

# Технология программирования

Курс лекций для групп 3057, 4057

## Лекция №2

Лектор: проф. И.В. Штурц

2010 г.

# Содержание

1. Введение
2. Модели жизненного цикла ПП
3. Модели команды разработчиков
4. Управление проектами
5. Словесная коммуникация
6. Структуризация ПП: модульное и компонентное программирование
7. Языки и методы проектирования
8. Тестирование и верификация ПП
9. CASE-системы
10. Надежность ПП
11. Стандарты качества технологии программирования

# Фазы жизни ПП

	Ср. длительность
Разработка (Development):	0,5 - 2 года
– Проектирование (Design)	
– Реализация (Implementation)	
Эксплуатация и сопровождение (Maintenance)	1 - 10 лет
Сопровождение:	
• Обучение пользователей и консультации	
• Исправление ошибок, выпуск «заплаток» (patches)	
• Адаптация к изменениям требований	
• Для долгоживущих продуктов – <i>продолжающаяся разработка</i> с выпуском новых версий каждые несколько лет	

# Виды деятельности при разработке ПП

Проектирование	Выявление и анализ требований	Requirements elicitation and analysis
	Планирование	Planning
	Проектирование	Design
Реализация	Кодирование	Coding
	<i>Верификация*)</i>	Verification
	<i>Аттестация **)</i>	Validation } V&V
	Модульное тестирование	Unit testing
	Комплексное тестирование	System testing
	Написание документации	Documenting
	Внедрение	Installation or Deployment

\*) Верификация – «Правильно ли идет разработка?»  
«внутренняя» проверка промежуточных результатов на соответствие ТЗ, а процесса – на соответствие стандартам

\*\*) Аттестация – «Правильный ли продукт разрабатывается?»  
«внешняя» проверка результатов на соответствие ТЗ и ожиданиям пользователей

Способы V&V: тестирование, инспекция кода, моделирование, ...

# Понятие жизненного цикла ПП

Software life cycle, aka Software development process

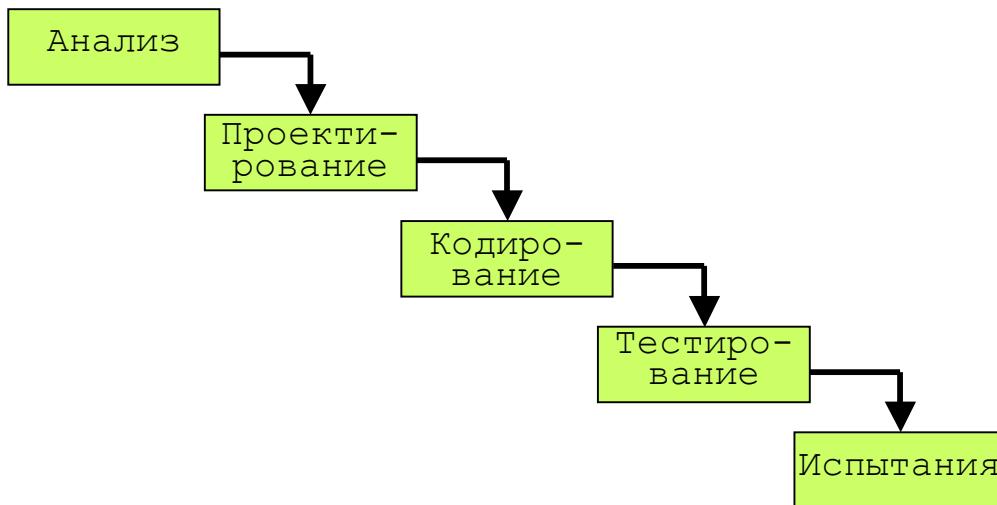
Модель жизненного цикла ПП - абстрактное описание последовательности действий (этапов) и их взаимосвязи при разработке ПП

Основные типы моделей ЖЦ:

- Хаотичная, неупорядоченная
- Водопадная (waterfall), или каскадная
- Итерационная
  - Пошаговая
  - Спиральная
    - ✓ Быстрое прототипирование
    - ✓ Инкрементальная
- Эволюционная

# Водопадная модель ЖЦ (1970)

- линейная последовательность неперекрывающихся этапов (стадий):



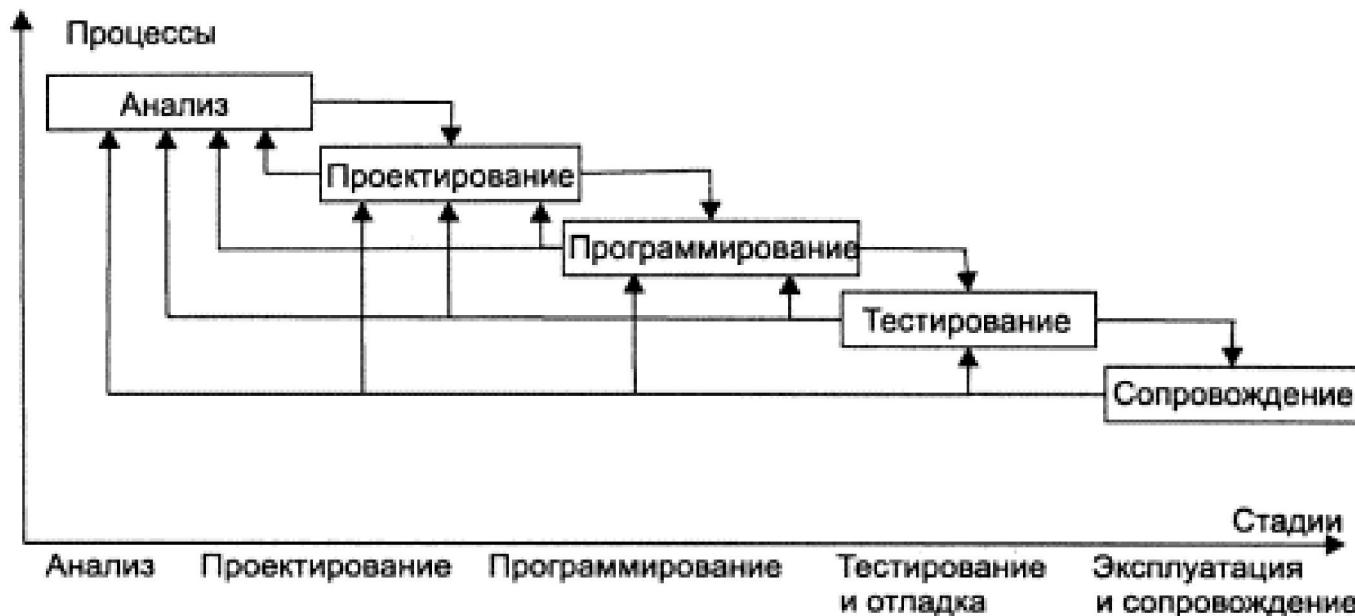
Дискретный во времени, преимущественно односторонний процесс:

- каждый этап завершается выпуском контролируемого промежуточного результата – *артефактов* (документов, кода, данных, ...)
- новый этап не начинается, пока полностью не завершится предыдущий
- результаты пройденных этапов замораживаются (фиксируются), и возврат к ним не приветствуется

# Этапы ЖЦ: результаты и трудоемкость

Этап	Артефакты	Доля в общей трудоемкости (в среднем)
<i>Проектирование</i>		30%
Анализ требований	Требования (тех. задание)	5%
Общее проектирование	Внешняя спецификация; архитектура	10%
Детальное проектирование	Внутренние (проектные) спецификации	15%
<i>Реализация</i>		70%
Кодирование	Исходный код	15%
Автономное тестирование	Журналы ошибок;	20%
Комплексное тестирование	исполняемый код	35%

# Итерации в водопадной модели

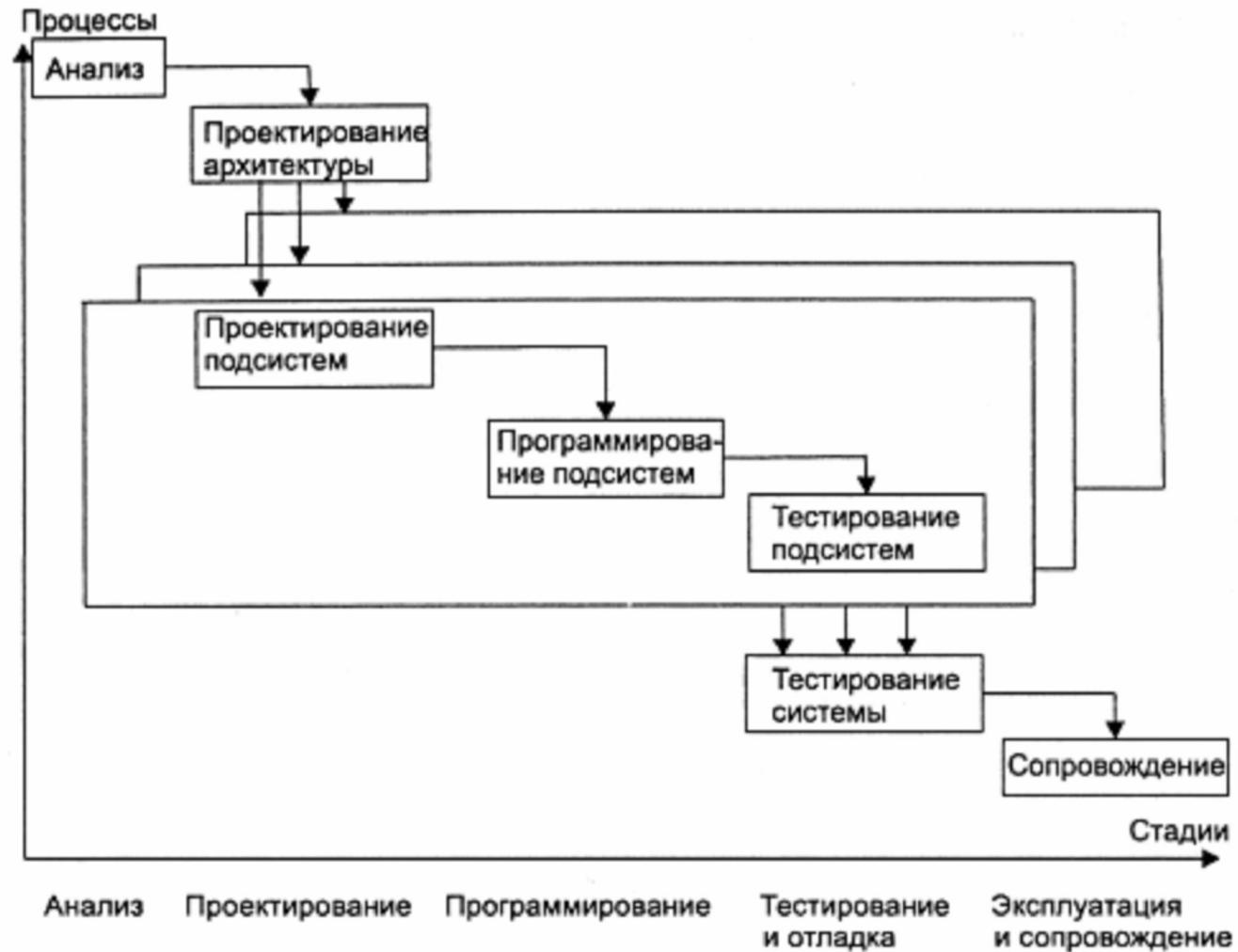


Итерация процесса - возврат и повторение пройденных этапов из-за:

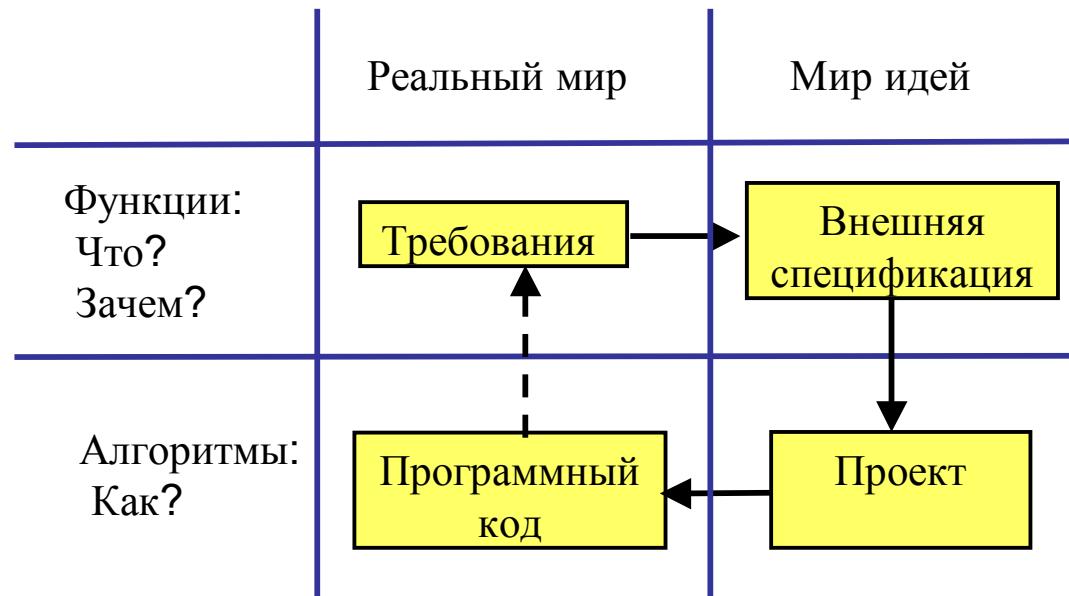
- ✓ ранее допущенной ошибки, обнаруженной при верификации/аттестации
- ✓ изменений и дополнений в требованиях
- ✓ выявленных ограничений нижнего уровня реализации

Итераций здесь стараются избегать, т.к. они вносят хаос и удлиняют процесс, причем непредсказуемым образом

# Вариант водопадной модели с параллельными подпроцессами разработки подсистем

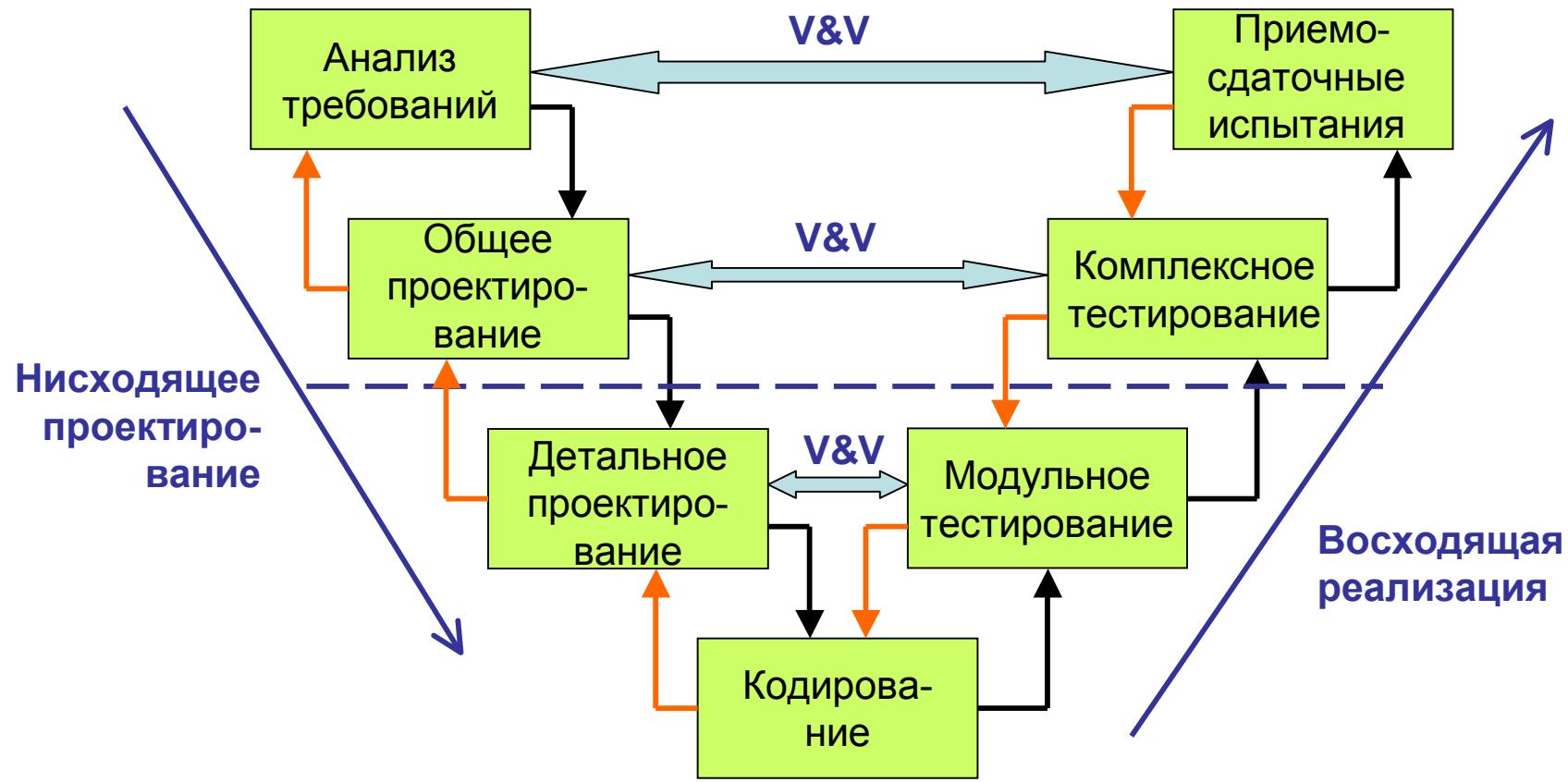


# ЖЦ: от реальности к идеям и обратно



Типичный цикл инженерной деятельности при создании  
любого продукта

# V-образная водопадная модель ЖЦ



Чем раньше допущена ошибка, тем позже ее можно обнаружить путем проверки ↔ и тем дороже ее исправление (далее откат)

# Достоинства водопадной модели

в 1970-90-х – стандарт министерства обороны США

- Упорядоченный, предсказуемый **дискретный** процесс → хорошо планируется и контролируется
  - дискретность этапов: точки промежуточного контроля и решений
  - удобно для менеджеров: всегда известен % выполнения плана
  - удобно для заказчика: контракт с фиксированной стоимостью работ
- Уменьшается вероятность ошибок, вызванных перекрытием этапов при нисходящем проектировании
  - нельзя начинать кодирование, пока не закончено проектирование! (Отсутствие перекрытий этапов характерно для отраслей материального производства, напр. строительной)

Цель хорошей технологии ТП - предупреждение ошибок и возможно более раннее их выявление

Этому служит тщательное проектирование с верификацией/аттестацией каждого этапа

# Недостатки водопадной модели

- Отрицательное отношение к итерациям проекта
  - и требования, и возможности часто меняются в ходе разработки
- Слишком большая длительность периода от начала процесса до первого отчуждаемого результата (до испытаний ПП)
  - заказчики/пользователи могут увидеть первые результаты только по завершении проекта, и радикально изменить требования
- Стратегия разработки «сверху-вниз» (от общего к частному) не всегда удобна
  - некоторые продукты – библиотеки удобнее «выращивать» снизу-вверх
- Требование не переходить к очередному этапу до полного завершения предыдущего – слишком жесткое
  - требуется жесткая синхронизация активности разработчиков
  - при проектировании бывает нужно закодировать ключевые функции для проверки достижимости нужных результатов

Водопадная модель хорошо подходит только для «легких» проектов, в которых требования полностью формулируются с самого начала и не изменяются в ходе разработки:

- чисто математические задачи
- встроенные системы управления, особенно оборонного назначения

# Как преодолеть недостатки водопадной модели?

- ✓ Принять неизбежность итераций проекта: разрабатывать ПП не сразу, а в несколько заходов
  - На каждом заходе выполнять полный цикл проектирования и реализации
  - Постепенно расширять функциональность, детализировать проект и уточнять требования
- ✓ Как можно раньше демонстрировать работу незавершенного ПП (т.е., прототипа или версии) пользователям

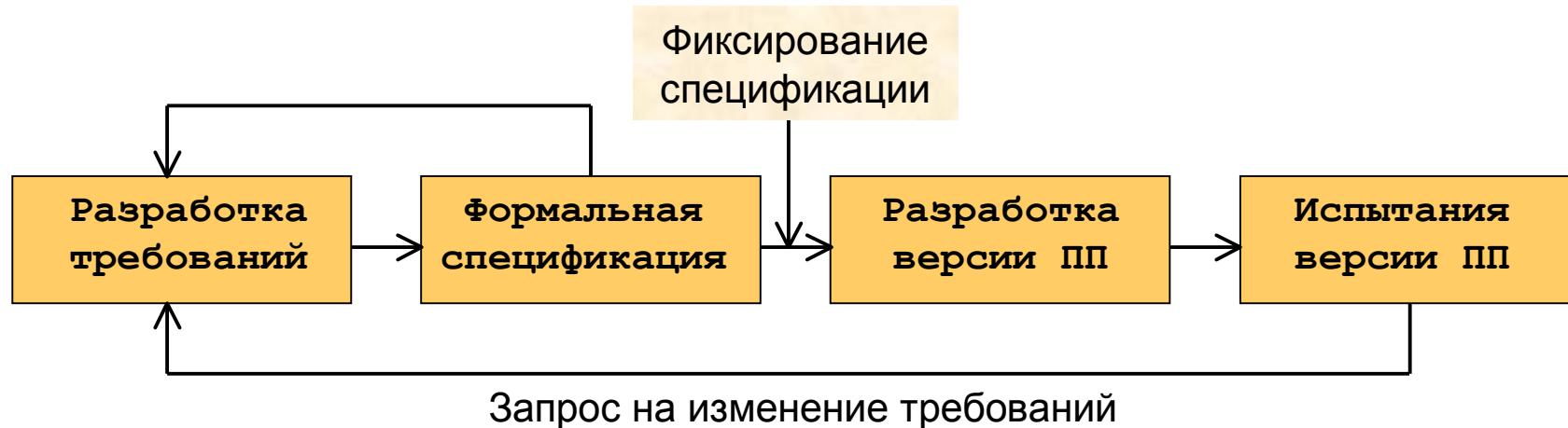
Приходим к идее итерационной модели: цикл проектирование - реализация выполняется несколько раз

(Это цикл на слайде 10)

Два близких друг к другу варианта:

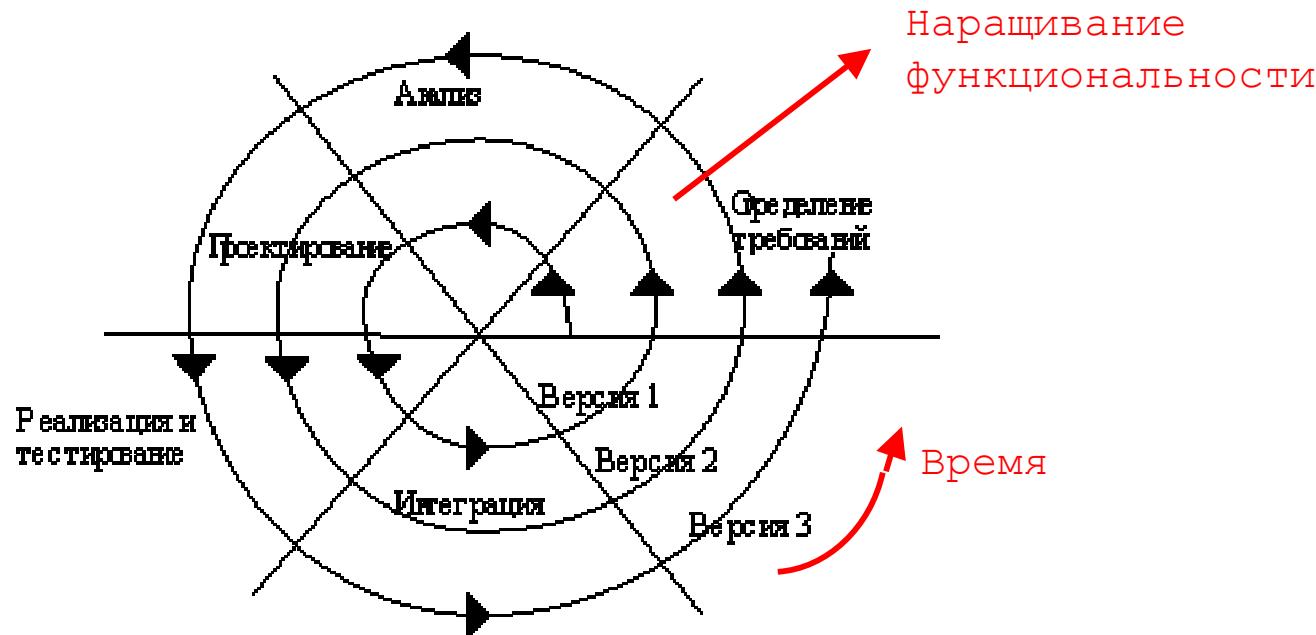
- Модель пошаговой разработки
- Спиральная модель

# Модель пошаговой разработки



- Выполнение каждого цикла итераций – по водопадной модели
- Число повторений цикла определяется заказчиком/  
пользователями; это возможно в проектах с открытым бюджетом
- Частный случай этой модели – в методе «чистая комната»  
(Cleanroom, IBM)

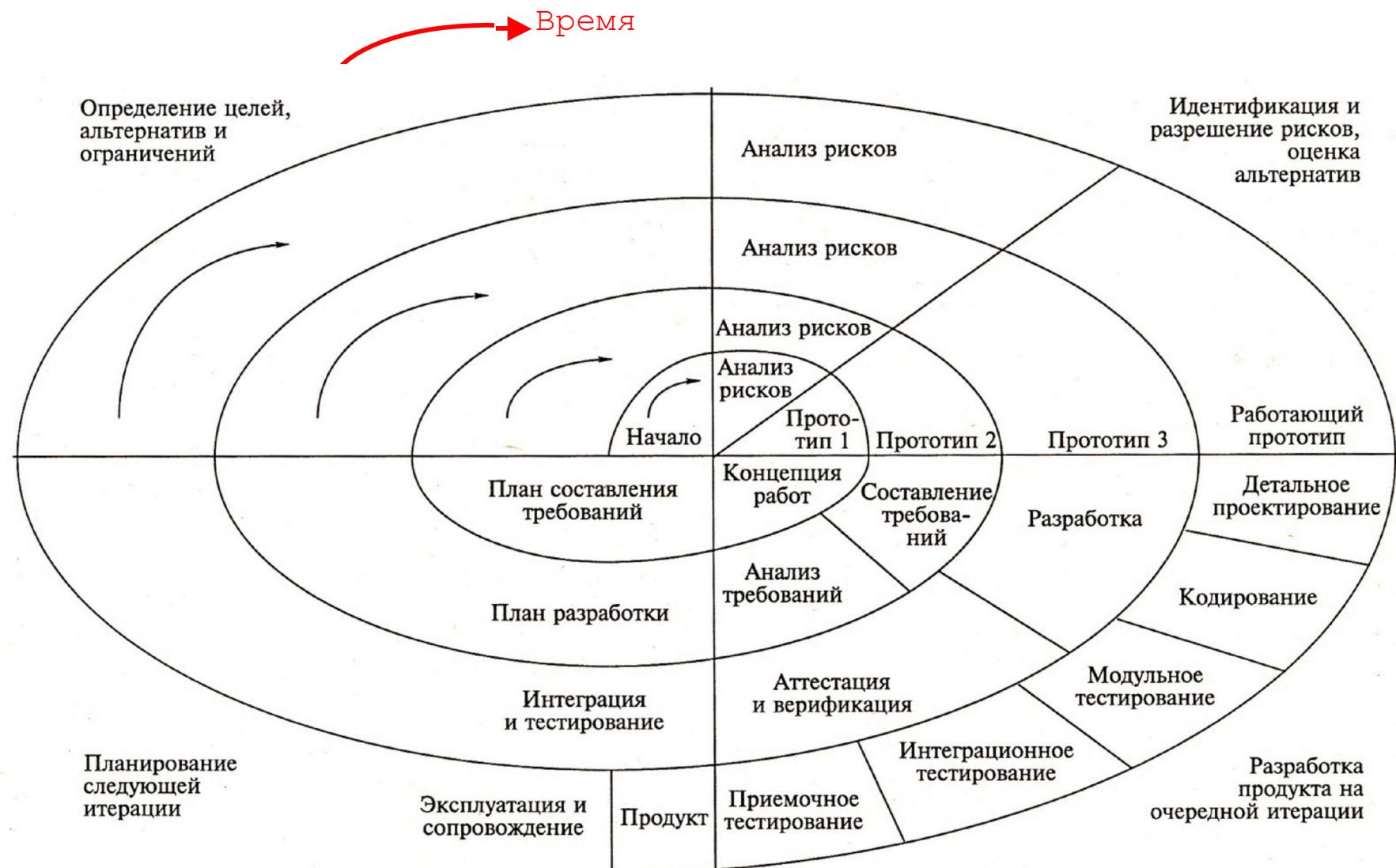
# Идея спиральной модели ЖЦ (1988)



Весь процесс, от начального эскиза ПП до его конечной реализации, развивается по спирали с постепенным наращиванием функциональности

На каждом витке спирали – этапы водопадной модели

# Сpirальная модель ЖЦ



# Варианты спиральной модели

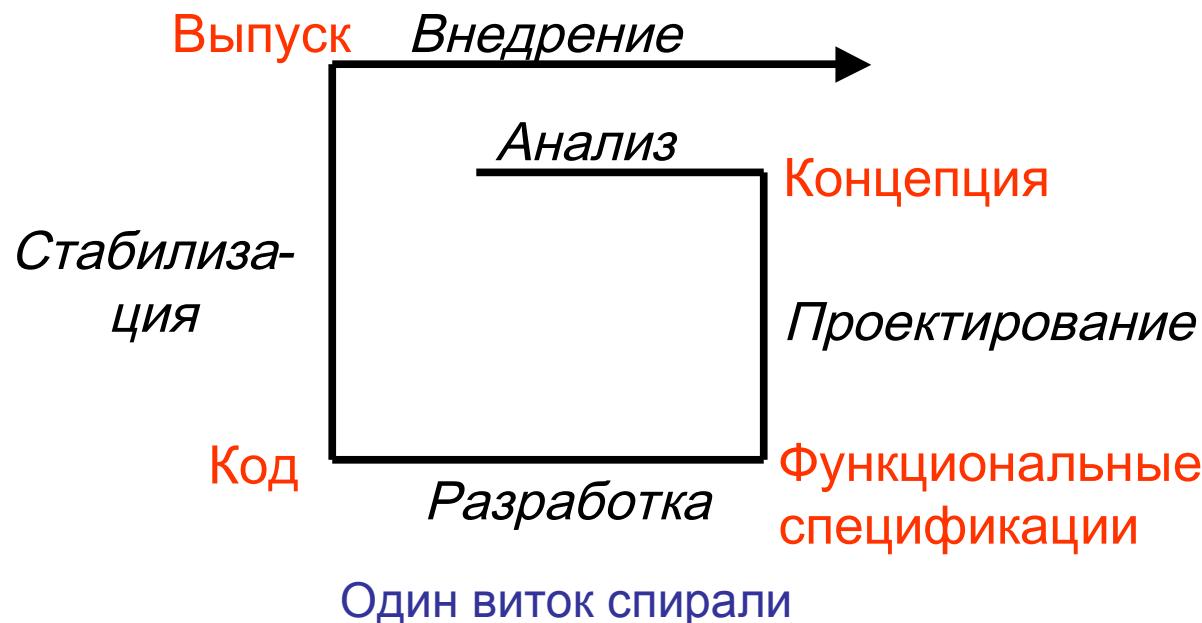
## Быстрое прототипирование

- На каждом витке спирали создается очередной *прототип* - макет, упрощенная версия будущего продукта
- Первый прототип может создаваться для:
  - проверки ключевых концепций, алгоритмов, ...
  - подтверждения *осуществимости (feasibility)* проекта
  - демонстрации интерфейса пользователя
  - оценки требуемых ресурсов и рисков
- Обычно длительность одного витка – несколько месяцев

## Наращиваемая (или инкрементальная) разработка

- На каждом витке спирали (начиная с некоторого) создается *инкремент* - полностью работоспособная версия, поступающая в эксплуатацию
- Удобно для больших систем и для продолжающейся разработки
- Примеры
  - ПО бортовой системы управления Shuttle: 7 инкрементов за 10 лет (IBM)
  - Системы ERP так создаются и внедряются годами

# Модель ЖЦ в технологии MSF (1998) (Microsoft Solutions Framework )



Терминология:

*Разработка* = реализация

*Стабилизация* = тестирование

*Выпуск (release)* – ПП с документацией пользователей

*Внедрение (deployment)* – испытания у заказчика и передача продукта группе сопровождения

# Детали модели ЖЦ MSF

- Каждый *этап (phase)* завершается *вехой (milestone)* - появлением и фиксацией некоторого отчуждаемого артефакта
- Этап анализа завершается выпуском *концепции* проекта - набора документов; основной из них - *внешняя спецификация*
- Проектные (внутренние) спецификации:
  - проектные документы, модели, прототипы
  - программа и методика испытаний
  - календарный план выполнения проекта
  - план управления рисками
- Число витков спирали различно для различных продуктов; оно планируется заранее и корректируется по промежуточным результатам
- Длительность одного витка не более 6 месяцев
- При планировании определяются промежуточные вехи – точки синхронизации активностей; на этапах разработки и стабилизации это промежуточные сборки (builds) и версии ПП

# Версия модели ЖЦ MSF 2002 г.



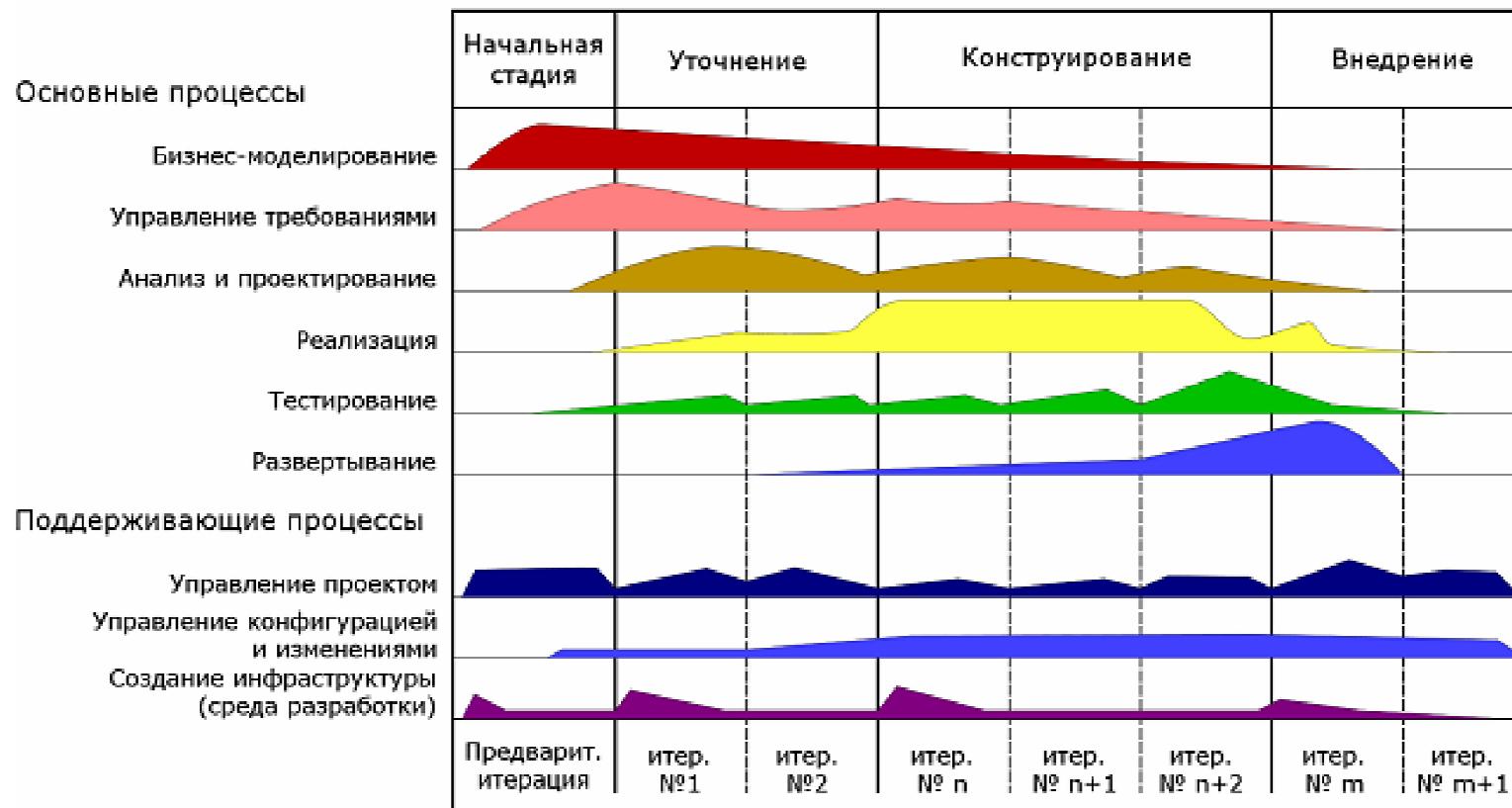
Этап анализа здесь назван *предвидением (envisioning)*, этап проектирования - планированием

*Охват проекта (project scope)* определяет объем и содержание работ  
(В некоторых организациях охват проекта называется *Описанием работы - Statement of work, SOW* или *Техническим заданием*)

# Модель ЖЦ RUP (Rational Unified Process)

## Рабочие процессы

## Стадии



## Итерации

Полный жизненный цикл разработки продукта состоит из четырех стадий, каждая из которых включает в себя один или несколько этапов (итераций) по 2-6 недель; на каждом выпускается версия для внутреннего или внешнего использования

# Стадии ЖЦ RUP

## 1. Начало (Inception)

- Формируются видение и охват (scope) проекта
- Создается экономическое обоснование (business case)
- Определяются основные требования, ограничения и ключевая функциональность продукта
- Создается базовая версия модели прецедентов и оцениваются риски

## 2. Проектирование, или уточнение (Elaboration)

- Анализ предметной области и построение исполняемой архитектуры (структуры верхнего уровня)
- Документирование требований
- Спроектированная и оттестированная архитектура
- Обновленное экономическое обоснование и уточненные оценки сроков и стоимости; сниженные основные риски

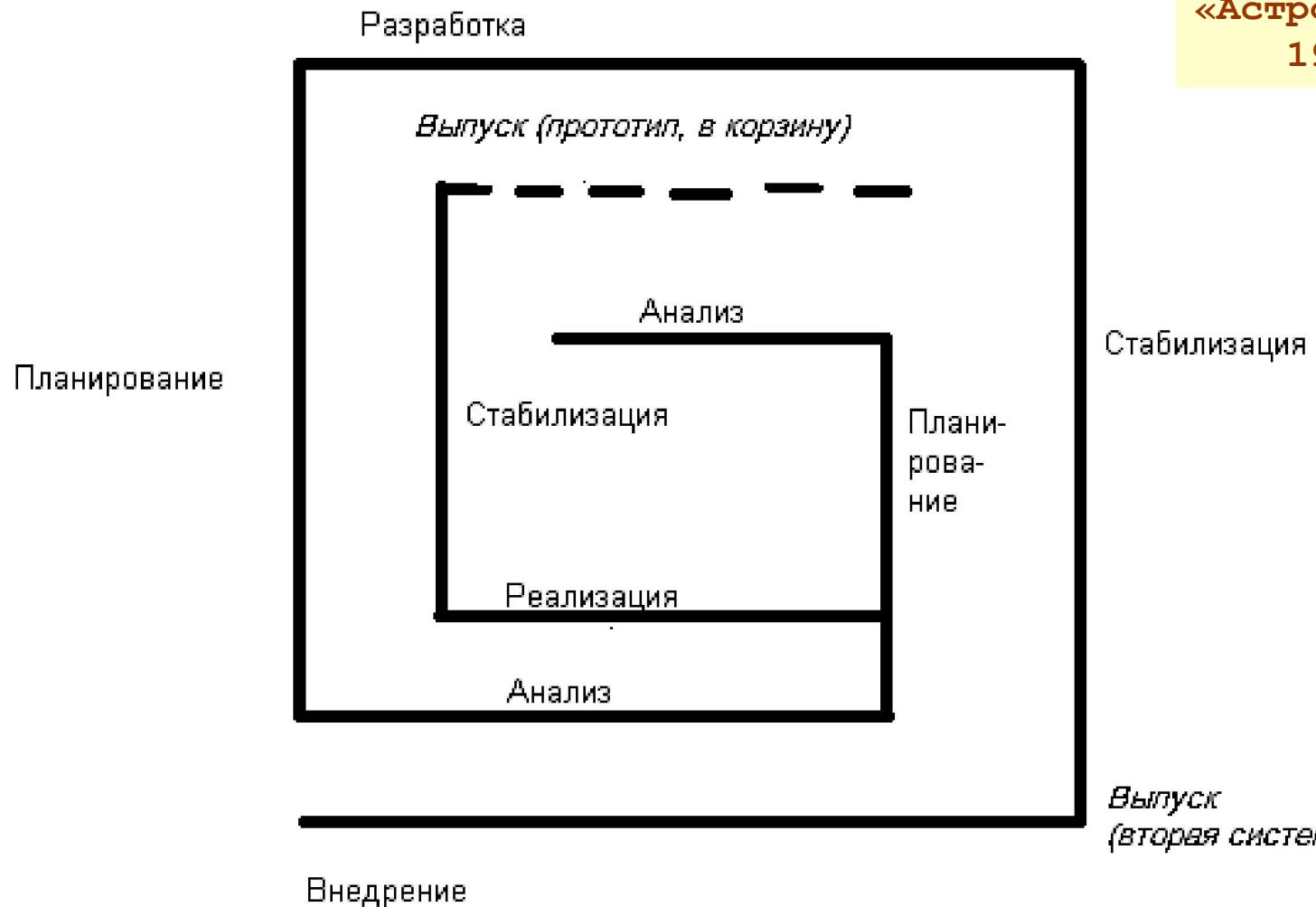
## 3. Построение, или конструирование (Construction)

- Реализация большей части функциональности продукта (2-4 итерации)

## 4. Внедрение (Transition)

- Создается финальная версия продукта и передается заказчику
- Если качество не соответствует ожиданиям пользователей или критериям, установленным на стадии Начало, стадия Внедрение повторяется снова

# Еще одна модель ЖЦ



Фирма  
«Астрософт»,  
1998

# Идея гибких методологий

Рассмотренные технологии называются «тяжеловесными»,  
планируемыми (plan-driven), или жесткими

Их общие недостатки:

- трудность адаптации к изменениям и уточнениям требований в ходе разработки (в течение одного витка спирали)
- жесткая дисциплина (неперекрывающиеся этапы, подробная документация и т.п.) - удобна для менеджеров, но не для рядовых разработчиков

Как реакция на эти недостатки, в 1990-е появились **гибкие (agile)**,  
**или облегченные (lightweight)** методологии - под лозунгами:

- индивидуумы и их взаимодействие важнее процессов и инструментов
- работоспособное программное обеспечение важнее обширной документации
- сотрудничество с заказчиком важнее слепого следования бумажному контракту
- готовность к изменениям важнее следования плану

Agile = ловкий, шустрой, проворный, расторопный

# Peopleware как главный ресурс

Гибкие методологии ориентированы на человека,  
а не на процесс

- ✓ Работа по созданию программных продуктов должна приносить удовольствие
- ✓ Человек - существо, которому необходимо личное, устное общение
- ✓ Людям нравится отсутствие точных предписаний относительно их собственного поведения
- ✓ Человек больше любит изобретать, а не находить готовые решения
- ✓ Человек хорошо работает, опираясь на примеры
- ✓ Он делает ошибки и с трудом меняет привычки
- ✓ Он обучается с помощью наблюдения и практики
- ✓ Личный стиль руководителя имеет огромное значение



# Общие принципы гибких методов

Сформулированы в Agile Manifesto (2001)

- Отказ от жесткого долгосрочного планирования (трудно выполнимого в реальной жизни)
- Постоянная и быстрая поставка полезного продукта заказчику
  - Работающие версии поставляются часто (с периодом в недели, а не месяцы)
  - Работающие версии – основная мера прогресса проекта
- Ежедневное тесное сотрудничество разработчиков с заказчиком
- Приветствуются даже поздние изменения требований
- Личная беседа – это лучшая форма коммуникации
- Простота структуры программы
- Постоянное внимание техническому совершенству
- Самоорганизующиеся команды
- ПП создается мотивированными личностями, доверяющими друг другу
- Регулярная адаптация к изменяющимся обстоятельствам: требованиям, инструментам и пр.

# Экстремальное программирование (XP)

## (1995)

Методология нацелена на «экстремальные значения» очевидных принципов: простота, быстрые короткие итерации, тщательное тестирование и т.п.

1. *Игра в планирование (Planning Game)* – постоянный диалог между заказчиками и разработчиками с целью определить текущие задачи, приоритеты и сроки реализации
2. *Частые выпуски версий (Small Releases)* – каждые 2 недели
3. *Метафора системы (System Metaphor)* – простая аналогия, интуитивно понятная всем членам команды
  - описывает внутреннее строение и работу продукта
  - заменяет проектный документ «Общая структура»
4. *Простой дизайн* – простота структуры проектируемого ПП
5. *Непрерывное тестирование* – в любой момент можно убедиться в корректности работы системы
  - тесты для модулей должны выполняться безукоризненно
  - заказчики пишут тесты для системы
  - «тестируй, а затем кодируй»: тесты разрабатываются параллельно анализу требований и проектированию

# Практики XP (2)

6. *Постоянная интеграция* – сборки по несколько раз в день
7. *Реорганизация (Refactoring)* – постоянное улучшение кода путем его переписывания (упрощение, устранение дублирования частей и пр.)
8. *Коллективное владение кодом* – любой член команды имеет право вносить изменения в любую часть кода
9. *Парное программирование (Pair Programming)* – код разрабатывается одновременно двумя программистами за **одним** компьютером; пары постоянно меняются
  - время разработки сокращается на 40-50%
  - качество кода улучшается
10. *Представитель заказчика* – член команды
11. *40-часовая рабочая неделя*
12. *Строгие стандарты кодирования*

# Еще две гибкие методологии

## Scrum («потасовка» - термин из игры в регби)

- Процесс не обязан быть линейным – от анализа требований до тестирования; этапы могут следовать в любом порядке, меняющемся по ходу дела
  - контролируемый возврат к хаотичной модели ЖЦ
- Длительность одного витка – 1 месяц
- Ежедневные 15-мин. совещания всей командой
  - обсуждается, чем все будут заниматься в течение следующего дня
  - руководителям компании дают знать о состоянии разработки

## FDD (Feature-Driven Development – разработка, продвигаемая функциональностями)

- Любая функциональность разбивается на более мелкие - features - те, чья разработка укладывается в двухнедельный срок
- Индивидуальное владение частями (классами) кода и персональная ответственность за них
- Feature Teams – небольшие динамически формируемые команды
- Постоянная инспекция кода
- Видимый прогресс - благодаря частым отчетам о состоянии проекта

# Резюме по гибким методологиям

Гибкие методы не подходят в случае:

- Критически важного (mission-critical) ПП
- Большой команды проекта (>20 чел.)
- Распределенной команды

Хорошо подходят, если:

- Требования к ПП плохо определены и/или изменяются в ходе проекта
- Проект небольшой - длительностью до 1 года
- Команда состоит из профессионалов высокого уровня
  - им достаточно исходного кода как единственной документации
  - совместная ответственность за ошибки (из-за коллективного владения кода)

В последнее время многие организации включили элементы гибких методов в свои технологии, в том числе, в MSF

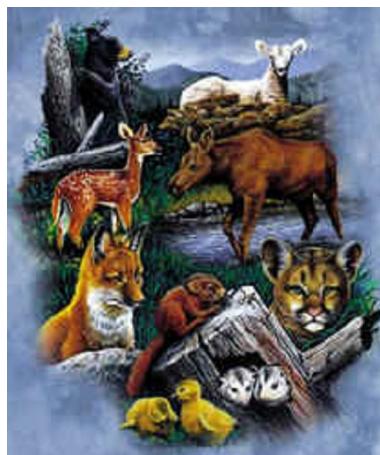
# Эволюционная модель ЖЦ

- применима для больших долгоживущих систем  
Примеры: Linux, Google, Web-порталы и социальные сети

Условия среды и требования изменяются многократно и непредсказуемым образом во время жизни системы

Система создается аналогично живым природным системам или городам:

- развивается постепенно, начиная с небольшого ядра
- создаются и используются инструменты для (само)развития
- Разработчики распределены в пространстве и времени



15 фев 2010



32

# Заключение

- Две полярных модели ЖЦ ПП:
  - водопадная – для хорошо определенных, не изменяемых требований; для систем критического назначения
  - модель ЖЦ в гибких методологиях – для небольших, плохо определенных ПП, требования к которым изменяются в ходе разработки
- Между этими полюсами располагаются различные варианты итерационной, чаще всего спиральной модели - наиболее распространенная практика
- Общая тенденция последних лет – включение элементов гибких методологий в существующие корпоративные технологии

# Дополнительные вопросы

1. Чем плоха хаотичная модель ЖЦ?
2. Приведите примеры непрерывных и дискретных технологических процессов в различных отраслях промышленности. В чем их достоинства и недостатки с точки зрения эффективности процесса и его управляемости ?
3. Почему проектирование упрощается при спиральной модели ?
4. Какова должна быть минимальная функциональность у первого прототипа корпоративной информационной системы? Для программы численного моделирования физического процесса? Для графической библиотеки?
5. Что такое альфа и бета версии ПП?
6. Есть ли принципиальные различия между двумя моделями жизненного цикла: MSF и RUP? Если есть, то какие?
7. В чем идея ЖЦ фирмы «Астрософт»?
8. Какие из практик экстремального программирования Вы готовы выполнять, а какие Вам представляются трудными для выполнения?