ТЕХНОЛОГИЯ И ПРОЦЕСС РАЗРАБОТКИ ПО (Л-2)

к.п.н., доцент Касаткин Д. А.
e-mail: kasatkinda@cfuv.ru

Унифицированный процесс разработки ПО (Rational Unified Process, RUP)

- Унифицированный процесс это вариант итеративного процесса, разработанный компанией Rational Software, (сейчас подразделение IBM).
- общая модель процесса, созданная для ОО разработки с использованием UML.
- Унифицированный процесс (UP) разрабатывается с 1967 года.
  - управляемым рисками и прецедентами (требованиями);
  - архитектуро-центричным;
  - итеративным и инкрементным.
- Это сложившийся открытый процесс разработки ПО от авторов UML.
- Унифицированный процесс компании Rational (RUP) – это коммерческое расширение UP.

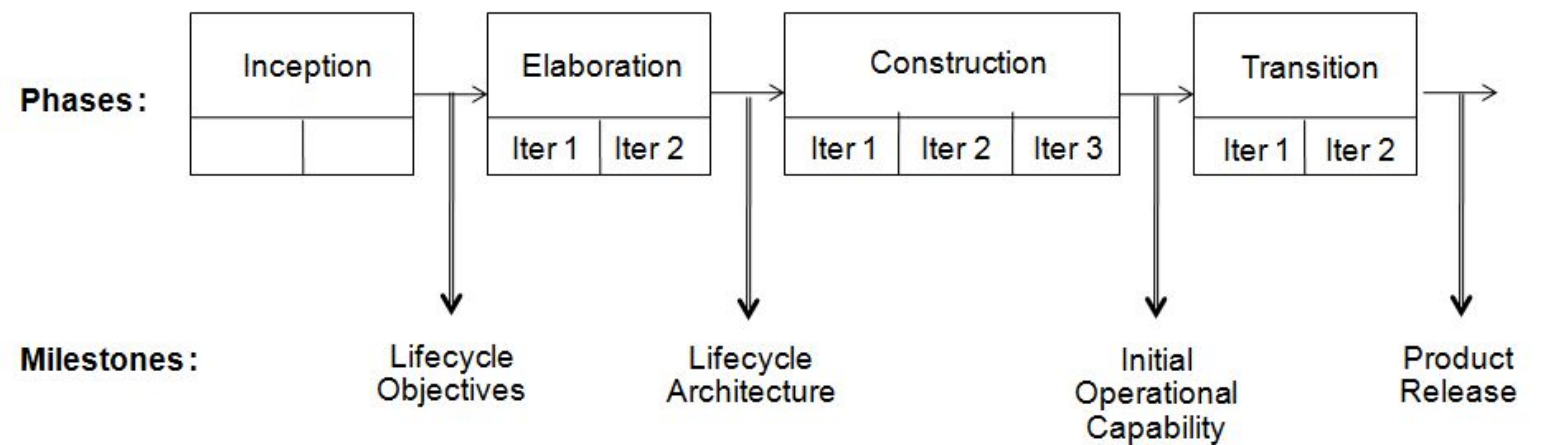
- Процесс, направляемый вариантами использования
- Архитектуро-центрированный процесс
- Итеративный и инкрементный процесс

Унифицированный процесс (UP)

- Унифицированный процесс (UP) разрабатывается с 1967 года.
  - управляемым рисками и прецедентами (требованиями);
  - архитектуро-центричным;
  - итеративным и инкрементным.
- Это сложившийся открытый процесс разработки ПО от авторов UML.
- Унифицированный процесс компании Rational (RUP) – это коммерческое расширение UP.
- Он полностью совместим с UP, но более полный и детализированный.
- UP (и RUP) должны настраиваться под каждый конкретный проект путем добавления внутренних стандартов и др.

Схема модели RUP

- Программное обеспечение разрабатывается в течении 4 этапов (фаз):
  1. фаза анализа и планирования требований (начало) (inception)
  2. фаза проектирования (развития, уточнения) (elaboration)
  3. фаза построения (разработки) (construction)
  4. фаза внедрения (перехода)



- Обычно, каждая фаза выполняется в виде отдельного проекта, целью которого является дополнение дополнительных возможностей существующей системе (разработанной в предыдущем цикле).

Фазы унифицированного процесса разработки

У каждой фазы есть

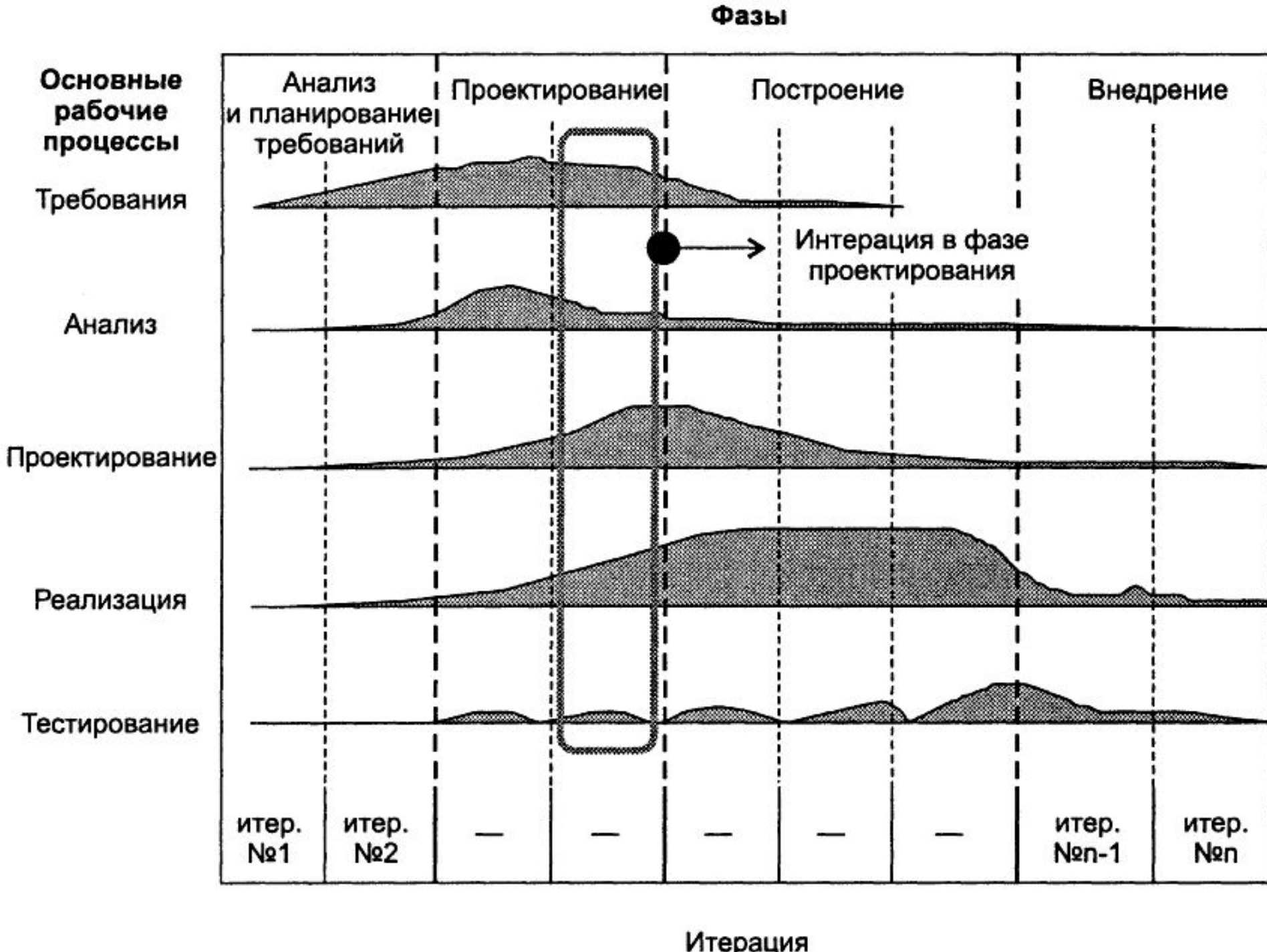
1. цель,
2. основная деятельность с акцентом на одном или более рабочих потоках, и
3. контрольная точка.

Цели и результаты фаз разработки

1. **Начало** – проект сдвигается с «мертвой точки»:
   - Результат: определяются цели жизненного цикла;
2. **Проектирование** (уточнение) – развитие архитектуры системы:
   - Результат: определяется архитектура программной системы;
3. **Построение** – построение программного обеспечения:
   - Результат: разрабатывается базовая функциональность;
4. **Внедрение** – развертывание программного обеспечения в пользовательской среде:
   - Результат: выполняется выпуск продукта и его развертывание.

пять основных рабочих потоков:

- В каждой фазе выполняется пять основных рабочих потоков:
  1. **определение требований** – выяснение того, что должна делать система;
  2. **анализ** – конкретизация и структурирование требований;
  3. **проектирование** – реализация требований в архитектуре системы (как система это делает);
  4. **реализация** – построение программного обеспечения;
  5. **тестирование** – проверяется, работает ли должным образом реализация.


Фазы
Основные рабочие процессы
Анализ и планирование требований
Проектирование
Построение
Внедрение
Требования
Интерация в фазе проектирования
Анализ
Проектирование
Реализация
Тестирование
итер. №1
итер. №2
итер. №n-1
итер. №n
Итерация

- Результатом каждого цикла является новый выпуск системы, а каждый выпуск - это продукт, готовый к поставке.
- Он включает в себя тело — исходный код, воплощенный в компоненты, которые могут быть откомпилированы и выполнены, плюс руководство и дополнительные компоненты поставки.
- Однако готовый продукт должен также быть приспособлен для нужд не только пользователей, а всех заинтересованных лиц, то есть всех людей, которые будут работать с продуктом.
- Программному продукту представляет нечто большее, чем исполняемый программный код.
- Окончательный продукт включает в себя
- требования,
- варианты использования,
- нефункциональные требования и
- варианты тестирования.
- архитектуру и визуальные модели — артефакты, смоделированные на Унифицированном языке моделирования. Фактически,
- все элементы, которые мы обсуждали в этой главе, поскольку они позволяют заинтересованным лицам — заказчикам, пользователям, аналитикам, проектировщикам, кодировщикам, тестерам и руководству — описывать, проектировать, реализовывать и использовать систему.
- Более того, именно эти средства позволяют заинтересованным лицам использовать систему и модифицировать ее от поколения к поколению.

1.Фаза анализа и планирования требований (начало)

- Целью начального этапа является определение целей и масштаба проекта, а результатом данного этапа являются цели данной итерации (цикла).
- Полученный результат должен определять общее описание и высокоуровневые возможности созданной системы:
  - выгоды для бизнеса;
  - некоторые показательные варианты использования системы;
  - ключевые риски проекта;
  - базовый план проекта, включающий затраты и сроки.
- На основе этого принимается решение о том,

2. Фаза проектирования (развития)

- На этапе развития на основе детального анализа требований проектируется архитектура системы.
- Ожидается, что в конце данного этапа будет
  - выявлено и детально описано большинство требований;
  - разработана и описана архитектура системы, с учетом выявленных на раннем этапе технических рисков.
- Кроме этого, должен быть разработан высокоуровневый план проекта, описывающий оставшиеся этапы и итерации, и текущее понимание рисков.
- К концу данного этапа принимаются критически важные инженерные решения, касающиеся выбора технологий, архитектуры и т.п. и формируется детальное понимание данного проекта.

3. Фаза построения

- На этапе построения разрабатывается и тестируется ПО.
- Результатом данного этапа является переданный заказчику программный продукт, вместе с инструкциями (пользовательскими и др.).
- Успешное завершение данного этапа означает, что была достигнута веха начальных операционных возможностей.

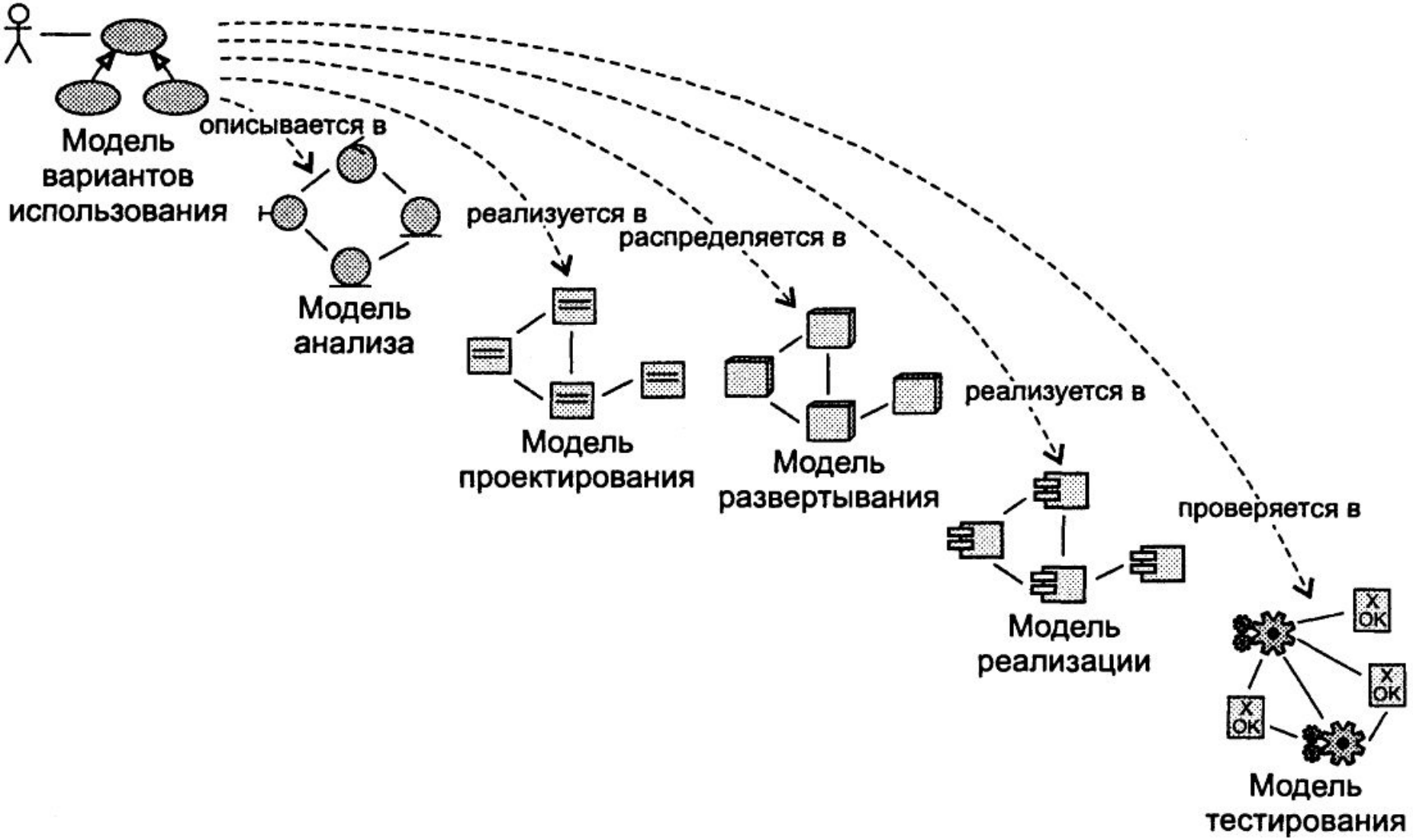
4.Фаза внедрения (перехода)

- Целью данной фазы перехода является перенос ПО из среду разработки в среду заказчика, где она должна работать.
- Это достаточно сложная задача, требующая
  - дополнительного тестирования,
  - преобразование старых данных заказчика для их использования в новой системе,
  - обучение персонала и
  - т.п.
- Успешное выполнение данной фазы приводит к достижению контрольной точки выпуска новой версии ПО.

Итерации выполнения фаз

- Хотя эти фазы выполняются последовательно, в рамках одной фазы могут выполняться несколько итераций.
- В результате каждой итерации выполняется предоставление внутренним или внешним потребителям некоторого хорошо определенного результата, который часто является частью конечной версии результата данной фазы.
- Обычно предполагается, что фаза построения будет включать несколько итераций, в каждой из которых создается работающая ПС.
- Результат каждой итерации оценивается пользователями и их отзывы используются в других итерациях

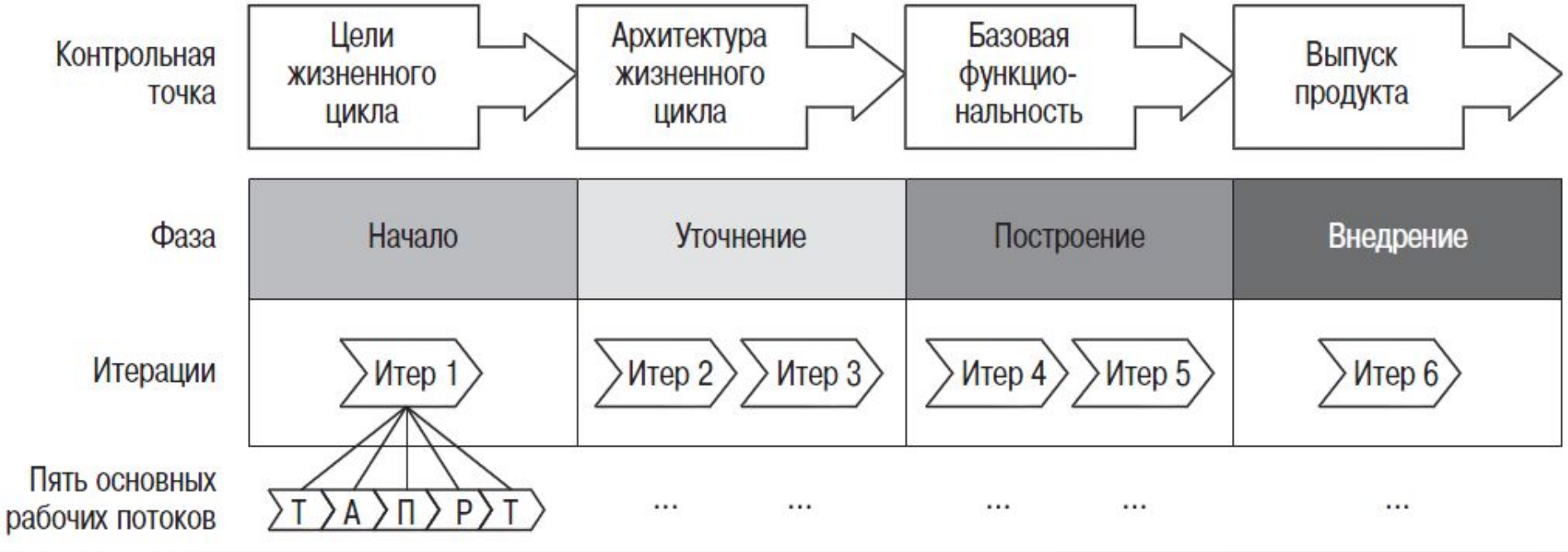
Модели Унифицированного процесса



Степень активности подпроцессов на разных этапах RUP

	Начало	Развитие	Построение	Внедрение
Требования	Высокая	Высокая	Низкая	Нет
Анализ и проектирование	Низкая	Высокая	Средняя	Нет
Реализация	Нет	Низкая	Высокая	Низкая
Тестирование	Нет	Низкая	Высокая	Средняя
Развертывание	Нет	Нет	Средняя	Высокая
Управление проектом	Средняя	Средняя	Средняя	Средняя
Управление конфигурирование	Низкая	Низкая	Высокая	Высокая

Структура UP



Фаза «Начало»

- Фаза Начало осуществляет инициирование проекта.
- Цель фазы: Начало – «сдвинуть проект с мертвой точки».
- Начало включает:
  - Обоснование выполнимости – может включать разработку технического прототипа с целью проверки правильности технологических решений или концептуального прототипа для проверки **бизнес-требований**.
  - Разработка экономического обоснования для демонстрации того, что проект обеспечит выраженную в количественном отношении коммерческую выгоду.
  - Определение основных требований для создания предметной области системы.
  - Выявление наиболее опасных рисков.

Исполнители ФАЗЫ

Основными исполнителями в данной фазе являются:

1. руководитель проекта и
2. архитектор системы.

- Основное внимание обращено на определение требований и анализ.
- Однако если принято решение о создании технического или подтверждающего концепцию прототипа, может быть проведено некоторое проектирование и реализация.
- Тестирование обычно не применяется в данной фазе, поскольку единственными программными артефактами здесь являются прототипы, которые не будут больше нигде использоваться.

№	Условия принятия	Поставляемые артефакты
1	Заинтересованные стороны согласовали цели проекта.	Общее описание, определяющее основные требования, характеристики и ограничения проекта.
2	Заинтересованные стороны определили и одобрили предметную область системы.	Исходная модель прецедентов (выполненная только на 10–20%).
3	Заинтересованные стороны определили и одобрили ключевые требования.	Глоссарий проекта.
4	Заинтересованные стороны одобрили затраты и план работы.	Исходный план проекта.
5	Руководитель проекта формировал экономическое обоснование проекта.	Экономическое обоснование.
6	Руководитель проекта провел оценку рисков.	Документ или база данных оценки рисков.
7	Посредством технических исследований и/или создания прототипа была подтверждена выполнимость.	Один или более одноразовых прототипов.
8	Архитектура намечена в общих чертах.	Документ с исходной архитектурой.

Цели фазы «проектирования»

1. Основная цель - создание исполняемой базовой версии архитектуры;
2. детализация оценки рисков;
3. определение атрибутов качества (скорости выявления дефектов, приемлемые плотности дефектов и т. д.);
4. выявление прецедентов, составляющих до 80% от функциональных требований;
5. создание подробного плана фазы **Построение**;
6. формулировка предложения, включающего ресурсы, время, оборудование, штат и стоимость.

Фаза проектирование

- В фазе Проектирование основное внимание в каждом из основных рабочих потоков обращено на следующее:
  - определение требований – детализация предметной области системы и требований;
  - анализ – выяснение, что необходимо построить;
  - проектирование – создание стабильной архитектуры;
  - реализация – построение базовой версии архитектуры;
  - тестирование – тестирование базовой версии архитектуры.
- Основное внимание в фазе Проектирование направлено на рабочие потоки определения требований, анализа и проектирования.
- Реализация приобретает значение в конце фазы при создании исполняемой базовой версии архитектуры

Условия принятия	Поставляемые артефакты
Создана гибкая надежная исполняемая базовая версия архитектуры.	Исполняемая базовая версия архитектуры.
Исполняемая базовая версия архитектуры демонстрирует, что важные риски были выявлены и учтены.	Статическая UML модель. Динамическая UML модель. UML модель прецедентов.
Представление продукта стабилизировалось.	Общее описание проекта
Оценка рисков пересмотрена.	Обновленная оценка рисков
Экономическое обоснование проекта пересмотрено и одобрено всеми заинтересованными сторонами.	Обновленное экономическое обоснование проекта.
Создан достаточно детальный план проекта, что обеспечило возможность сформулировать реалистичную заявку на затраты времени, денег и ресурсов в следующих фазах. Заинтересованные стороны одобрили план проекта.	Обновленный план проекта.
Проведена проверка экономического обоснования проекта согласно плану проекта.	Экономическое обоснование проекта.
Заинтересованные стороны достигли соглашения о	Подписанный документ

Фаза «построение»

- Построение превращает исполняемую базовую версию архитектуры в за конченную рабочую систему.
- Цель фазы Построение –

1. завершить определение требований,
2. анализ и проектирование и
3. развить исполняемую базовую версию архитектуры, созданную в фазе Уточнение, в завершенную систему.

- Главная проблема Уточнения – *поддержание целостности архитектуры системы.*
- Очень часто при установлении сроков поставки и переходе к написанию кода начинается пренебрежение формальностями, что приводит к нарушению архитектурного представления, низкому качеству конечной системы и высоким затратам на обслуживание.

Построение – контрольная точка: Базовая функциональность

Условия принятия	Поставляемые артефакты
Программный продукт достаточно стабилен и качественен для распространения среди пользователей.	Программный продукт. UMLмодель. Тестовый комплект.
Заинтересованные стороны одобрили и готовы к введению программного продукта в свое окружение.	Руководство для пользователя. Описание данной версии.
Расхождения реальных расходов с предполагаемыми приемлемы.	План проекта.

Фаза внедрение

- Внедрение направлено на развертывание законченной системы в сообществе пользователей.
- Внедрение начинается, когда завершено бета-тестирование и система окончательно развернута.
- Сюда относится устранение всех дефектов, обнаруженных при бетатестировании, и подготовка к массовому выпуску программного обеспечения на все пользовательские сайты.

Цели фазы внедрения

- Цели этой фазы можно обобщить следующим образом:

1. исправление дефектов;
2. подготовка пользовательских сайтов под новое программное обеспечение;
3. настройка работоспособности программного обеспечения на пользовательских сайтах;
4. изменение программного обеспечения в случае возникновения не предвиденных проблем;
5. создание руководств для пользователей и другой документации;
6. предоставление пользователям консультаций;
7. проведение послепроектного анализа.

Внедрение – на что обращено внимание

- Основное внимание концентрируется на рабочих потоках реализации и тестирования.
- Для исправления всех ошибок проектирования, выявленных при бета-тестировании, выполняется существенный объем проектирования.
- Надо стремиться к тому, чтобы в фазе Внедрение рабочие потоки определения требований и анализа оставались практически не задействованными

Внедрение – контрольная точка: Выпуск продукта

Условия принятия	Поставляемые артефакты
Бета-тестирование завершено, необходимые изменения сделаны и пользователи согласны с тем, что система успешно развернута.	Программный продукт.
Сообщество пользователей активно использует продукт.	План поддержки пользователя.
Стратегии поддержки продукта согласованы с пользователями и реализованы.	Обновленные руководства для пользователей.

Выводы по методологии RUP

Достоинства	Недостатки	Подходящие типы проектов
Все преимущества итеративной модели. Предоставляет гибкий каркас для выполнения большого разнообразия проектов.	Для каждого проекта должен быть спроектирован его процесс.	Может быть применен к широкому набору проектов, т.к. это позволяет его гибкость.

5. Модель временных ящиков (Timeboxing Model)

- Для ускорения разработки может быть использована параллельность (одновременность) выполнения разных итераций.
- Новая итерация начинается прежде чем буде выпущена ПС созданная на данной итерации и усл-но разработка новой версии (варианта) происходит параллельно с разработкой текущей версии.
- Путем организация начала следующей итерации до того, как завершится предыдущая итерация возможно уменьшить среднее время предоставления ПП в итерациях.
- Однако, чтобы поддерживать параллельное выполнение, каждая итерация должна быть правильно структурирована и соответствующим образом должны быть сформированы команды исполнителей.

Модель временных ящиков

- Модель временных ящиков предлагает подход, соответствующий параллельной разработке.
- В модели временных ящиков основной единицей разработки является временной интервал (time box), имеющий фиксированную продолжительность.
- Т.к. эта продолжительность является фиксированной, то ключевым фактором выбора требований или возможностей, которые должны быть разработаны во временных ящиках, является то, что может поместиться в данный временной ящик.
- Это находится в противоречии с обычными итеративными подходами, в которых вначале выбирается функциональность, а затем определяется время ее программной реализации.

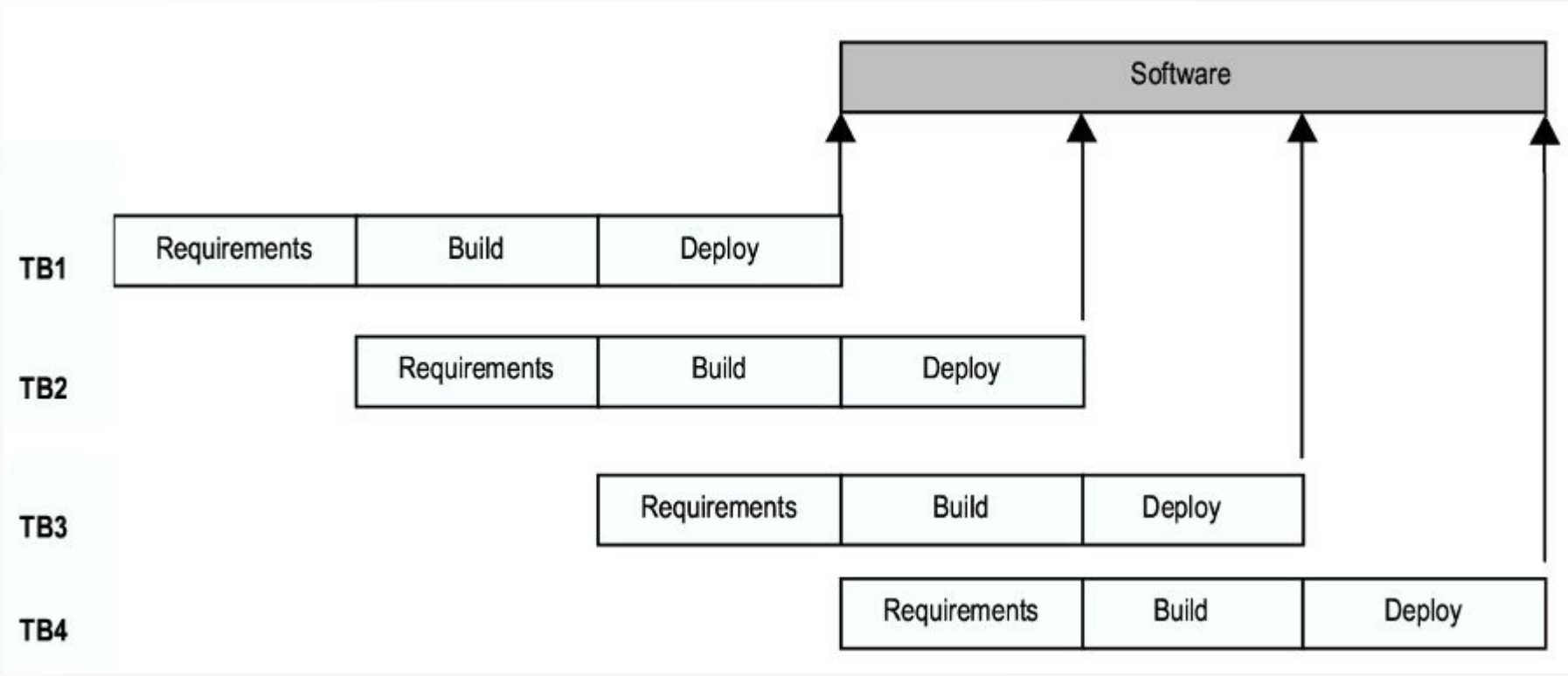
Временные интервалы

- Каждый временной ящик делится на последовательность этапов, как и в модели водопада.
- Каждый этап выполняет некоторую точно определенную задачу для данной итерации и создает точно определенный результат.
- Данная модель требует, чтобы продолжительность каждого этапа, т.е. время, требуемое для завершения задачи данного этапа, была примерно одинаковой.

Специальные команды разработчиков

- Более того, данная модель требует, чтобы были организованы специальные команды для выполнения каждого этапа.
- Т.о., команда организованная для некоторого этапа выполняет только задачи данного этапа – задачи для других этапов выполняются другие специальные команды.
- Это очень сильно отличается от других итеративных моделей, в которых неявно предполагается, что одна и та же команда выполняет все различные задачи проекта или итерации.
- Если имеются итераций во временных ящиках, разделенных на этапы равной продолжительности, и специальные команд, которые могут выполнять такие этапы, то появляется возможность организации конвейерной обработки (pipelining).

Схема модели временных ящиков



Задачи разных команд

Requirements Team	Requirements Analysis for TB1	Requirements Analysis for TB2	Requirements Analysis for TB3	Requirements Analysis for TB4
Build Team		Build for TB1	Build for TB2	Build for TB3	Build for TB4
Deployment Team			Deployment for TB1	Deployment for TB2	Deployment for TB3

Выводы по модели временных ящиков (Timeboxing)

Достоинства	**Недостатки**	**Подходящие типы проектов**
Такие же, как и у итеративной модели Планирование итераций делать несколько проще Очень короткое время предоставления ПО.	Управление проектом становится более сложным. Размер команды увеличивается. Усложнения (упущения) могут вести к потерям.	Когда очень важно короткое время разработки ПО. Когда существует гибкость группировки возможностей ПО. Стабильная архитектура.

6. Гибкие процессы разработки ПО

- набор подходов к разработке ПО, использующих
  - итеративную разработку;
  - динамическое формирование требований;
  - обеспечение их реализации путем постоянного взаимодействия внутри самоорганизующихся рабочих групп, состоящих из специалистов различного профиля.
- начали создаваться с начала 90-х годов в ответ на требование детального документирования и бюрократические процессы других подхода (в особенности модели водопада).
- это легковесные подходы в отличие от тяжеловесных подходов (водопад, RUP и т.п.).

Гибкая процессы разработки (*agile software development*)

- Набор подходов к разработке программного обеспечения, ориентированных на
  - использование итеративной разработки и
  - динамическое формирование требований и
  - обеспечение их реализации в результате постоянного взаимодействия внутри самоорганизующихся рабочих групп, состоящих из специалистов различного профиля.
- Существует несколько методик, относящихся к классу гибких методологий разработки, в частности, известны как гибкие методики
  - экстремальное программирование,
  - DSDM,
  - Scrum.

Принципы гибкой разработки

Гибкие подходы основываются на следующих базовых принципах [www.extremeprogramming.org]:

1. Работающее ПО является основным средством измерения развития проекта.

2. В таком случае, для развития проекта требуется, чтобы ПО разрабатывалось и предоставлялось быстро путем добавления небольших улучшений.

3. Даже поздние изменения в требованиях должны быть приняты (модель разработки на основе небольших приращений функциональности помогает их принимать).

4. Личное общение является более предпочтительным, чем обмен документами.

5. Непрерывная обратная связь с потребителем и вовлечение потребителя в разработку является необходимым для создания высококачественного ПО.

6. Простое проектирование, которое развивается и улучшается со временем, является лучшим подходом в сравнении с выполнением заранее детального проектирования, которое учитывает все возможные сценарии использования создаваемого ПО.
7. Сроки передачи готового ПО определяются квалифицированными командами талантливых разработчиков (а не путем распоряжений).

Разновидности гибких подходов

- XP,
- Agile,
- Lean,
- Scrum,
- Kanban,
- Theory of Constraints,
- System Thinking,
- Flow-Based Product Development
- и т.д.

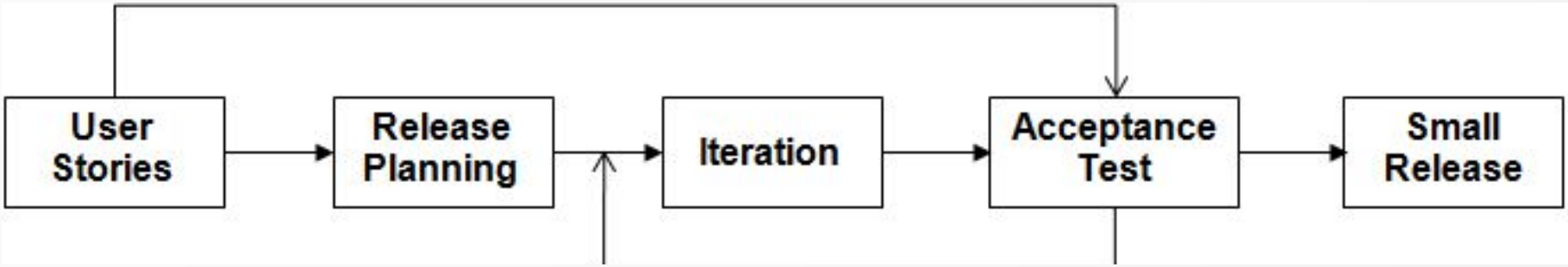
Экстремальное программирование (eXtreme Programming, XP)

- один из популярных и широко используемых подходов гибкой разработки ПО;
- предполагает, что изменения неизбежны и вместо борьбы с ними, разработка должна быть готова быстро их реализовать.
- считается, что для приспособления к изменениям, процесс разработки должен быть ***облегченным*** и

- В XP ПО разрабатывается итеративно и не создается детальная и многочисленной документации, которую трудно поддерживать.
- Основное внимание личному общению, простоте и обратной связи, для гарантирования того, что желаемые изменения быстро и правильно отражались в программах.

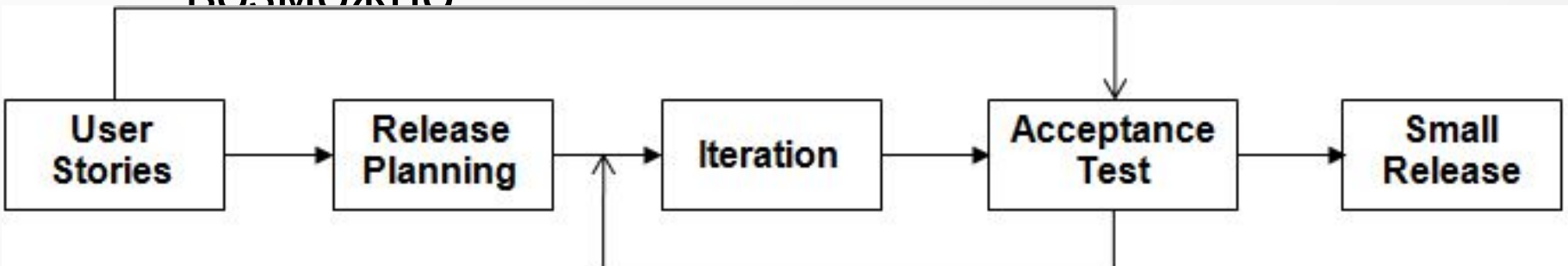
Схема XP модели процесса

- Проект начинается с «историй пользователей», которые являются короткими (в несколько предложения) описаний сценариев, которые потребители и пользователи хотели бы, чтобы поддерживало разрабатываемое ПО.

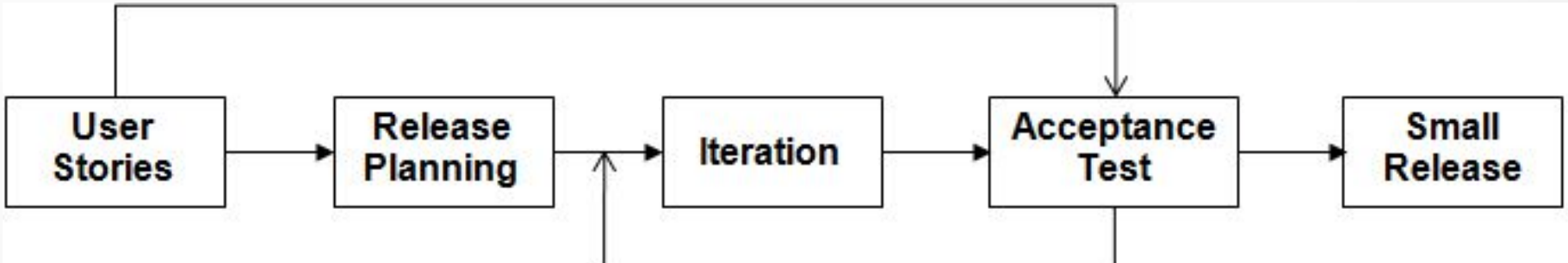


Истории пользователей

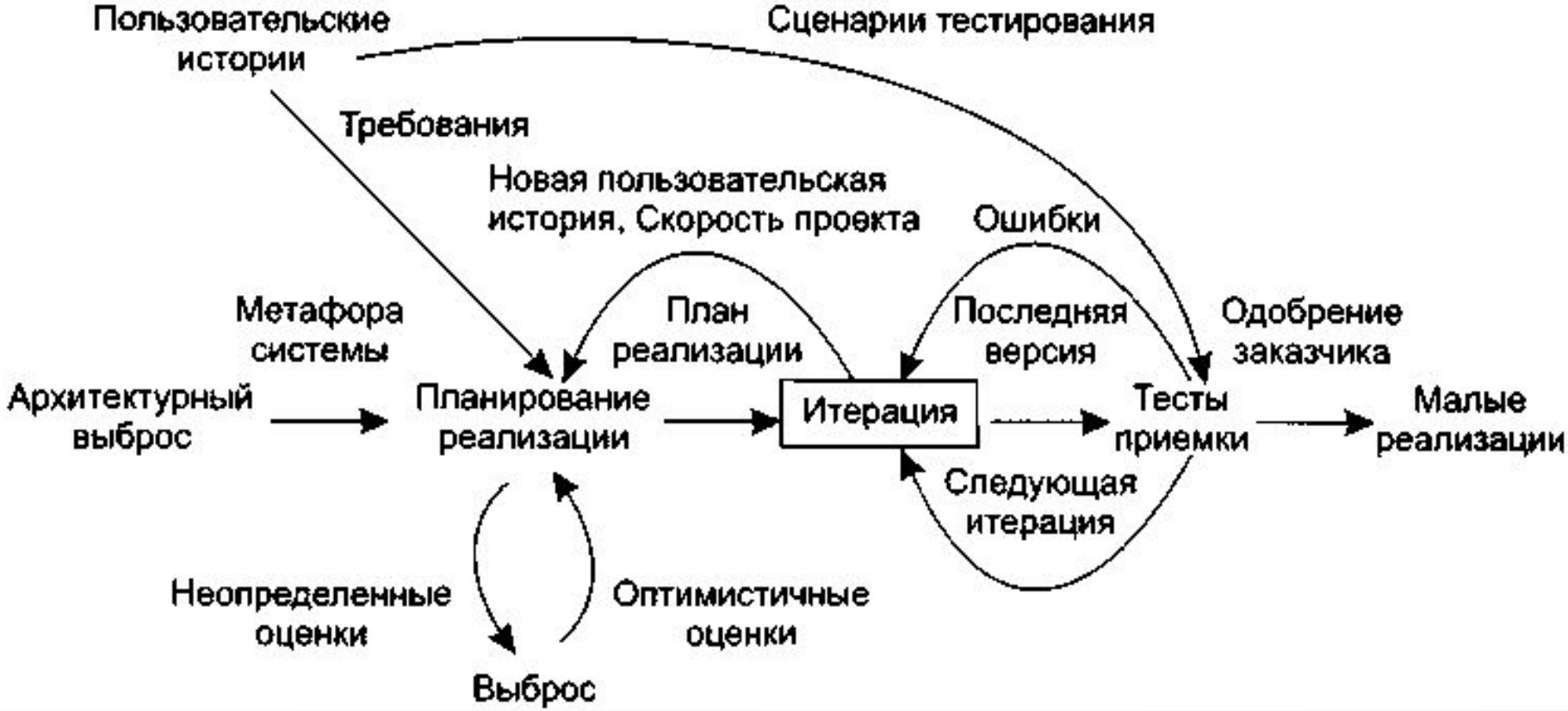
- отличаются от традиционных спецификаций (детальных описаний) требований в основном уровнем детальности:
  - не содержат детальные требования, которые будут определены только тогда, когда данная история будет реализована;
  - разрешается подробные требования предоставлять так поздно, настолько это возможно



- Каждая история записывается на отдельной карточке, так чтобы их можно было просто группировать.
- Назначенная команда разработчиков оценивает, как долго потребуется реализовать историю пользователя.
- Данные оценки являются грубыми и обычно задаются в неделях.

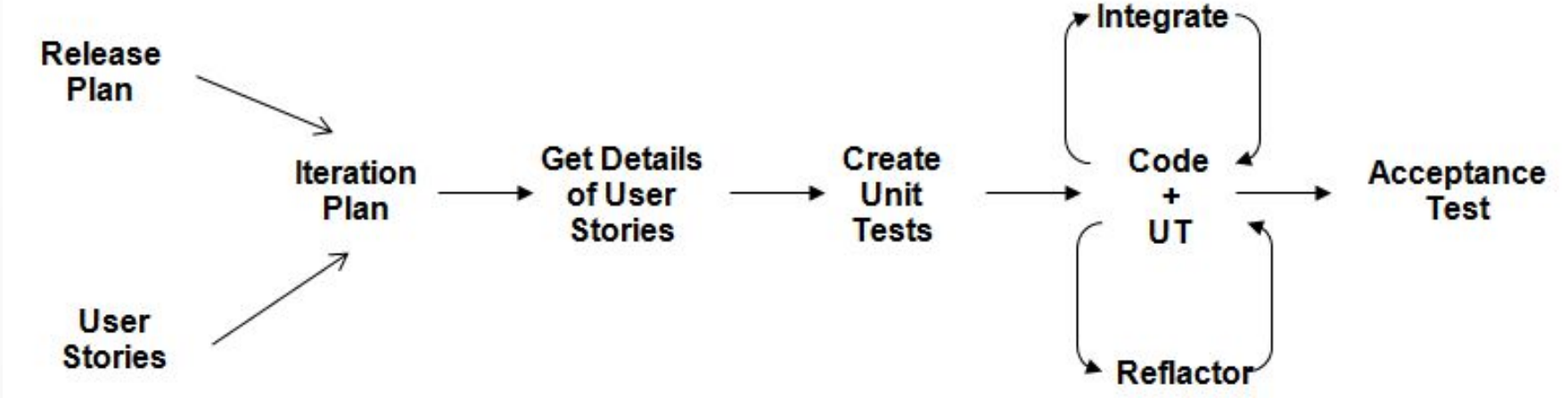


- С помощью таких оценок и историй выполняется планирование выпуска версий ПО, которое определяет какие истории должны быть реализованы в каких вариантах и даты их окончания.
- Поощряется частый выпуск небольших версий ПО, а для создания версий используются итерации.
- На основе историй также разрабатываются, ***приемочные*** тесты, которые используются для тестирования версий ПО перед их выпуском.


Пользовательские истории
Сценарии тестирования
Требования
Новая пользовательская история, Скорость проекта
Ошибки
Метафора системы
План реализации
Последняя версия
Одобрение заказчика
Архитектурный выброс
Планирование реализации
Итерация
Тесты приемки
Малые реализации
Следующая итерация
Неопределенные оценки
Оптимистичные оценки
Выброс

Итерации разработки

- Разработка выполняется в результате нескольких итераций. Каждая итерация продолжается несколько недель.
- Итерация начинается с планирования итерации, в ходе которого выбираются истории пользователей, которые должны быть реализованы на данной итерации.
- Для выполнения разработки определяются детали выбранных историй.
- Создаются тесты.
- Выполняется кодирование и тестирование



Особенности разработки в итерациях

1. Рассчитывается, что разработка выполняется парами программистов (т.н. парное программирование).
2. Предполагается, что до реального написания кода программных единиц создаются автоматизированные тесты, а затем код должен составляться так, что проходить эти тесты.
3. Используется разработка управляемая тестами (test-driven development), которая отличается от обычного подхода, когда сперва составляют код, а затем думают о том, как его

4. Поощряется создание простых решений и их дальнейшее изменение.
   - Ожидается, что проект решения разработанный заранее может в некоторый момент времени стать не подходящим для дальнейшей разработки.
5. Поощряется частая интеграция разных программных единиц.

Дополнительные правила выполнения итерации

- В XP имеется много других правил, таких, как:

1. Распределение прав между программистами и потребителями.
2. Взаимодействие между членами команда и использование метафор.
3. Доверие и открытость для всех заинтересованных лиц.
4. Коллективное владение кодом, при котором любая пара может изменить

5. Командное управление
6. Построение быстрых пробных решений, для апробации трудных с технических и архитектурных вопросов, для исследования некоторого подхода, способа ликвидации ошибок.
7. устанавливаются правила,
   - как проводить собрания разработчиков.
   - как должен начинаться день разработчиков.

- Выбор разных правил при выполнении текущей итерации определяется достигнутыми результатами предыдущей итерации.

Лекция 3