Какие проблемы идентифицированы и как они распределены по приоритетам для проекта
- Какова способность к изменениям

Это те факторы, которые Вы обычно раскрываете в ходе оценки (впоследствии мы подробно рассмотрим действие RUP «Оценка текущей организации» и артефакт «Оценка организации разработки»).

«Изменить все»

Руководство проекта может решить принять полный RUP и начать использовать новый набор инструментов по одной или нескольким из следующих причин:

- В распоряжении нет никакого процесса и инструментальных средств, и проект нуждается



во всем – и в законченном процессе, и во всех инструментальных средствах.

- Все, или большая часть людей – вновь наняты и у них нет обычно принимаемых общих способов работы.
- Проект переходит на новую технологию для организации, что означает, что существующий процесс и инструментальные средства будут заменены.

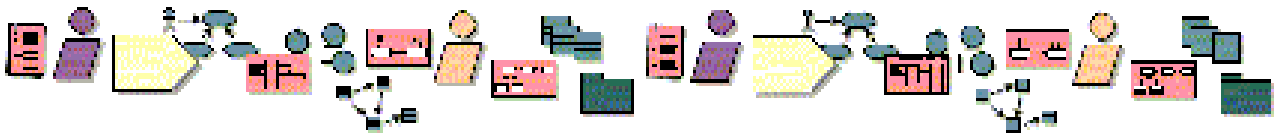
Если Вы решили вводить в ваш проект полный Rational Unified Process и инструментальные средства, то важно внедрять процесс и инструментальные средства с приращением (см. стр. 16-3). При пошаговой процедуре внедрения процесса и инструментальных средств проще управлять рисками. Кроме того, постепенность изменений не подавляет людей на проекте. Следующая диаграмма иллюстрирует возможный порядок разработки различных артефактов потока работ Среда по итерациям в жизненном цикле проекта.



Эволюция артефактов потока Среда в проекте при использовании подхода «изменить все».

Комментарии к плану:

- **Общий:** Поток работ Деловое моделирование пропущен целиком.
- **Начало:** Фокус на введении потоков работ Требования и Управление проектом. Чтобы уменьшить число новых факторов, часть, посвященная пользовательскому интерфейсу в потоке работ Требования, не вводится. Руководитель проекта решает, какие части потока работ Управление проектом он будет использовать.
- **Уточнение, итерация E-1:** Наиболее важными в стадии Уточнение является поток работ Анализ и проектирование и архитектура. Автоматизация тестирования и Управление конфигурацией и изменением в начале проекта не так критичны потому, что число членов



проектной группы относительно низкое. Они могут быть введены в проект позже.

- **Уточнение, итерация Е-2:** Инструменты и процесс тестирования вводятся для автоматизации тестирования. RequisitePro вводятся для управления изменяющимися требованиями.
- **Уточнение, итерация Е-3:** В стадии Конструирование совместная работа будет выполняться многими работниками параллельно. Поэтому критично иметь поток работ Управление конфигурацией и изменением готовым в конце стадии Уточнение. Администратор развертывания решает, как выполнять поток работ Развертывание.
- **Конструирование:** Ничего нового не вводится. С точки зрения потока работ Среда фокус в течение стадии Конструирование состоит в подключении дополнительных людей для увеличения скорости разработки проекта.
- **Переход:** Ничего нового не вводится. Процесс и инструментальные средства совершенствуются по необходимости.

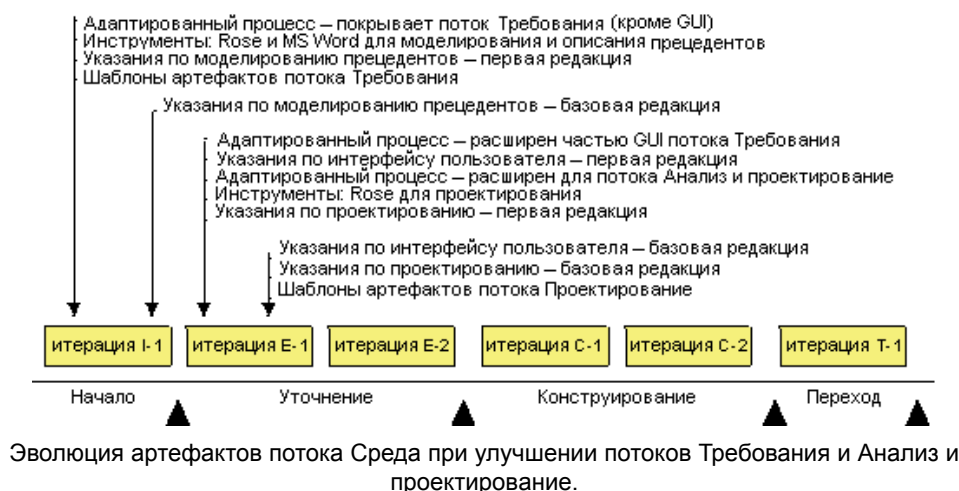
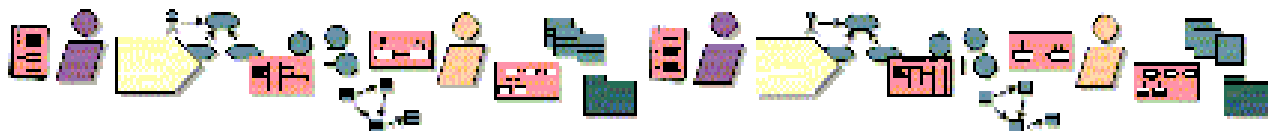
«Улучшить процесс»

Участники проекта в организации, где процесс и инструментальные средства уже есть, имеют возможность разрабатывать системы. Эти люди имеют общий способ работы, который является процессом, и который, может быть, более или менее хорошо задокументирован.

Долгосрочная цель может состоять в том, чтобы принять законченный Rational Unified Process и полную систему новых инструментальных средств. Однако, краткосрочная цель состоит в том, чтобы произвести улучшение в одной или нескольких областях процесса и инструментальной поддержки. Это должны быть области, которые имеют высокий потенциал уточнения.

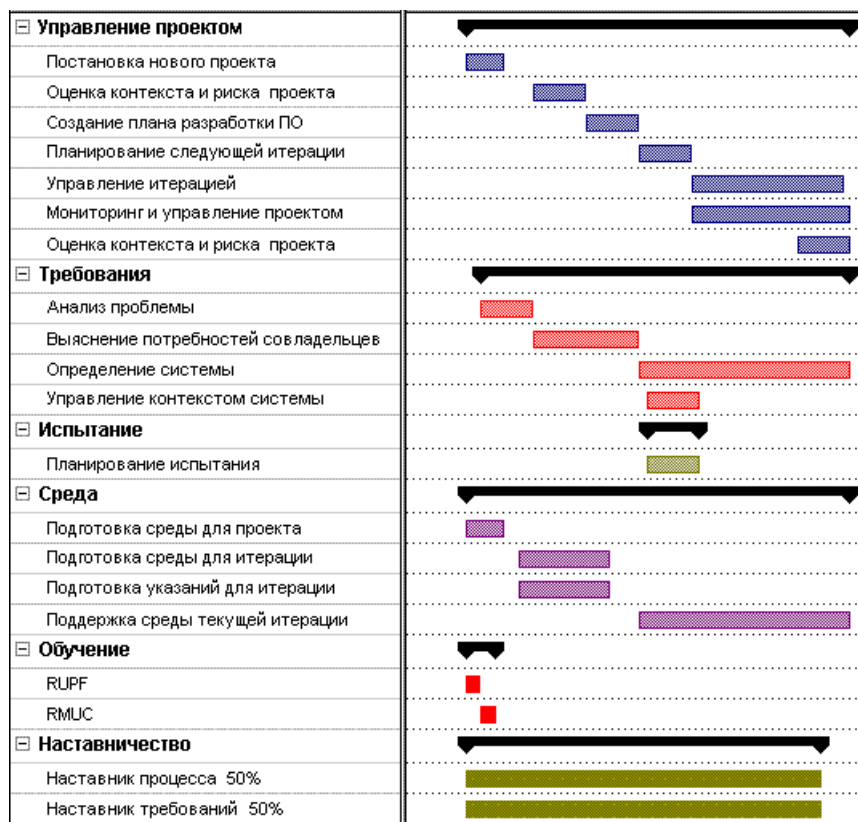
Диаграмма ниже показывает пример проекта, руководство которого решило внедрить поток работ Требования вместе с инструментальными средствами RequisitePro и Rose и улучшить способ, которым управляются требования. Кроме того, решено ввести поток работ Анализ и проектирование.

Важно понять, что диаграмма ниже – это только пример. Части процесса, который Вы решили улучшить, будут отличаться в различных проектах, в зависимости от проблем и потребностей специфического проекта. Вы должны оценить проект и выполняющую его организацию, чтобы определить те части процесса, которые Вы хотите улучшить, или инструментальные средства, которые Вы хотите ввести.

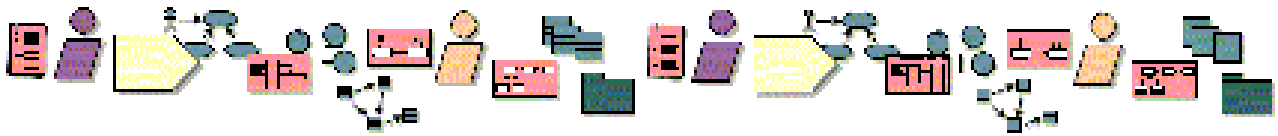


Пример итерации в стадии Начало

Ниже представлен пример итерации в стадии Начало, где вводится поток работ Требования. Каждая задача диаграммы Гантта достаточно подробно описана после диаграммы.



Пример плана итерации в стадии Начало



Эта иллюстрация подготовлена с использованием шаблона MSProject, представленного в разделе RUP «Типовой план итерации: стадия Начало», с нижеперечисленными изменениями и дополнениями.

Управление проектом

■ Часть потока работ: Постановка нового проекта

Проведите проект от зарождения идеи до точки, в которой может быть принято аргументированное решение о продолжении или об отказе от проекта. Главные результаты: Артефакт: Деловые обстоятельства, Артефакт: План разработки программного обеспечения и Артефакт: Список рисков (наброски).

■ Часть потока работ: Оценка контекста и риска проекта

Идентифицируйте риски в проекте, включая те риски, которые связаны с внедрением нового процесса и инструментальных средств. Результат: Артефакт: Список рисков.

■ Часть потока работ: Планирование разработки программного обеспечения

Спланируйте стадии. Главный результат – раздел «План проекта» в Артефакте: План разработки программного обеспечения. Он содержит план стадий, в котором Вы укажете главные вехи с критериями их достижения, включая критерии для потока работ Среда.

■ Часть потока работ: Планирование следующей итерации

Подробно спланируйте итерацию, включая поток работ Среда и все другие потоки работ. Главный результат: Артефакт: План итерации. В нем перечисляются все части и действия потока работ Среда и всех других основных потоков работ процесса.

■ Часть потока работ: Управление итерацией

Выполните оценку результатов использования процесса и инструментальных средств как часть оценки итерации. Результаты:

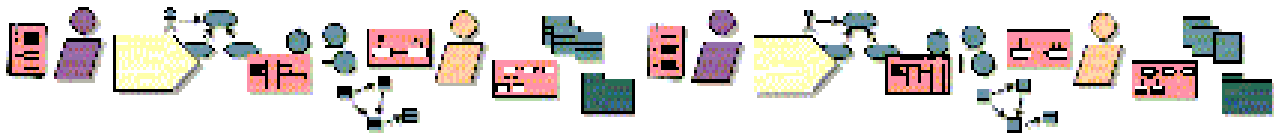
- Артефакт: Оценка итерации. Он описывает проблемы или последствия использования процесса и инструментальных средств, которые нужно учесть в последующих итерациях.
- Артефакт: Оценка состояния. Это артефакт используется для других оценок, непосредственно с итерацией не связанных.

■ Часть потока работ: Мониторинг и управление проектом

Руководитель проекта контролирует ежедневную работу, включая процесс и инструментальные средства.

■ Часть потока работ: Оценка контекста и риска проекта

В конце итерации произведите переоценку рисков, включая риски, связанные с процессом и инструментальными средствами. Некоторые риски смягчены в ходе итерации, и новые риски



идентифицированы. Основной результат - модифицированный Артефакт: Список рисков.

Требования

Специфических изменений нет.

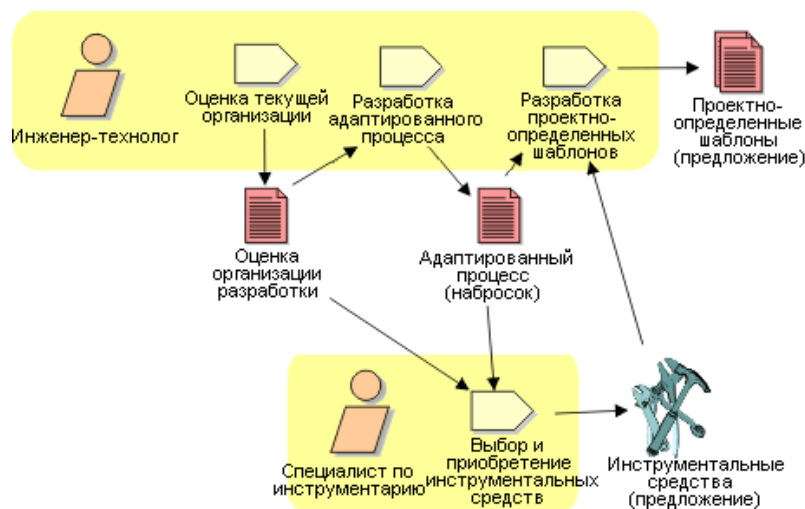
Испытание

■ Часть потока работ: Планирование испытания

Руководитель испытаний выполняет некоторое начальное планирование.

Среда

■ Часть потока работ: Подготовка среды для проекта

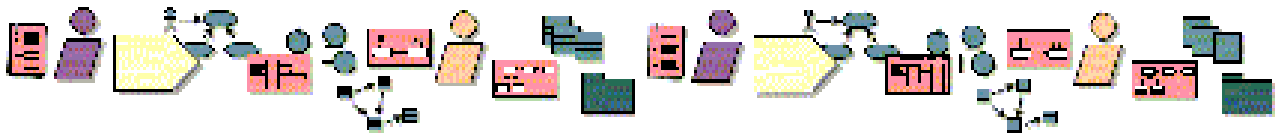


Часть потока работ «Подготовка среды для проекта» дисциплины Среда

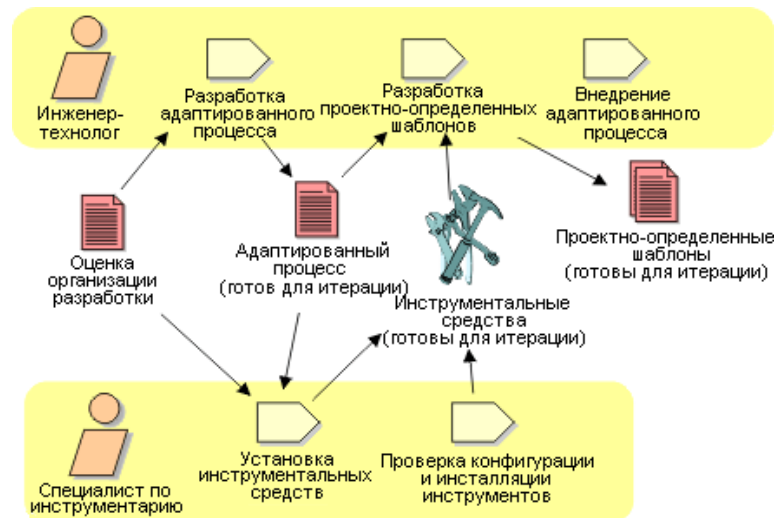
Примечание: Этот и последующие рисунки иллюстрируют содержание соответствующих частей потока работ дисциплины Среда, как они представлены в Rational Unified Process. Особенности их применения в рассматриваемом примере плана итерации отражаются в тексте.

Оцените текущее состояние организации и решите, на каких частях процесса и инструментальных средств нужно сосредоточиться в первых итерациях. Основываясь на этой оценке, примите решение о начале внедрения в проекте процесса и инструментальных средств. Результаты:

- Артефакт: Оценка организации разработки, где исследуются и документируются потребности проектной группы в уточнении процесса и в средствах разработки.
- Артефакт: Адаптированный процесс, начальная версия, которая дает краткий обзор процесса.



■ Часть потока работ: Подготовка среды для итерации

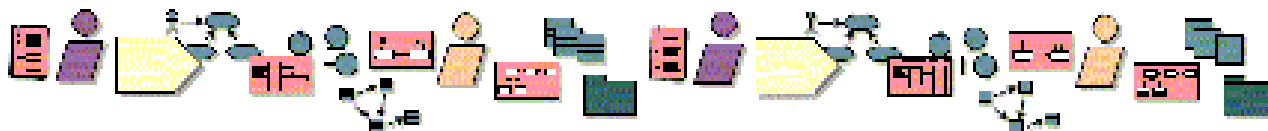


Часть потока работ «Подготовка среды для итерации» дисциплины Среда

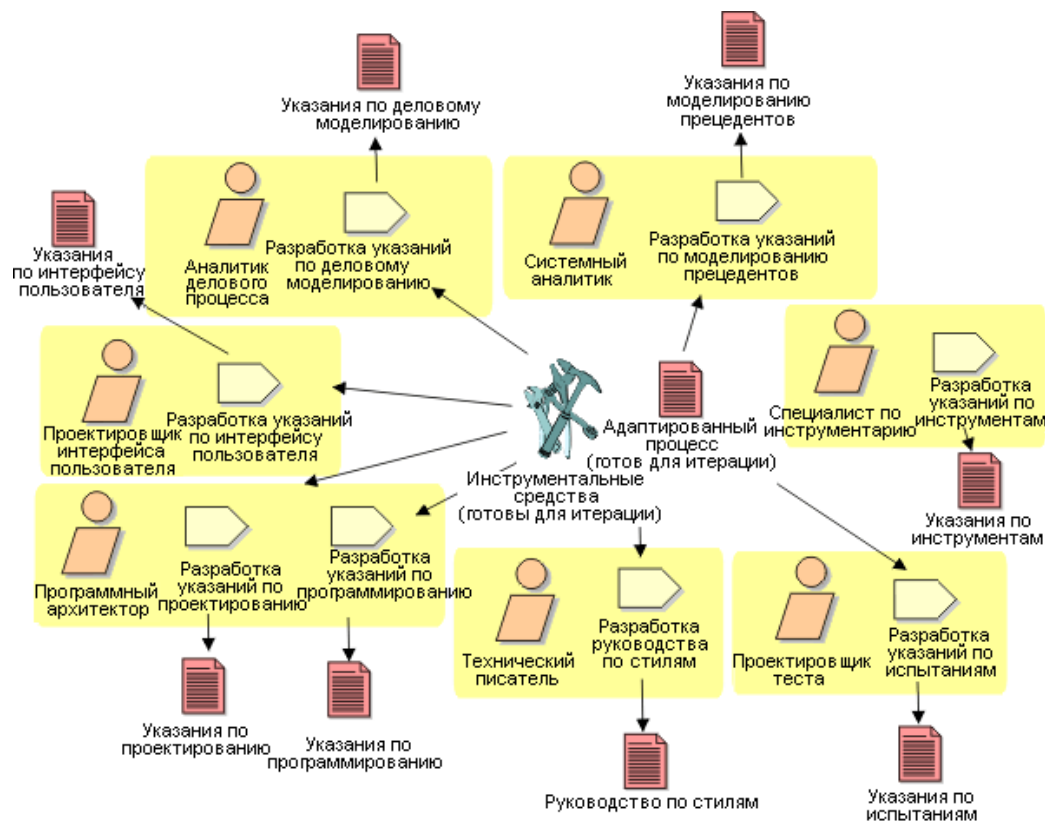
Подготовьте процесс и инструментальные средства для потока работ Требования вместе с инструментальными средствами поддержки так, чтобы члены проектной группы могли начать их использование. (Конечно, могут быть подготовлены и другие потоки работ.) Удостоверьтесь, что люди в проекте понимают, как использовать адаптированный процесс, рекомендации по моделированию прецедентов и инструментальные средства. В дополнение к стандартным учебным курсам, мы рекомендуем, чтобы Вы провели семинар, где члены проектной группы получили бы практический опыт.

Результаты выполнения части потока работ:

- Артефакт: Адаптированный процесс, где подробно описан поток работ Требования.
- Приспособленные шаблоны документов, см. Артефакт: Проектно-определенные шаблоны, для артефактов потока работ Требования, таких, как Артефакт: Видение, Артефакт: План управления требованиями и Артефакт: Запросы совладельцев.
- Инструментальные средства для потока работ Требования установлены и готовы к использованию участниками проекта.



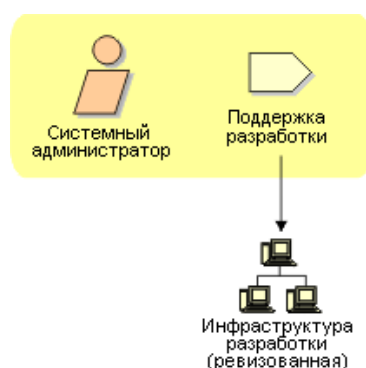
■ Часть потока работ: Подготовка указаний для итерации



Часть потока работ «Подготовка указаний для итерации» дисциплины Среда

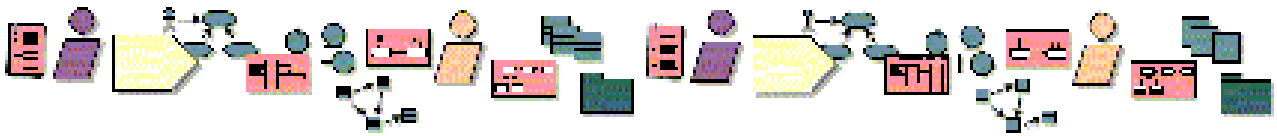
Результат: Артефакт: Указания по моделированию прецедентов.

■ Часть потока работ: Поддержка среды текущей итерации



Часть потока работ «Поддержка среды текущей итерации» дисциплины Среда

Системный администратор поддерживает разработчиков в течение итерации.



Обучение

- Все члены проектной группы посещают курсы, которые предоставляют краткий обзор RUP, чтобы получить краткий обзор жизненного цикла проекта.
- Люди, которые работают над требованиями, должны посетить курс, где они изучат подробности потока работ Требования в RUP.

Наставничество

- **Наставник процесса 50%.** Кто-то, кто действует как инженер-технолог и поддерживает руководителя проекта и других его участников при использовании и конфигурировании процесса.
- **Наставник требований 50%.** Кто-то, кто помогает в работе с требованиями, проводит семинары, рассматривает результаты и отвечает на специфические вопросы относительно управления требованиями.

