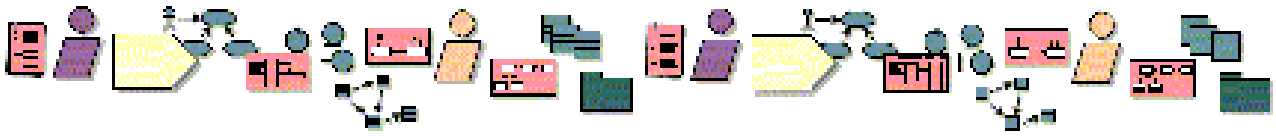


# Поток работ «Управление проектом» (продолжение)

## 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

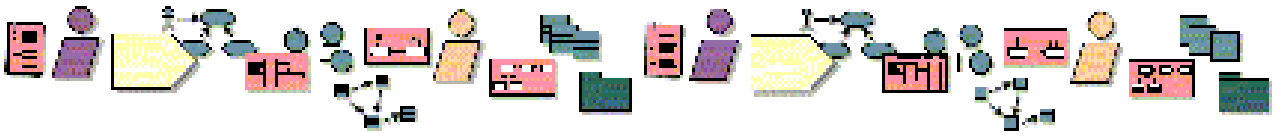
Планирование имеет дело также и с неодушевленными ресурсами, типа оборудования, инструментальных средств и бюджета, но мы не будем останавливаться на этих вопросах, потому что итерационный подход не оказывает на них существенного влияния.

### 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

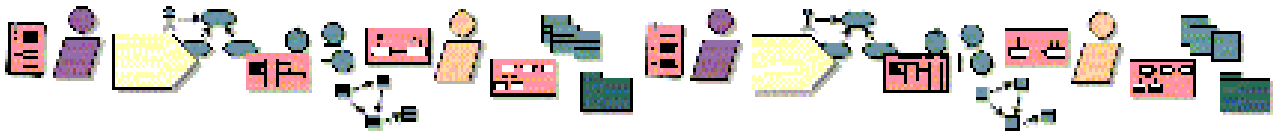
План проекта создается как можно раньше в стадии начала и модифицируется по мере необходимости.

### План итерации

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

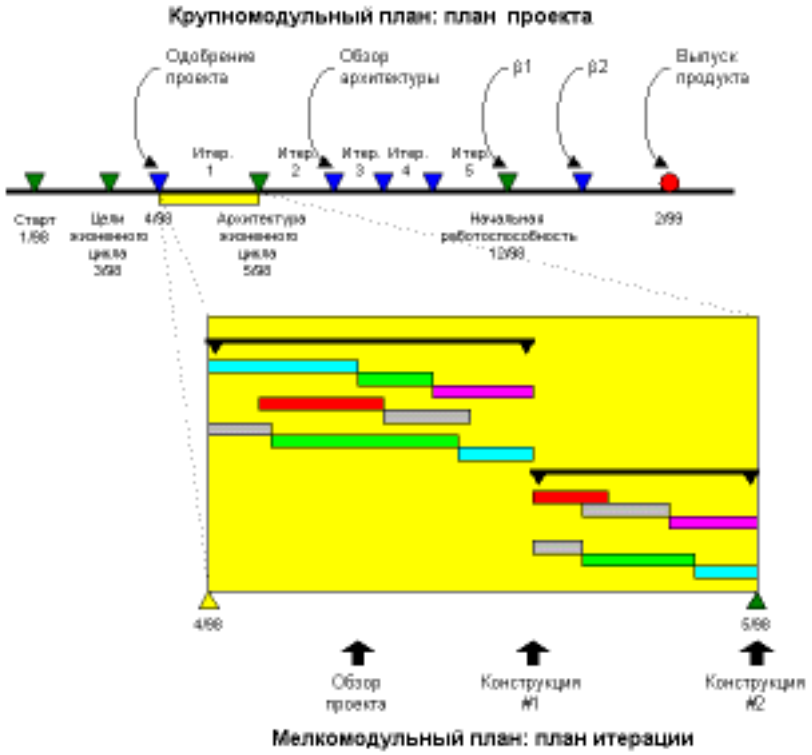
- План текущей итерации, который используется, чтобы проследить продвижение итерации
- План следующей итерации, который обычно начинают формировать во второй половине текущей итерации и который бывает готов к концу текущей итерации

План итерации формируется для определения задач и их распределения между личностями и группами с использованием традиционных методов и инструментальных средств планирования (диаграммы Гантта 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:



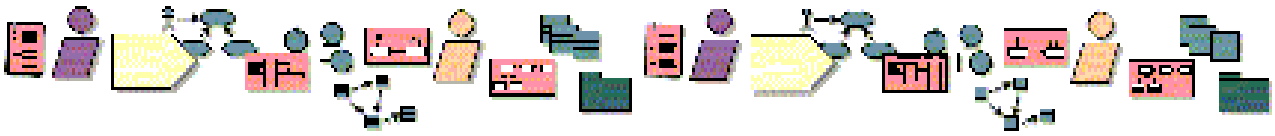

План итерации – это «увеличительное стекло», перемещаемое вдоль плана проекта

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.

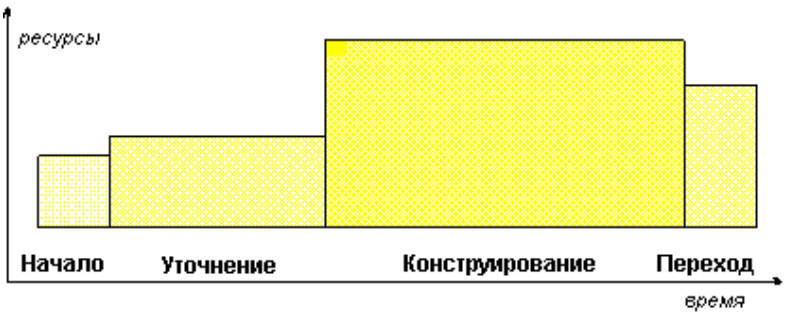
Чтобы планировать стадии проекта в начальном цикле развития, Вам, вероятно, придется сделать некоторые предположения на основе:

- 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:

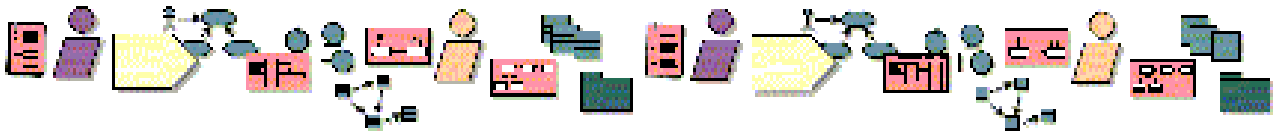
| | Начало | Уточнение | Конструирование | Переход |
|---|---|---|---|---|
| Объем работы | ~5 % | 20 % | 65 % | 10% |
| Время | 10 % | 30 % | 50 % | 10% |

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:

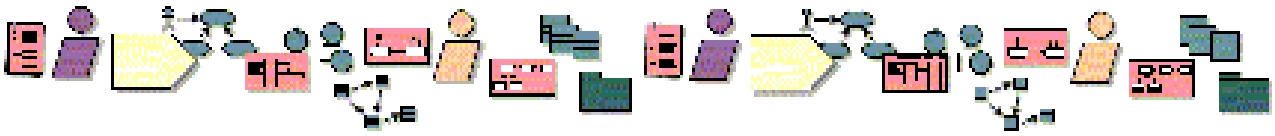
- Согласие с совладельцами при определении оценок контекста и стоимости/графика
- Понимание требований, которые подтверждаются точностью описания одного из основных прецедентов
- Степень точности оценок стоимости/графика, приоритетов, рисков и процесса развития
- Глубина и широта архитектурного прототипа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



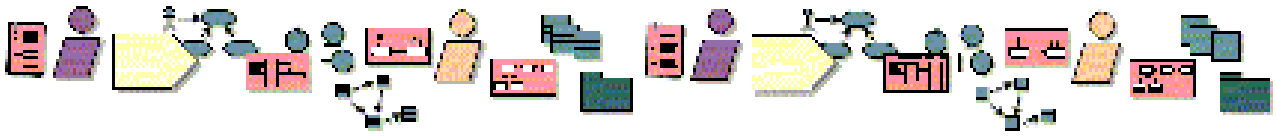
- Имеется ли устойчивое видение программы?
- Действительно ли устойчива архитектура?
- Показывает ли выполненная демонстрация, какие главные элементы риска были обнаружены и достоверно разрешены?
- Достаточно ли детализирован и точен план стадии конструирования, и основан ли он на достаточно вероятных 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

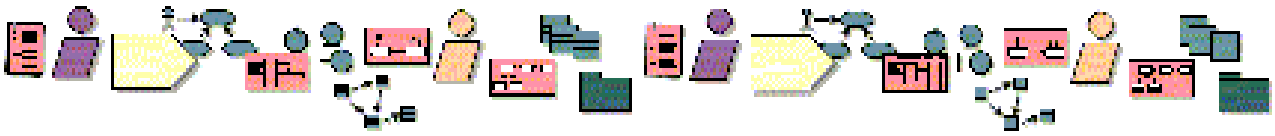
Когда будут определены длины стадий проекта, должны быть определены число итераций и их длина.

***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:

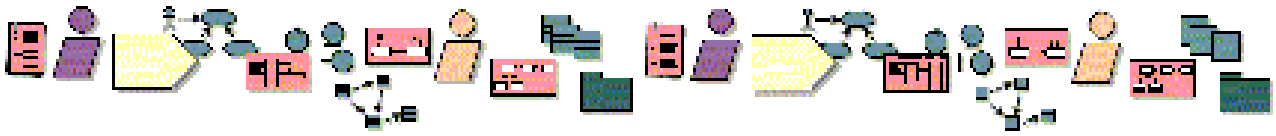
| Объем программы в строках кода (SLOC) | Число разработчиков | Продолжительность итерации |
|---|---|---|
| 10,000 | 5 | 1 неделя |
| 50,000 | 15 | 1 месяц |
| 500,000 | 45 | 6 месяцев |
| 1,000,000 | 100 | 1 год |

- **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 1месяца**, 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



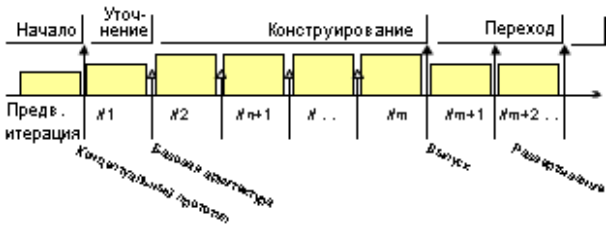
интерактивной версии Rational Unified Process описаны четыре различных стратегии построения жизненного цикла проекта.

- **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:

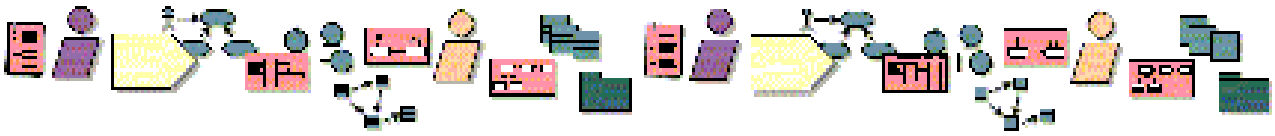
  - прикладная область известна
  - риски хорошо понятны
  - проектная группа надежна

- **Эволюционный жизненный цикл**

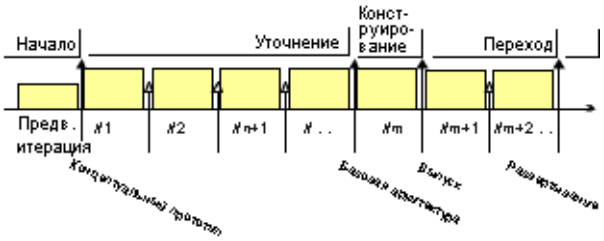
  Эволюционная стратегия отличается от пошаговой тем, что потребности пользователя не поняты полностью, и все требования не могут быть определены с первого взгляда, они уточняются в каждой следующей конструкции.

  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

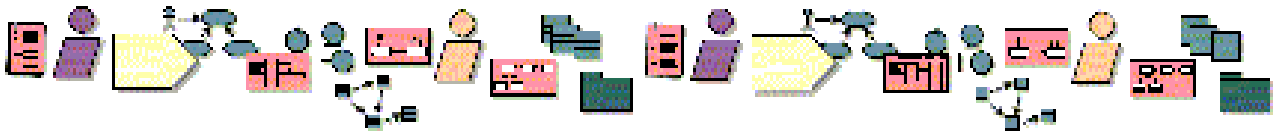
- **Жизненный цикл пошаговой поставки**

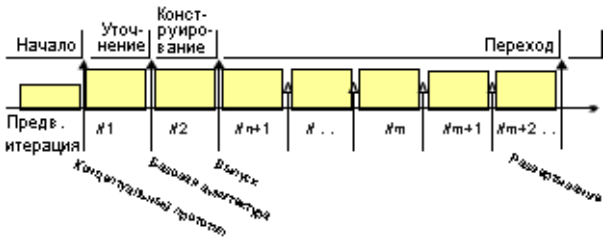
Некоторые авторы могут поэтапно поставлять заказчику постепенно возрастающие функциональные возможности. Эта стратегия может быть необходима, когда поджимают сроки поставки продукта на рынок, или когда может быть выгодной ранняя готовность некоторых главных возможностей.

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 компонуется 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.

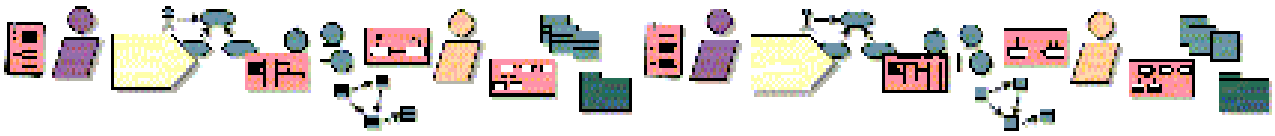
Характерны следующие итерации:

- 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 приспосабливать 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

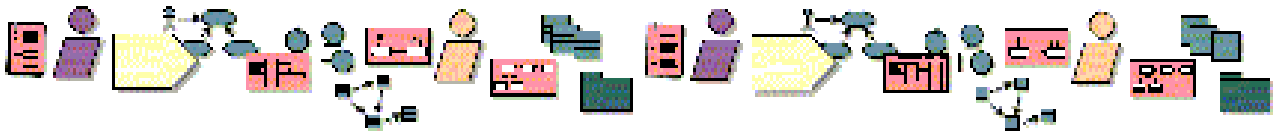
К концу стадии начала стадии могут быть запланированы более точно, принимая во внимание:

- 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

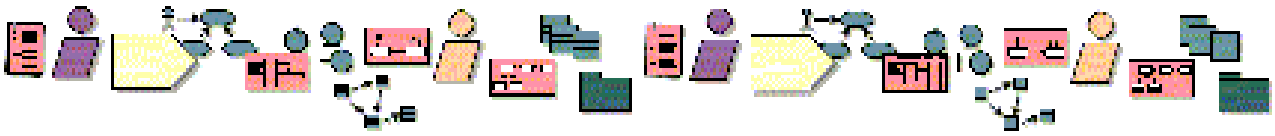
Сама по себе итерация – это набор задач, которые очень конкретно выделяются для создания выполнимой программы. Для всех итераций, кроме последней итерации перехода, это - полуфабрикат, производимый с целью смягчения рисков и доведения проекта до успешного завершения. Акцент на выполнимости программы 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 приспосабливать 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.

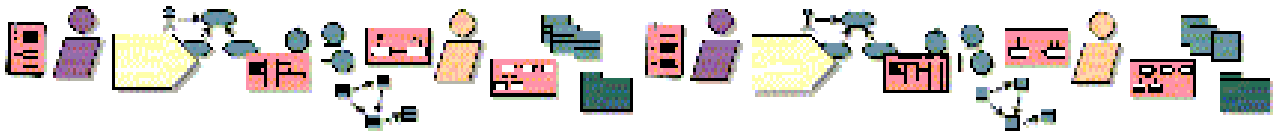
И, наконец, хотя сосредоточение на рисках важно, нужно иметь в виду главное назначение системы. Решение всех трудных проблем - это хорошо, но это не должно делаться в ущерб функциональным возможностям: удостоверьтесь, что критичные функции или услуги системы действительно охвачены (**критичность**),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

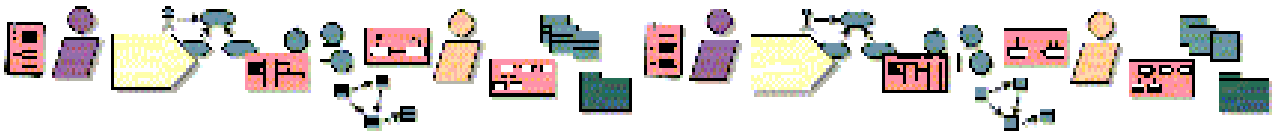
***Оценка начальной итерации***

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 недостает 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 сравненению 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.

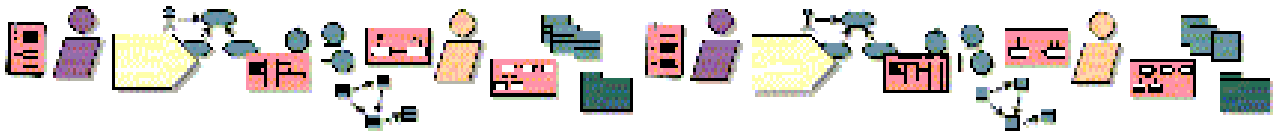
Ошибки обнаруживаются различными способами: инспектирование и обзор кода, тестирование функциональности, эксплуатационные испытания и испытания на нагрузку. Каждая методика ориентируется на обнаружение определенного набора дефектов, и каждая находит свое место. Специально эти методы обсуждаются в потоке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

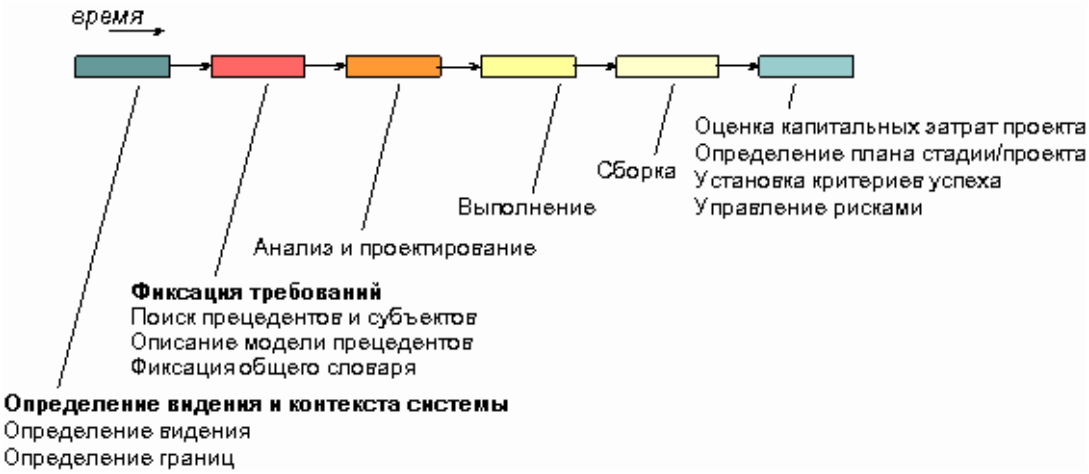
Объем, в котором выполняются эти действия, изменяется в зависимости от итерации и стадии проекта. Отдельные потоки работ (требования, анализ и проектирование, испытание и т.д.) определяют универсальные действия, которые, в свою очередь, приспосабливаются к организации в ходе процесса конфигурирования.



### *Различные виды итераций*

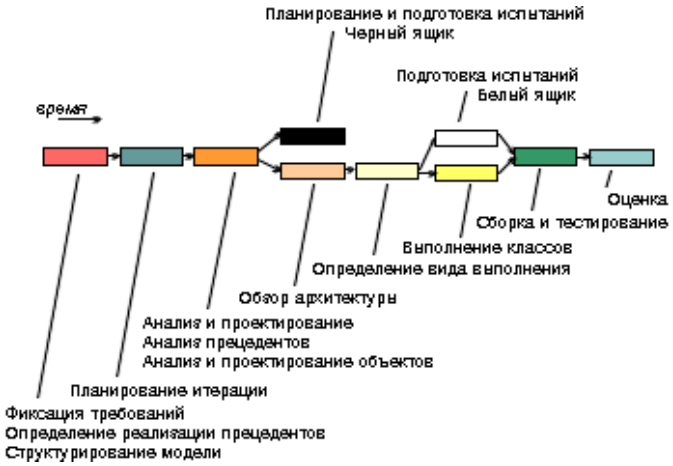
Цели стадий различны, поэтому характер итераций меняется от одной стадии к другой, даже если они проходят через одни и те же действия. Интерактивная версия Rational Unified Process предлагает брать за основу следующие типичные виды итераций:

- **В стадии начала**



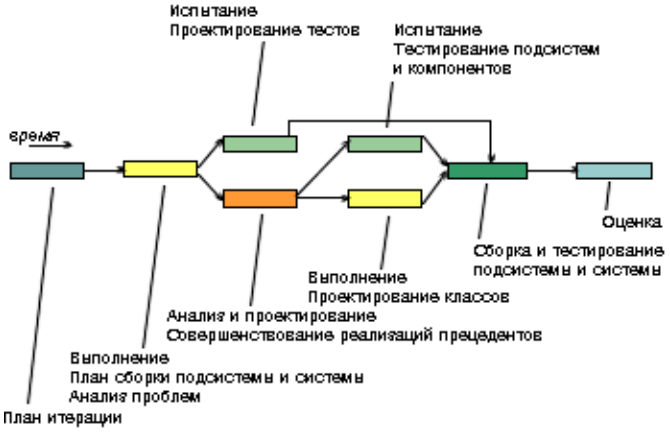

Типичная итерация стадии начала

- **В стадии уточнения**




Типичная итерация стадии уточнения

- **В стадии конструирования**




Типичная итерация стадии конструирования

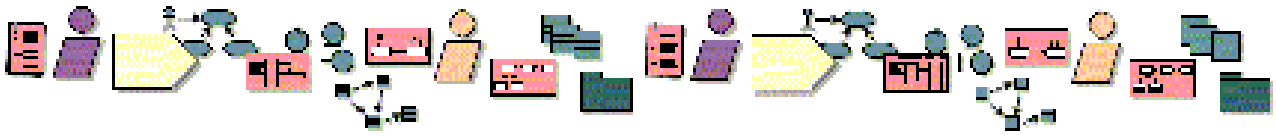
***Идентификация затрагиваемых артефактов и включение действий***

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 перечисленны 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 инсталлированы в среду вашего Microsoft Word, после чего могут настраиваться для нужд организации и конкретного проекта стандартными средствами.

## Использование Microsoft Project

Интерактивная версия Rational Unified Process содержит шаблон плана проекта.

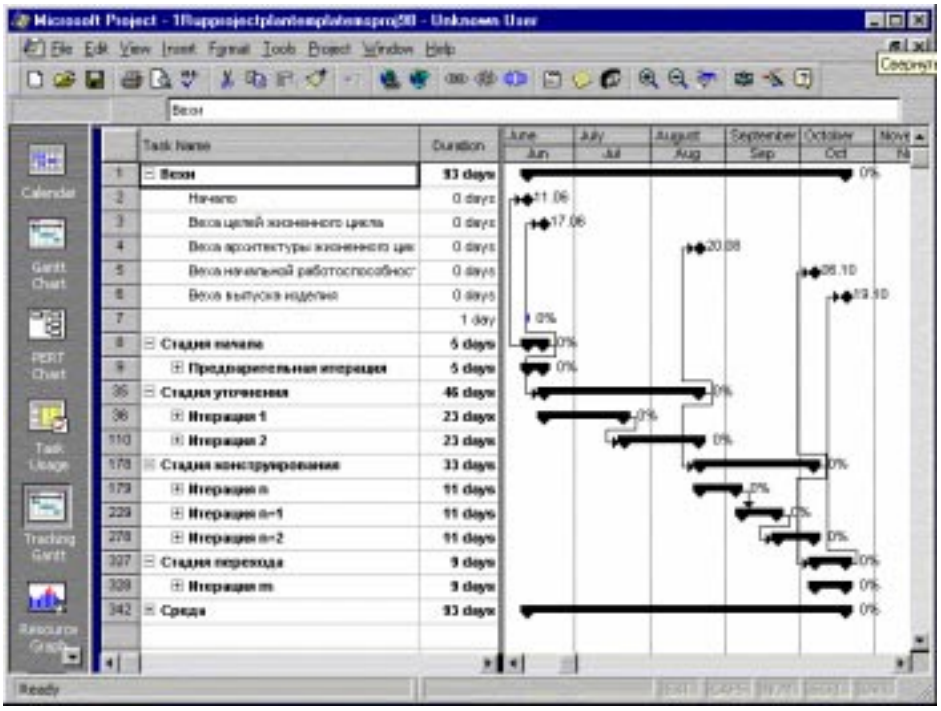
Шаблон плана проекта – это шаблон в формате Microsoft Project, который представляет типичный проект, использующий RUP. Ему предназначено стать исходной точкой для формирования плана проекта.

Связанные действия Rational Unified Process:

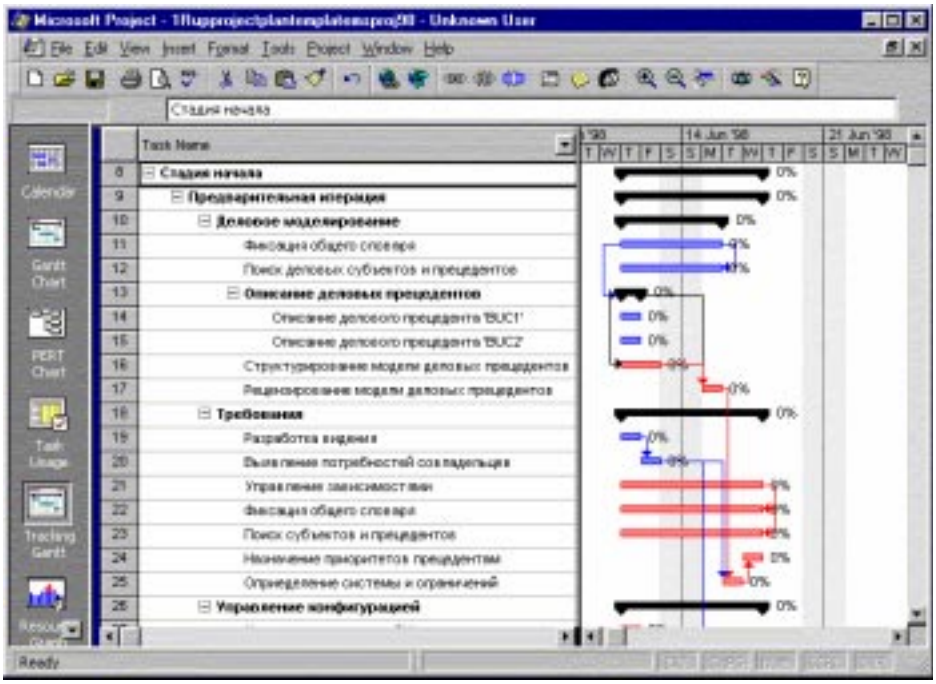
- Разработка плана проекта
- Разработка плана итерации
- Выполнение плана итерации

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.

Microsoft Project позволяет просматривать шаблон (и план) с выбранной степенью детализации и в выбранном масштабе времени. Эта возможность обеспечивает реализацию той идеи, что план итерации – это «увеличительное стекло», перемещаемое вдоль плана проекта (см. стр. 7-3).




Предопределенный шаблон Microsoft Project отображен со степенью детализации и в масштабе времени, удобными для работы с планом проекта



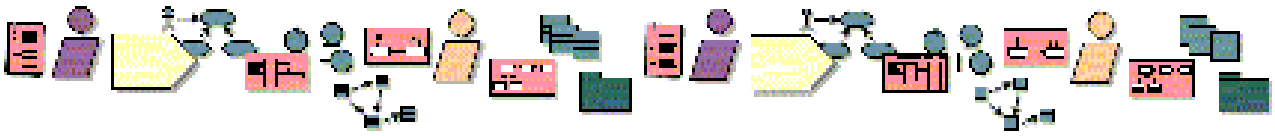

Тот же шаблон отображен со степенью детализации и в масштабе времени, удобными для работы с планом итерации (конкретно – с планом предварительной итерации стадии начала)

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.