Адаптивные модели процесса разработки программного обеспечения

Лекция 10

Прогностические процессы

- Все ранее рассмотренные модели соответствуют так называемым *прогностическим* (*тяжеловесным*) процессам разработки ПС
- Они предполагают планирование всего объема работ и, соответственно, достаточно большой объем документации

Прогностические процессы

- Основная цель таких процессов – отделить успешные практики разработки и сопровождения ПО от конкретных людей, умеющих их применять
- Многочисленные вспомогательные действия имеют целью выполнение успешной разработки с помощью имеющихся работников, не обязательно являющихся суперпрофессионалами

Адаптивные процессы

- Альтернативой такому подходу являются *адаптивные* или *облегченные, «живые» (agile)* процессы разработки
- Они не требуют столь жесткой регламентации, допускают возможность частых и существенных изменений требований заказчиков

Адаптивные процессы

- Адаптивные процессы делают упор на использование хороших разработчиков, а не хорошо отлаженных процессов разработки
- Они избегают фиксации четких схем действий, чтобы обеспечить б***о***льшую гибкость в каждом конкретном проекте и не требуют создания дополнительных промежуточных документов

История

- В феврале 2001 года на лыжном курорте The Lodge at Snowbird в горах Юты несколько известных разработчиков ПО (Kent Beck, Martin Fowler, Alistair Cockburn и др.) пришли к соглашению о необходимости документального оформления новых идей в организации процесса разработки
- Ими был согласован и представлен профессиональному сообществу документ под названием *Agile Manifesto*

Текст Agile-манифеста

- «Мы постоянно открываем для себя более совершенные методы разработки программного обеспечения, занимаясь разработкой непосредственно и помогая в этом другим.
- Благодаря проделанной работе мы смогли осознать, что:

Идеи

- **Люди и взаимодействие** важнее **процессов и инструментов**
- **Работающий продукт** важнее **исчерпывающей документации**
- **Сотрудничество с заказчиком** важнее **согласования условий контракта**
- **Готовность к изменениям** важнее **следования первоначальному плану**

- То есть, не отрицая важности того, что справа, мы всё-таки больше ценим то, что слева.»

Принципы

- Agile Manifesto декларирует следующие 12 принципов «живой» разработки:
  - наивысшим приоритетом является **удовлетворение заказчика** посредством ранней и постоянной поставки ценного ПО;
  - **изменяющиеся требования** приветствуются, даже на поздних стадиях разработки;

Принципы

- **частая поставка** рабочего программного обеспечения (каждый месяц или неделю или ещё чаще);
- тесное, ежедневное **общение заказчика с разработчиками** на протяжении всего проекта;
- проектом занимаются **мотивированные личности**, которые обеспечены нужными условиями работы, поддержкой и доверием;

Принципы

- самым эффективным методом передачи информации команде разработчиков и внутри неё является **личное общение**;
- **работающее программное обеспечение** — лучший измеритель прогресса;
- спонсоры, разработчики и пользователи должны все время выдерживать **постоянный темп**;

Принципы

- постоянное внимание улучшению **технического мастерства и удобному дизайну**;
- **простота** — искусство не делать лишней работы;
- лучшие технические требования, дизайн и архитектура получаются у **самоорганизующейся** команды;
- **постоянная адаптация** к изменяющимся обстоятельствам.

Адаптивные методологии

- **Crystal Methods** (1992 г.) – семейство методологий, послужившее отправной точкой в развитии идей адаптивной разработки; наиболее известная методология этого семейства Crystal Clear

Адаптивные методологии

- **Agile Unified Process** – упрощенная версия RUP, разработанная Скоттом Амблером
- **Agile Data Method** – группа итеративных методов разработки программного обеспечения, в которых требования и решения достигаются в рамках сотрудничества разных кросс-функциональных команд

Адаптивные методологии

- **DSDM** (Dynamic Systems Development Method, 1994 г.) – итеративный и инкрементный подход, придающий особое значение продолжительному участию в процессе пользователя/потребителя; основан на концепции быстрой разработки приложений (RAD)
- **Экстремальное программирование** (XP, 1995 г.) – декларирует двенадцать основных приёмов программирования

Адаптивные методологии

- **Feature driven development** (FDD, 1997 г.) — функционально-ориентированная разработка. Используемое в FDD понятие функции или свойства (feature) системы достаточно близко к понятию прецедента использования, существенное отличие — это дополнительное ограничение: «каждая функция должна допускать реализацию не более, чем за две недели»

Адаптивные методологии

- **Scrum** (1995 г.) – методология, делающая основной акцент на общении с заказчиком, на общении внутри команды, на достижении самоорганизации и высокого уровня самосовершенствования

XP-МОДЕЛЬ

Экстремальное программирование

- *Экстремальное программирование* (eXtreme Programming, XP-процесс) – одна из наиболее популярных адаптивных моделей
- Авторы методологии – Кент Бек, Уорд Каннингем, Мартин Фаулер и другие
- XP-процесс ориентирован на разработку качественного продукта группами малого и среднего размера в условиях неопределенных или быстро меняющихся требований

XP-процесс

- Основная идея XP-процесса – устранить высокую стоимость внесения изменений. Это достигается путем резкого (до двух недель) сокращения длительности отдельных итераций
- Базовыми действиями являются:
  - кодирование,
  - тестирование,
  - выслушивание заказчика,
  - проектирование

Принципы XP

- Высокий динамизм разработки обеспечивается следующими принципами:
  - непрерывная связь с заказчиком,
  - простота выбираемых решений,
  - быстрая обратная связь на основе оперативного тестирования,
  - профилактика рисков

Практики XP

- Реализация этих принципов достигается за счет использования следующих методов:
  - *Метафора* – вся разработка ведется на основе простой, общедоступной истории о том, как работает система
  - *Простое проектирование* – принимаются наиболее простые из возможных проектные решения

Практики XP

- *Непрерывное тестирование* как отдельных модулей, так и системы в целом; входным критерием для написания кода является отказавший тестовый вариант
- *Реорганизация (Refactoring)* – улучшение структуры системы при сохранении ее поведения
- *Парное программирование* – код пишется двумя программистами на одном компьютере

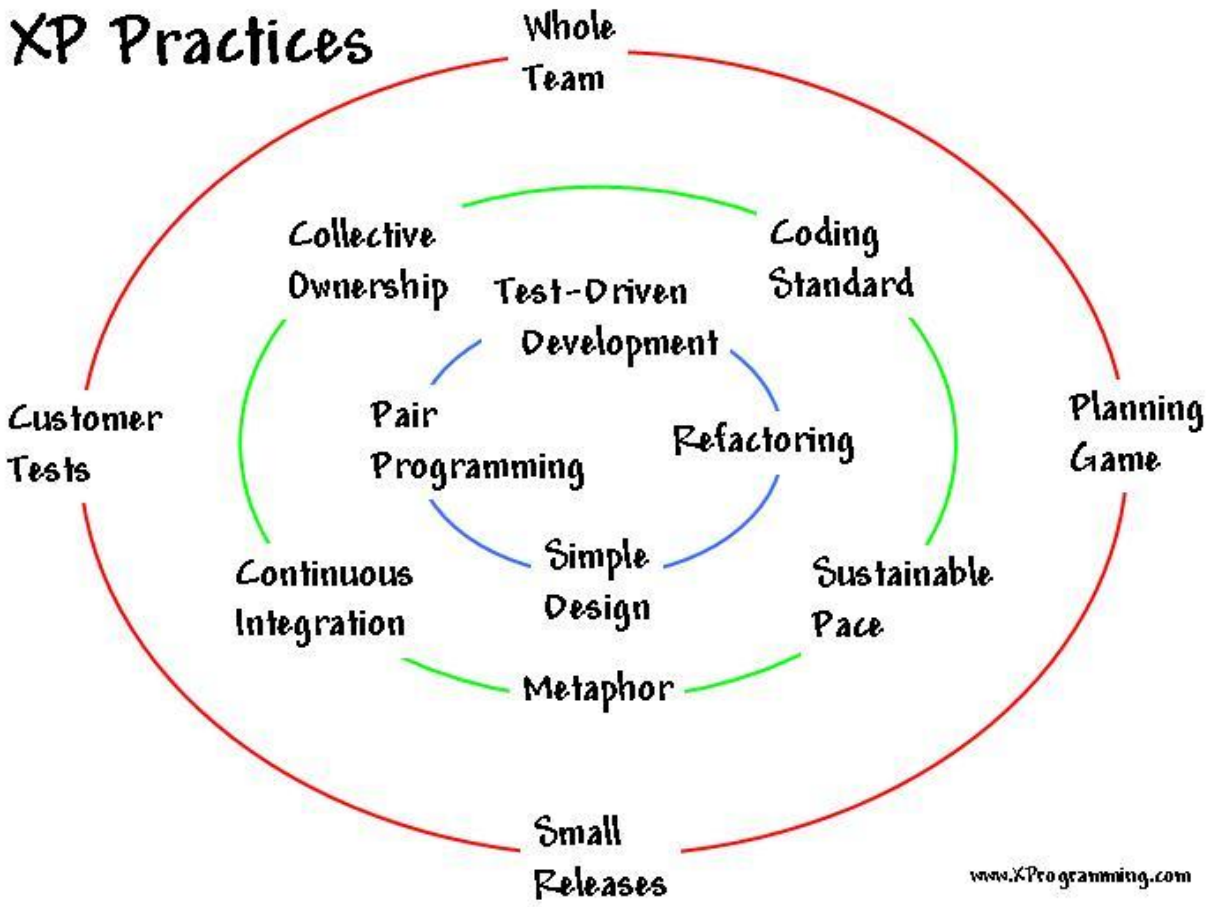
Практики XP

- *Коллективное владение кодом* – любой разработчик может улучшить код любого модуля системы
- *Непрерывная интеграция* – система интегрируется как можно чаще; непрерывное регрессионное тестирование гарантирует сохранение функциональности при изменении требований

Практики XP

- *Локальный заказчик* – в группе все время должен находиться компетентный представитель заказчика
- *Стандарты кодирования* – должны выдерживаться правила, обеспечивающие одинаковое представление кода во всех частях системы

XP в картинках



SCRUM-МОДЕЛЬ

Scrum-модель

- Является еще одним примером адаптивного процесса разработки
- Основные идеи модели сформулировали Хиротака Такеути и Икудзиро Нонака в 1986 году

Основная идея

- Экспериментальный факт: проекты, над которыми работают небольшие, кросс-функциональные команды, обычно систематически производят лучшие результаты

Основная идея

- Такеуки и Ноната объяснили это как «подход регби» и ввели и сам термин «scrum» - «*толкотня; схватка вокруг мяча (в регби)*»
- Впервые метод Scrum был представлен в документированном виде в 1996 году совместно Сазерлендом и Швабером

Роли

- Главные действующие роли:
  - *ScrumMaster*, тот кто занимается процессами и работает в качестве руководителя проекта,
  - *Владелец Продукта*, человек, который представляет интересы конечных пользователей и других заинтересованных в продукте сторон,
  - *Команда*, которая включает разработчиков

Этапы разработки

- Процесс разработки разбивается на отдельные этапы определенной длительности – *спринты* (обычно,15-30 дней)
- Каждому спринту предшествует этап, который называется *product backlog* – документирование запросов на выполнение работ

Планирование спринта

- Запросы на выполнение работ определяются на этапе *совета по планированию спринта – sprint planning meeting*

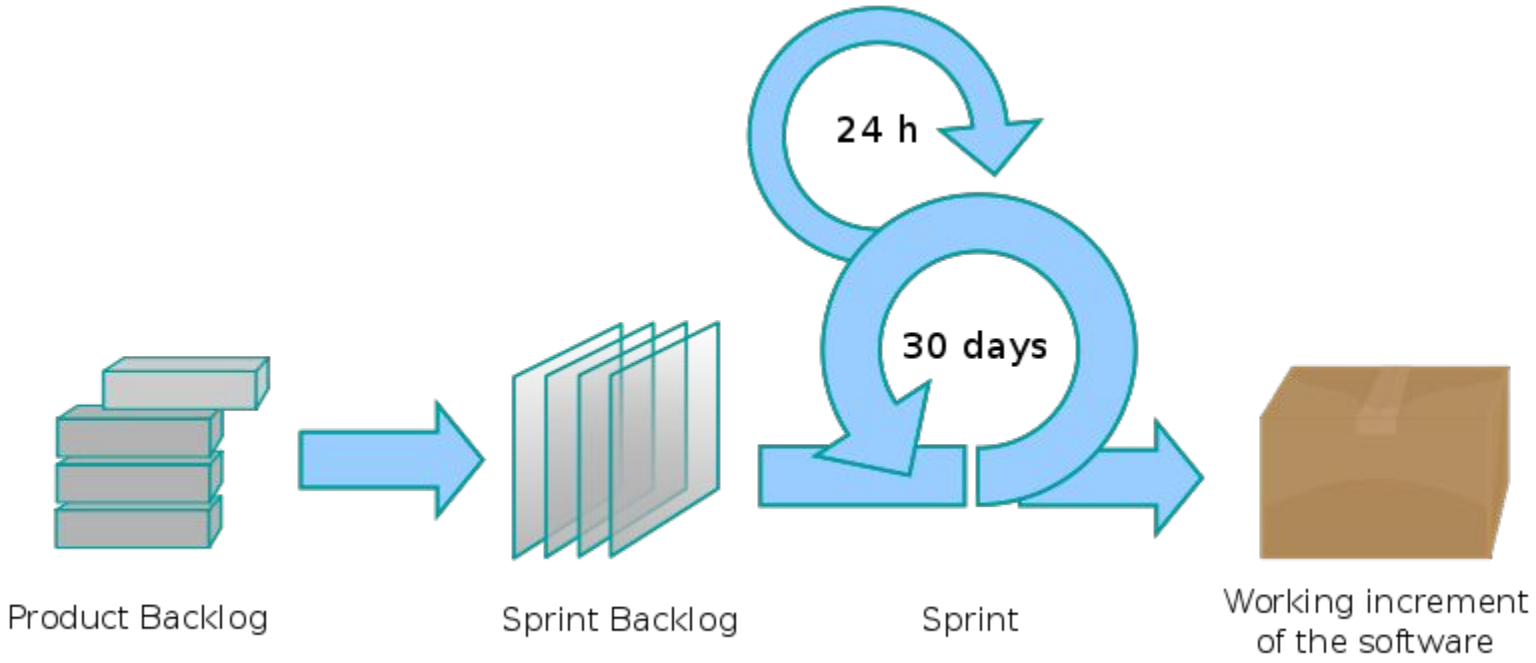
Планирование спринта

- На протяжении этого собрания *Владелец Продукта* информирует о заданиях, которые должны быть выполнены
- Команда определяет, сколько из желаемого они могут выполнить, чтобы завершить необходимые части на протяжении следующего спринта

Выполнение спринта

- Во время спринта команда выполняет определенный фиксированный список заданий - *backlog items,* наращивая функциональность программного продукта
- На протяжении этого периода никто не имеет права менять список требований к работе, что следует понимать, как заморозку требований (*requirements*) во время спринта

Scrum в картинках



RAD-МОДЕЛЬ

RAD-модель

- Модель быстрой разработки приложений (Rapid Application Development) является примером адаптивного процесса в рамках реализации инкрементной стратегии
- Основателем RAD считается сотрудник IBM Джеймс Мартин, который в 1980-х годах сформулировал основные принципы RAD, основываясь на идеях Барри Бойема и Скотта Шульца

Цели RAD-модели

- Основными целями RAD-модели процесса разработки ПО являются:
  - высокая скорость разработки;
  - низкая стоимость;
  - высокое качество

Основные принципы RAD

1. Работа ведется группами; типичный состав группы - руководитель, аналитик, два программиста, технический писатель.
2. Разработка базируется на моделях; моделирование позволяет оценить проект и выполнить его декомпозицию на составные части, каждая из которых может разрабатываться отдельной RAD-группой.

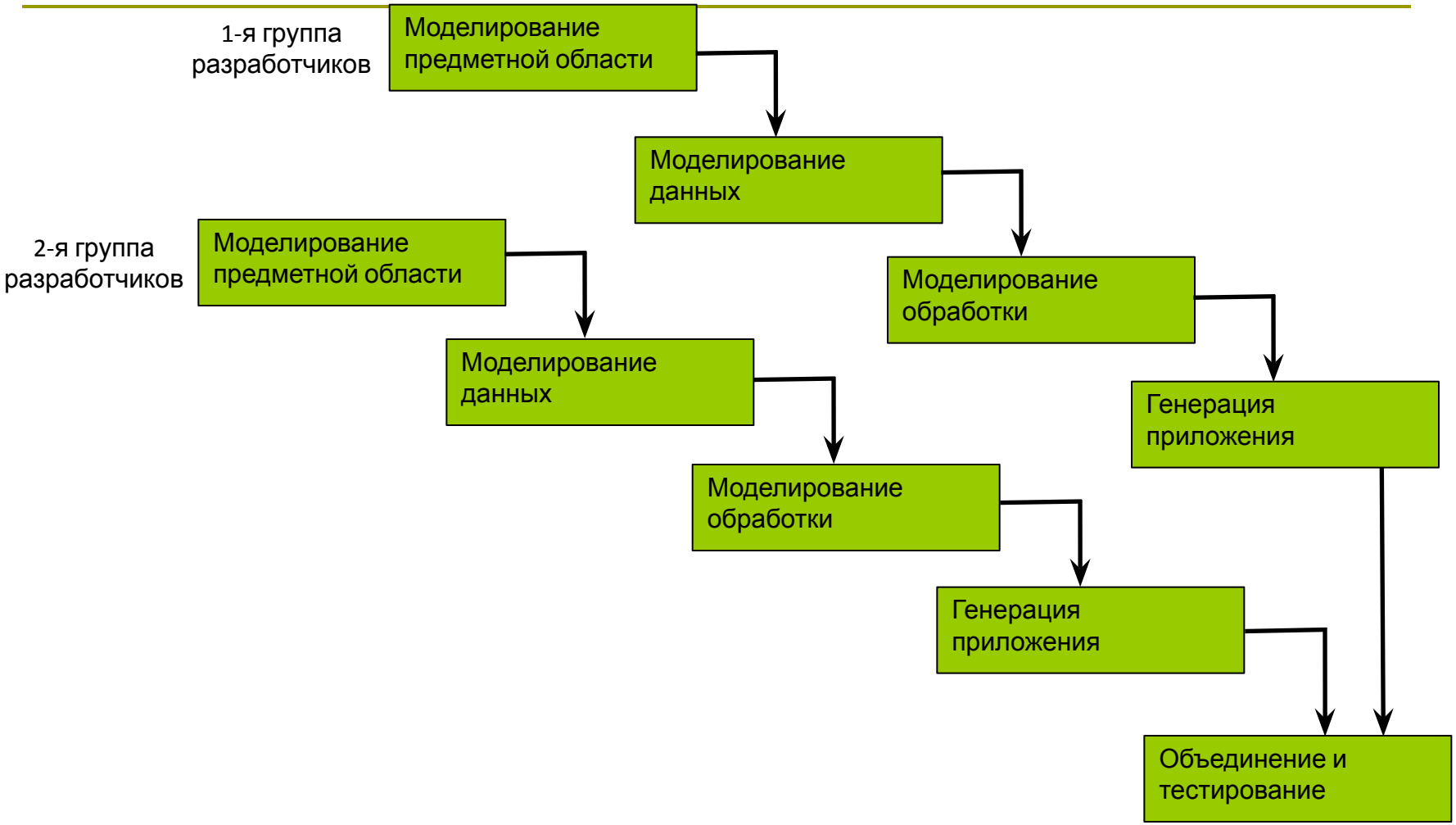
Основные принципы RAD

3. Разработка системы и предъявление ее заказчику осуществляется в виде последовательности развиваемых прототипов
4. RAD-группа всегда работает только над одним прототипом. Это обеспечивает единство целей, лучшую наблюдаемость и управляемость процессом разработки

Основные принципы RAD

5. RAD-группы должны использовать общие стандарты
6. Обязательно финальное тестирование полной системы
7. Обязательно использование инструментальных средств, автоматизирующих процесс разработки – визуальных сред проектирования и программирования

RAD-модель


1-я группа разработчиков
Моделирование предметной области
Моделирование данных
Моделирование обработки
Генерация приложения
2-я группа разработчиков
Моделирование предметной области
Моделирование данных
Моделирование обработки
Генерация приложения
Объединение и тестирование

Прототипы

- Любой из прототипов реализует определенную часть функциональности, требуемой от конечного продукта; каждый последующий прототип включает всю функциональность, реализованную в предыдущем прототипе, с добавлением новой
- Число прототипов определяется на основе учета разных параметров – размера проекта, анализа рисков, пожеланий заказчика и т. д.

Прототипы

- Традиционно для проектов ПО средней сложности разрабатываются три прототипа:
  - первый содержит весь пользовательский интерфейс с нулевой функциональностью; он дает возможность утвердить у заказчика экранные и отчетные формы;
  - второй прототип содержит реализованную на 70-80% функциональность системы
  - третий прототип содержит полностью реализованную функциональность

Итерации

- Основаниями для очередной итерации в процессе разработки являются:
  - *Замечания заказчика*. Если замечания носят характер исправлений, они учитываются в следующем прототипе, если же изменяются требования, то выполняется переоценка проекта и корректируются сроки и стоимость проекта

Итерации

- *Детализация*. Выполняется программирование нереализованной части системы в соответствии с составленным планом.
- *Анализ результатов программирования*. Исправляются ошибки, повышается эффективность программного кода и т. д.

Когда применяется RAD

- Применение технологии RAD целесообразно, когда:
  - требуется выполнение проекта в сжатые сроки (90 дней); быстрое выполнение проекта позволяет создать систему, отвечающую требованиям сегодняшнего дня
  - нечетко определены требования к ПО; в большинстве случаев заказчик весьма приблизительно представляет себе работу будущего программного продукта и не может четко сформулировать все требования к ПО

Когда применяется RAD

- проект выполняется в условиях ограниченности бюджета; разработка ведется небольшими RAD-группами в короткие сроки, что обеспечивает минимум трудозатрат и позволяет вписаться в бюджетные ограничения
- интерфейс пользователя (GUI) есть главный фактор; RAD-технология дает возможность продемонстрировать интерфейс в прототипе, причем достаточно скоро после начала проекта

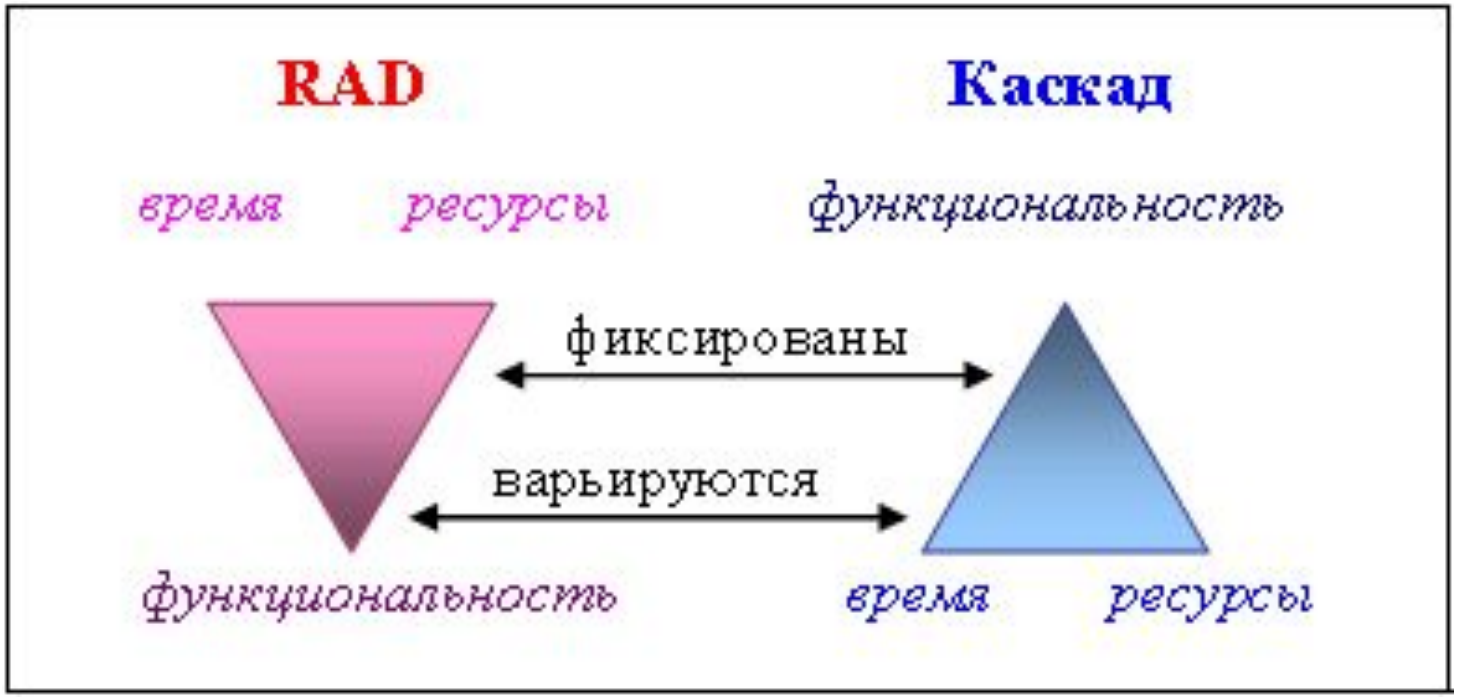
Когда применяется RAD

- проект большой, но поддается разделению на более мелкие функциональные компоненты
- ПО не обладает большой вычислительной сложностью

RAD не применяется

- В проектах, где требования к программному продукту четко определены и не должны меняться, и, следовательно, вовлечение заказчика в процесс разработки не требуется
- В проектах, сложность которых определяется необходимостью реализации сложных алгоритмов, а роль и объем пользовательского интерфейса невелик

Сравнение двух моделей



Характеристика модели

- Основным достоинством модели является уменьшение сроков разработки
- Ее главный недостаток заключается в необходимости использования большого числа квалифицированных разработчиков, что может существенно повысить стоимость разработки

Конец лекции