Программная инженерия. Организация процесса разработки

Основные понятия программной инженерии

Терминология

Инженерия обеспечивает решение поставленных задач посредством существующих теорий и методов.

Инженер начинает с постановки задачи и поиска инструментов для наилучшего решения задачи в рамках существующих организационных, финансовых и временных ограничений.

Программная инженерия делает значительный упор на методы и подходы, а не на инструменты.

Основная предпосылка: необходимость *перехода* от программирования *как искусства* к программированию *как индустрии* в связи с усложнением программного обеспечения.

В СССР основы программной инженерии (технологии программирования) были заложены академиком В. М. Глушковым в Институте кибернетики НАН УССР.

Терминология

Термин был предложен в 1968 г. в качестве темы конференции НАТО, посвященной вопросам максимальной загрузки самых мощных компьютеров такого времени:

«Основная причина кризиса программного обеспечения резкий рост мощностей вычислительных машин! Проще говоря: нет вычислительной техники нет проблем с разработкой программного обеспечения для неё; когда же появилось несколько слабых компьютеров, появились первые проблемы, связанные с разработкой программного обеспечения, сейчас у нас есть гигантские компьютеры, и программирование стало столь же гигантской проблемой»

– Э. Дейкстра

Программная инженерия (ПИ)

Программная инженерия (англ. software engineering) - система методов, средств и дисциплин планирования, разработки, эксплуатации и сопровождения программного обеспечения, готового к внедрению;

Программная инженерия - раздел компьютерных наук (англ. computer sciences), изучающий методы и средства построения компьютерных программ как продукта теоретической и инженерной деятельности разработчиков или их коллективов;

Программная инженерия – это инженерная дисциплина, отражающая все грани разработки программного обеспечения.

Первые 20 лет (70-80-е годы) в ПИ доминировали классический и процедурный подходы, а в следующие 20 лет (1990-2000-е годы) – объектно-ориентированных подход

Теоретический фундамент ПИ

Программная инженерия основывается на математических дисциплинах:

- *теория алгоритмов* - нормальные алгоритмы, вычислимые функции, машина Тьюринга, граф-схемы, модели алгоритмов;
- *математическая логика* - формальный вывод утверждений;
- *теория управления* - принципы, методы и общие законы планирования и управления в сложных системах;
- *теория доказательств* - математическая теория вывода по аксиомам и утверждениям, теория верификации программ;
- *теория множеств* - формальное представление совокупностей объектов из предметной области.

Компоненты ПИ

Методы	• обеспечивают решение широкого спектра технических задач
Средства	• обеспечивают автоматизированную или автоматическую поддержку методов (CASE-системы)
Процессы	• это наборы взаимосвязанных работ, которые преобразуют исходные данные в выходные результаты

Компоненты ПИ

Методы

- Планирование и оценка программного проекта;
- Анализ требований к компьютерной системе в целом и программному обеспечению в частности;
- Проектирование структур программ, входящих в состав ПО;
- Конструирование программного теста;
- Тестирование;
- Сопровождение ПО, уже используемого заказчиком.

Процессы

- Порядок применения методов и утилит;
- Формирование отчетов, форм по соответствующим требованиям;
- Контроль, который помогает обеспечивать качество и координировать изменения;
- Формирование «вех», по которым руководители оценивают прогресс.

Классификация процессов ПИ

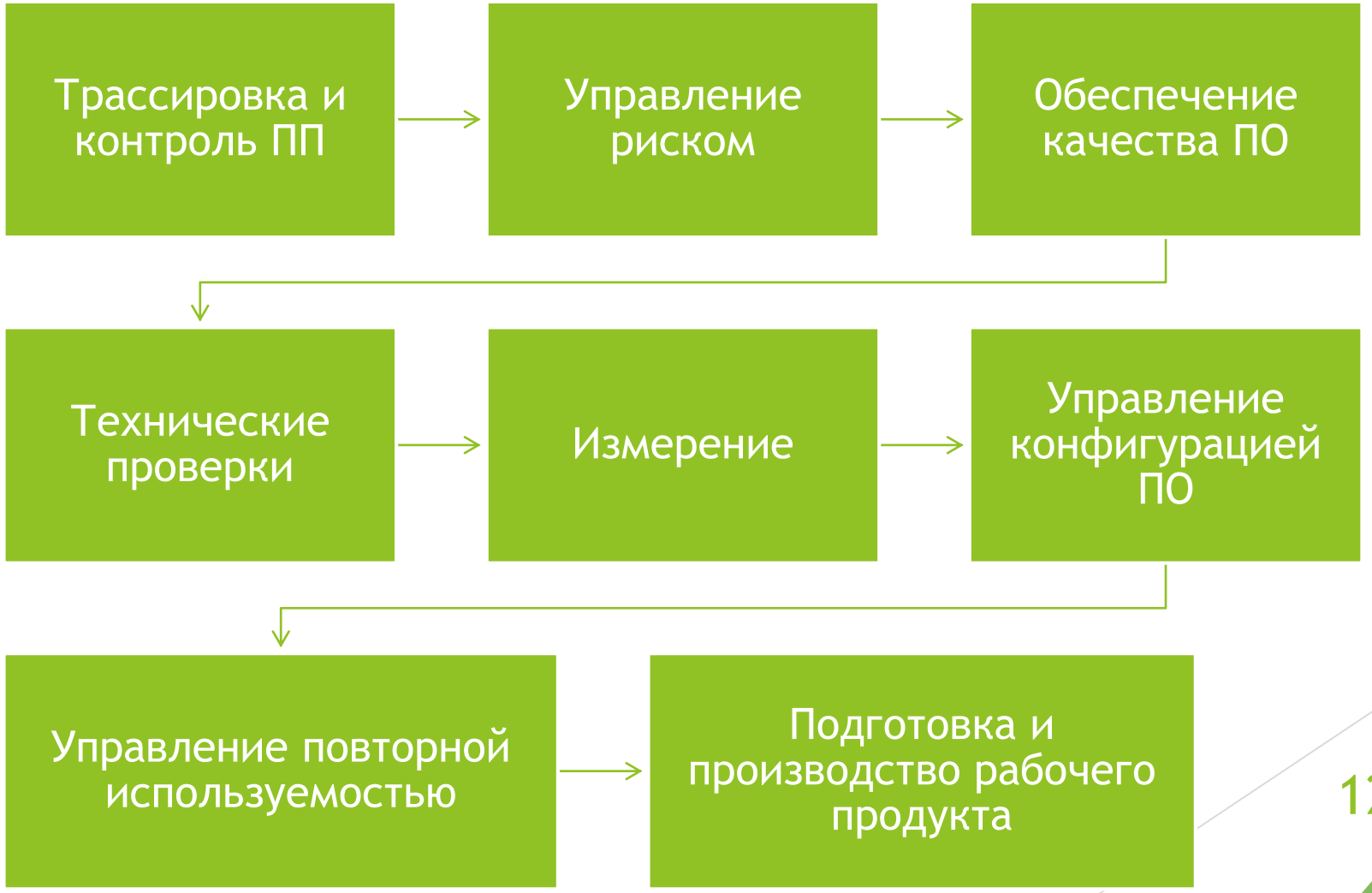
- **Процессы соглашения** состоят их 2 подпроцессов: приобретения и поставки;
- **Процессы организационного обеспечения проекта** состоят из 5 подпроцессов: процесса менеджмента модели жизненного цикла, процесса менеджмента инфраструктуры, процесса менеджмента портфеля проектов, процесса менеджменты людских ресурсов, процесса менеджмента качества;
- **Процессы проекта** используются для создания и совершенствования планов проекта, оценки фактического выполнения и продвижения относительно плановых заданий, а также управления выполнением проекта вплоть до полного его завершения;
- **Технические процессы** обеспечивают определение требований к системами, преобразование требований в полезный продукт, применение продукта и изъятие продукта из обращения. Они состоят из 11 подпроцессов.

Базис процессов разработки ПО

Деятельность	• Самый крупный элемент, ориентированный на достижение весомой цели и применяется независимо от прикладной области, размера проекта, сложности затрат и степени строгости использования «арсенала» ПИ.
Действие	• Средний элемент. Охватывает набор задач, которые производят этапный рабочий продукт.
Задача	• Самый мелкий элемент. Задача фокусируется на мелкой, но хорошо определенной цели, которая приводит к ощутимому реальному результату.

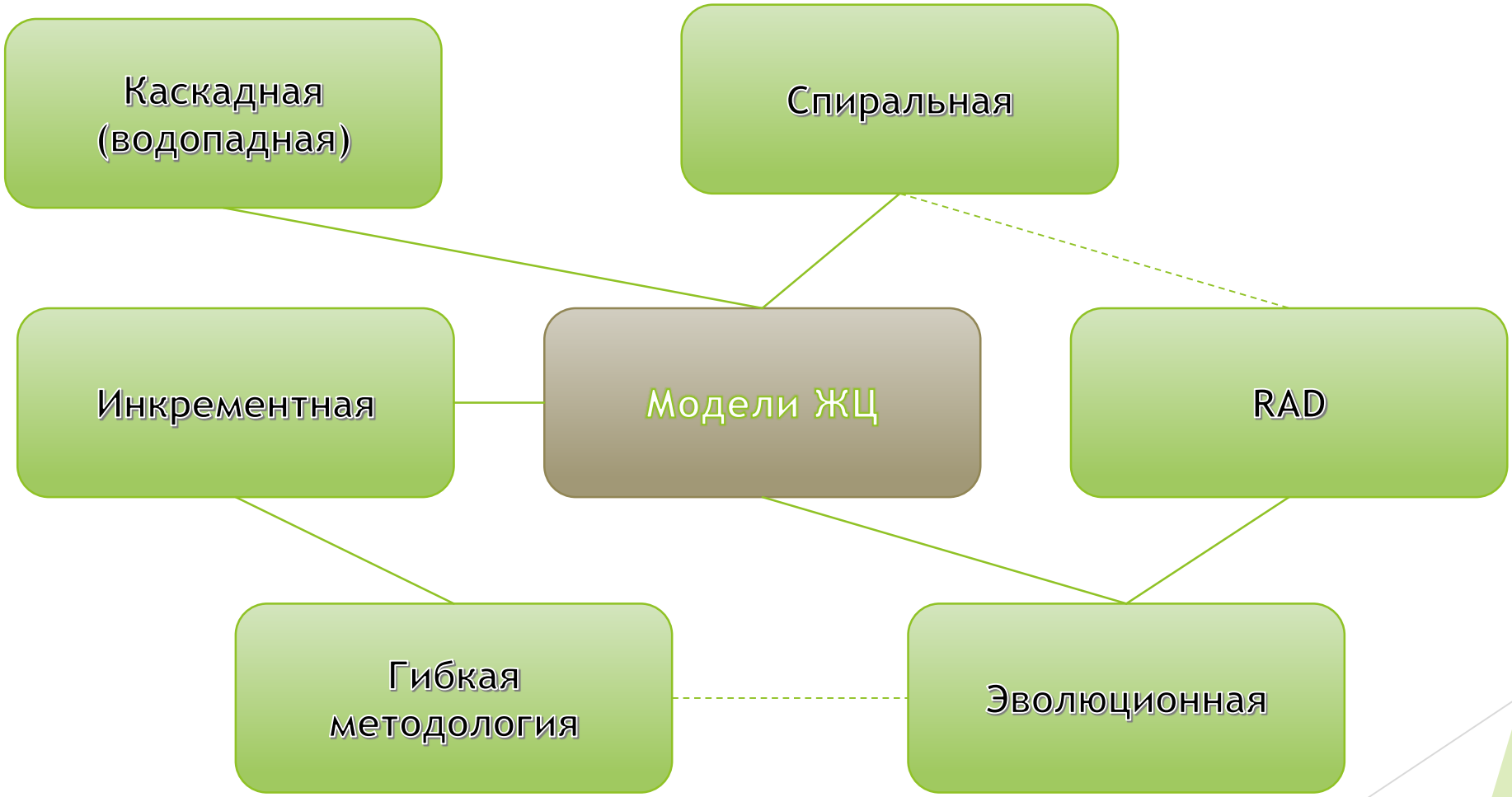
Основные виды деятельности

Основные виды защитной деятельности



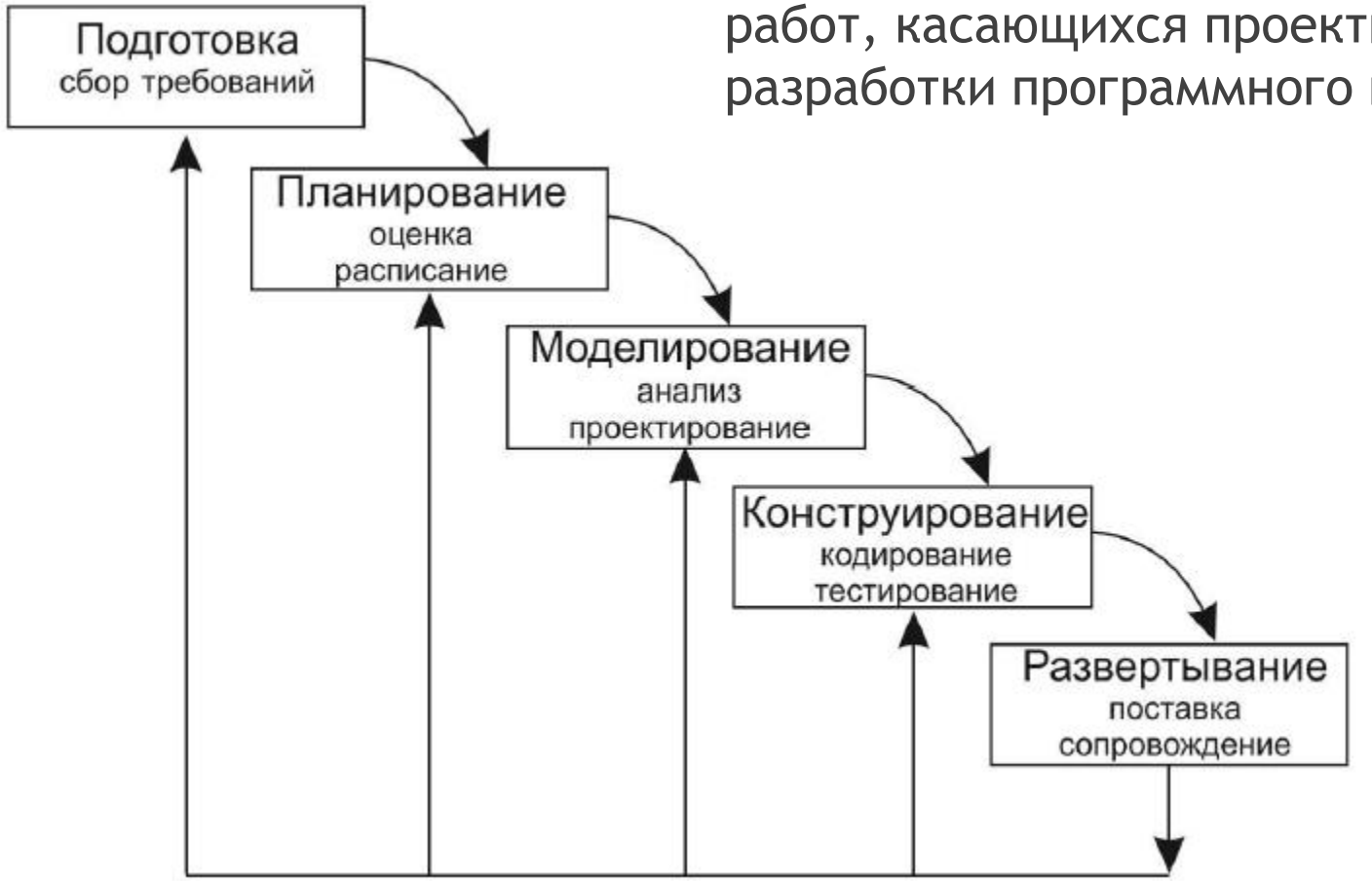
Модели процесса разработки ПО

Классификация моделей ЖЦ



Модель «классический жизненный цикл» (каскадная модель)

- **Жизненный цикл** - схема упорядочивания работ, касающихся проектирования и разработки программного продукта.



Старейшая модель процесса разработки ПО (автор Уинстон Ройс, 1970)

Модель «классический жизненный цикл»

Достоинства

- План и временной график по все этапам проекта;
- Упорядочивает ход разработки.

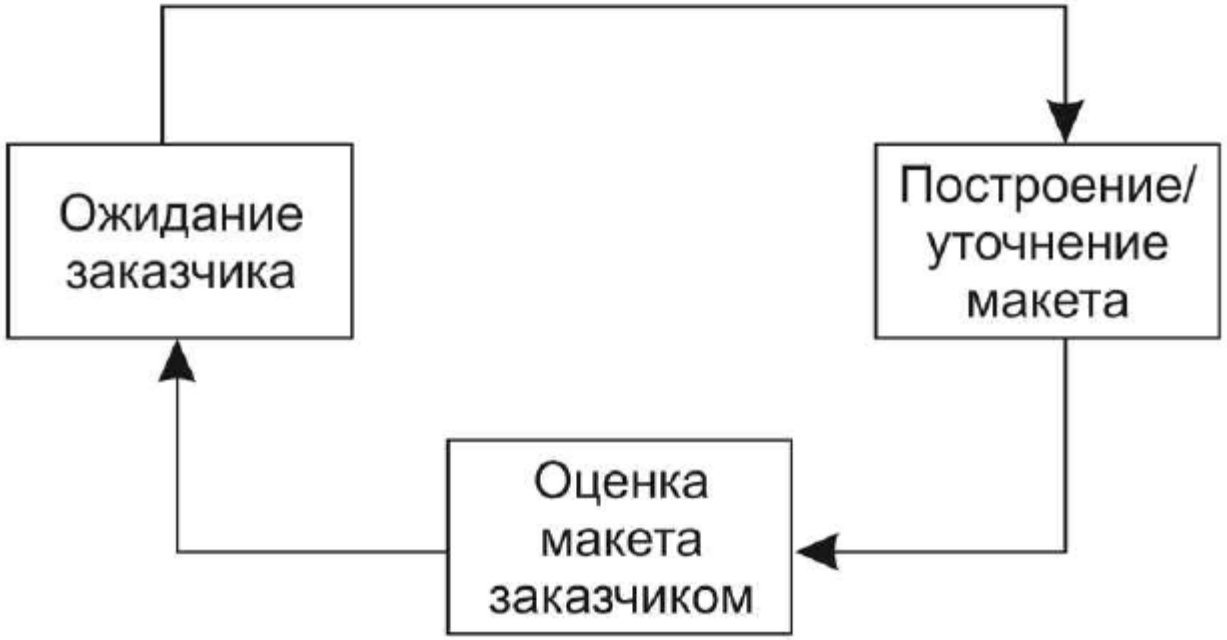
Недостатки

- Реальные проекты часто требуют отклонения от стандартной последовательности;
- Цикл основан на точной формулировке исходных требований к ПО;
- Результаты проекта доступны заказчику только в конце работы.

Макетирование

Модель может принимать одну из трех форм:

- бумажный макет или макет на основе ПК;
- работающий макет;
- существующая программа



Последовательность действий при макетирование

Макетирование

Достоинства

- Обеспечивает определение полных требований к ПО

Недостатки

- Заказчик может принять макет за продукт;
- Разработчик может принять макет за продукт.

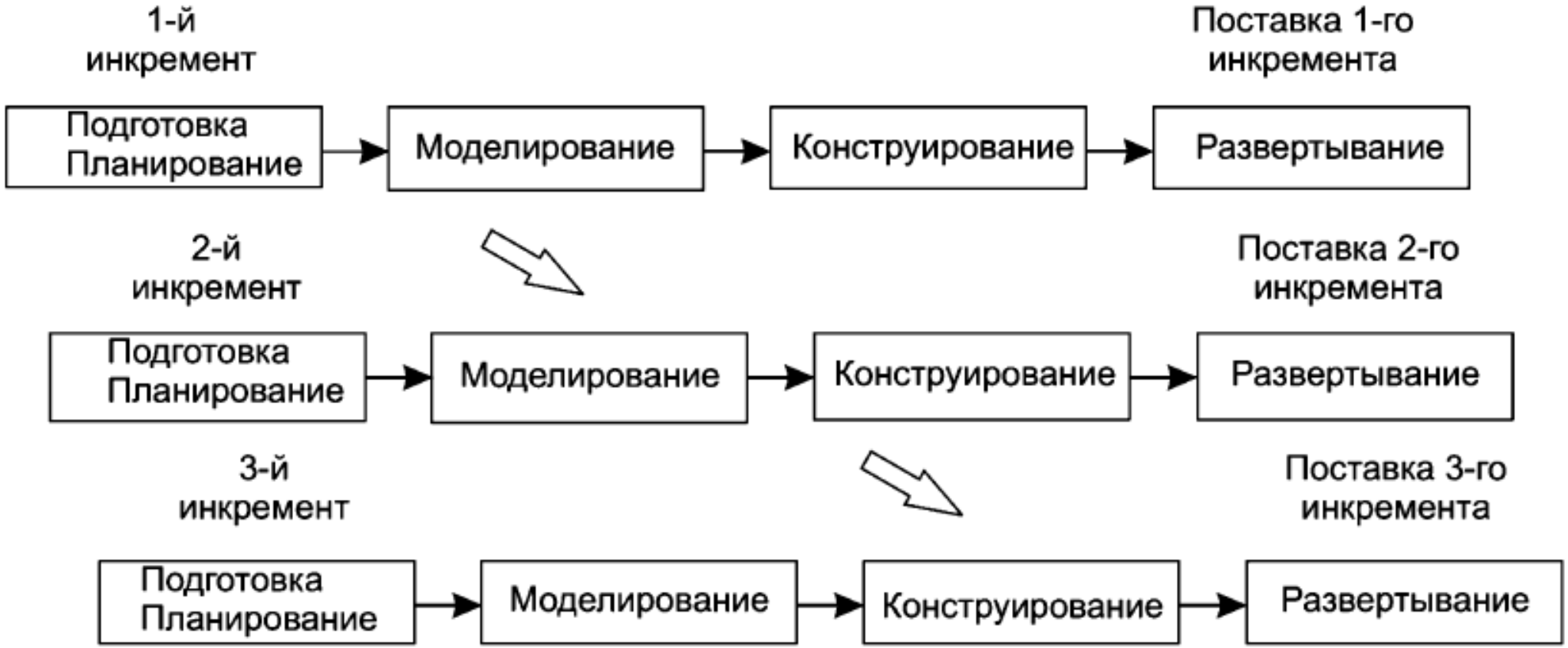
Стратегии разработки ПО

Однократный проход (водопадная стратегия)	• Линейная последовательность этапов разработки
Инкрементная стратегия	• В начале процесса определяются все пользователи и системные требования
Эволюционная стратегия	• Система строится в виде последовательности, но в начале процесса определены не все требования

Характеристики стратегий разработки

Стратегия разработки	В начале процесса определены все требования?	Множество циклов разработки?	Промежуточное ПО распространяется?
Однократный проход	Да	Нет	Нет
Инкрементная (запланированное улучшение продукта)	Да	Да	Может быть
Эволюционная	Нет	Да	Да

Инкрементная модель

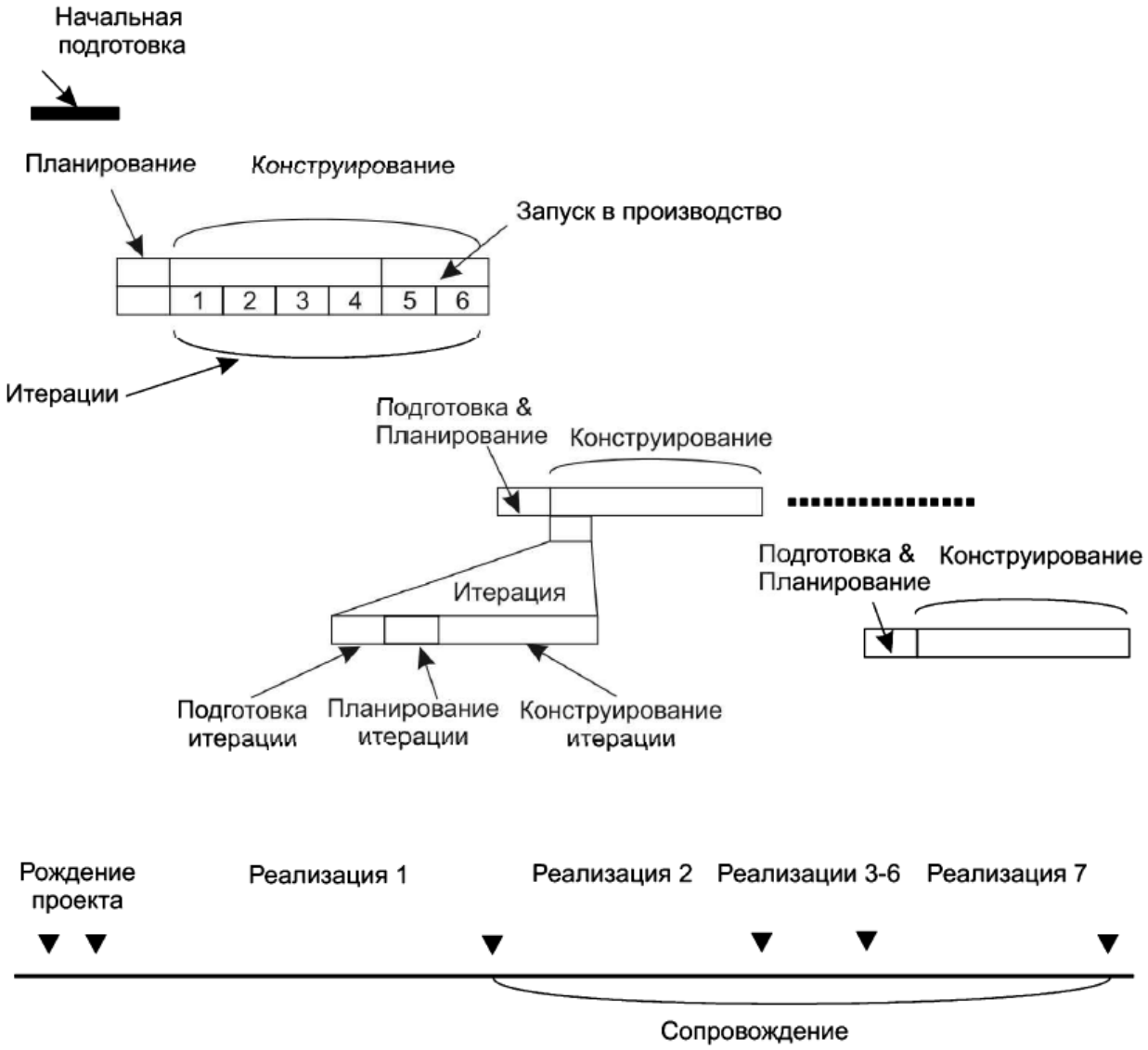


Экстремальное программирование XP

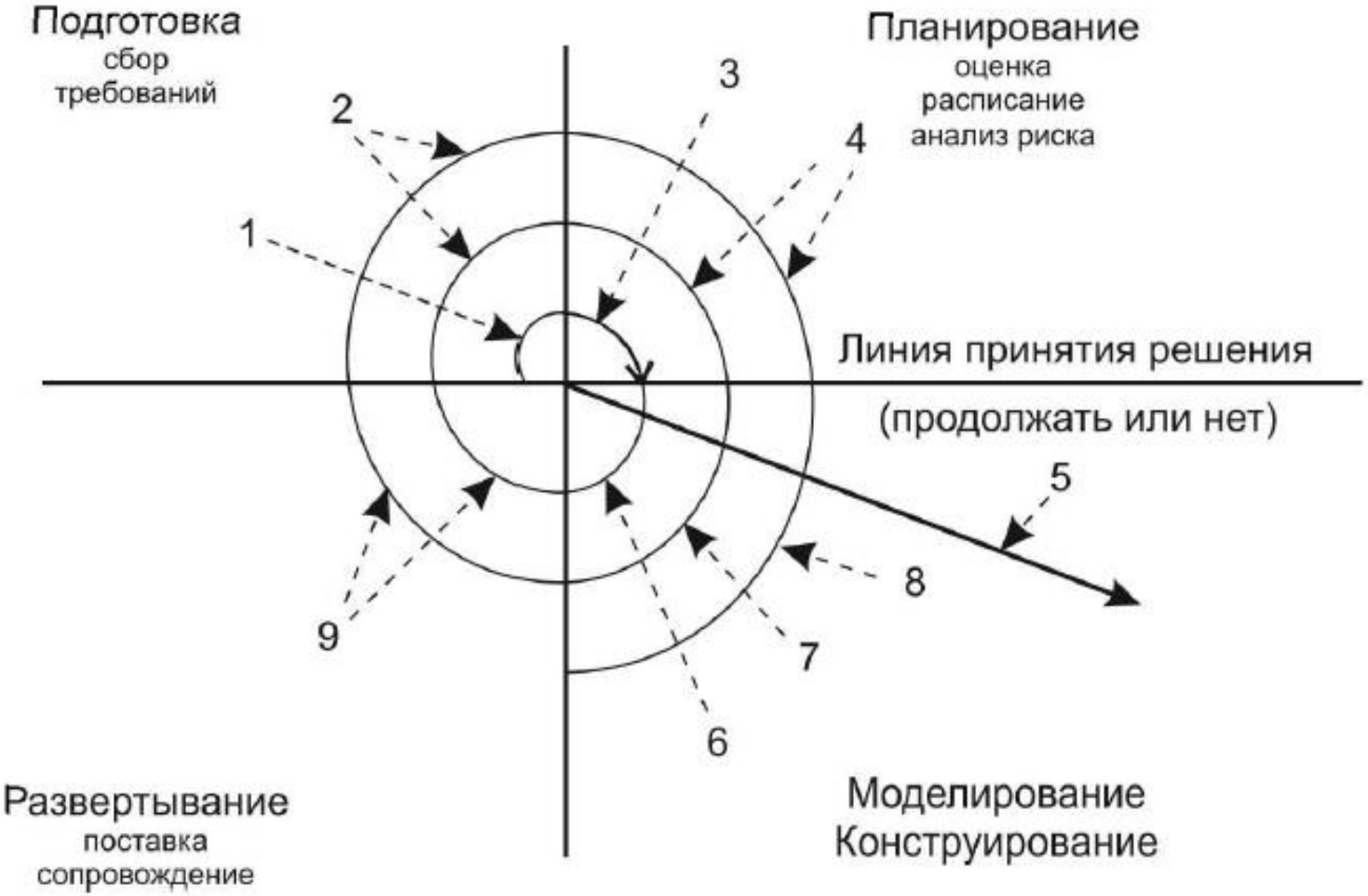
- Основная идея XP – установить высокую стоимость изменения, характерную для приложений с использованием объектов, паттернов и реляционных БД.

Практика здравого смысла	XP экстремум	XP реализация
Проверка кода	проверяется все время	парное программирование
Тестирование	выполняется все время, даже с помощью заказчиков	тесты модулей, приемки
Проектирование	является частью ежедневной деятельности каждого разработчика	рефакторинг
Простота	для системы выбирается простейшее проектное решение, поддерживающее ее текущую функциональность	самая простая вещь, которая могла бы работать
Архитектура	Каждый постоянно работает над уточнением архитектуры	метафора
Тестирование интеграции	Интегрируется и тестируется несколько раз в день	непрерывная интеграция
Короткие итерации	являются предельно короткими, продолжаются секунды, минуты, часы, а не недели, месяцы или годы	игра планирования

Идеальный XP процесс

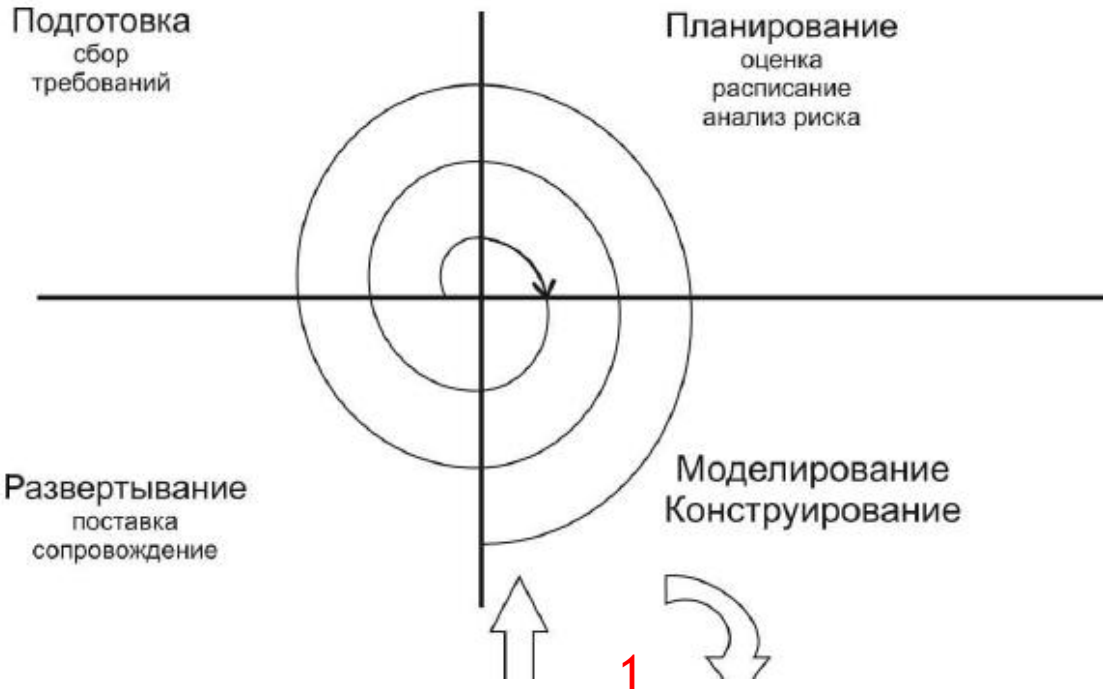


Спиральная модель

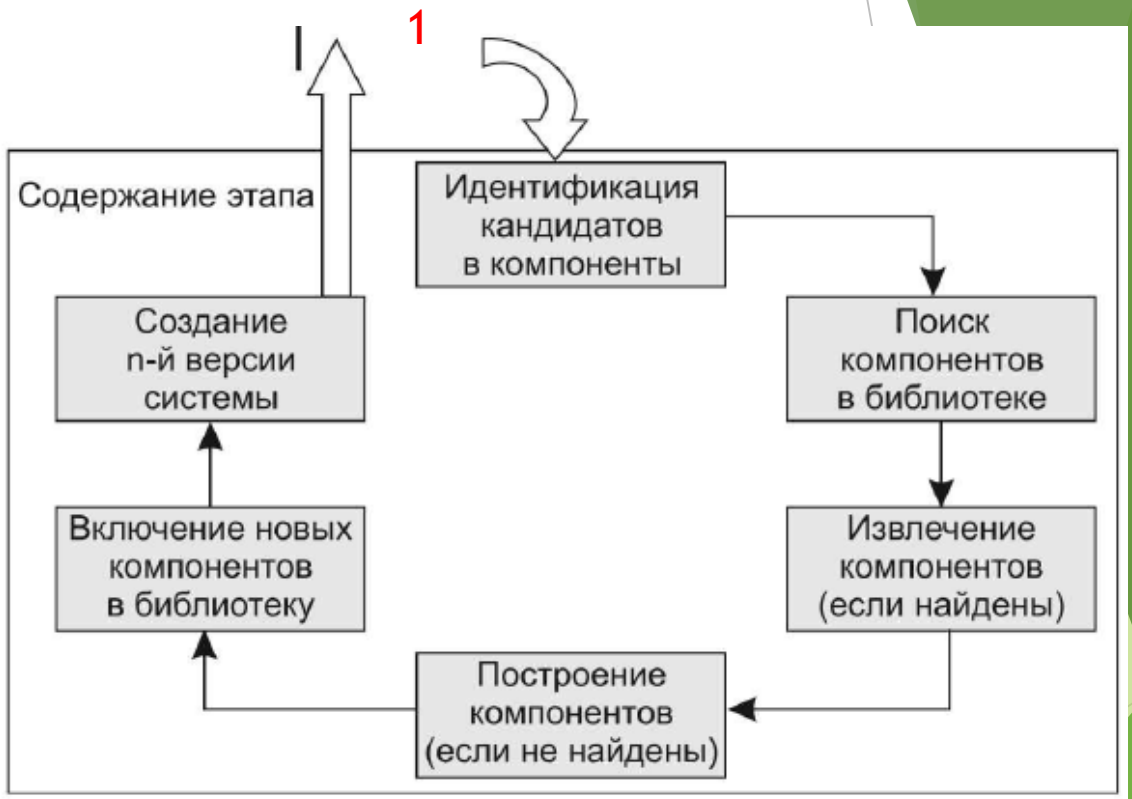


1 – Начальный сбор тестирований проекта;
2- та же работа, но на основе рекомендаций заказчика;
3- планирование проекта и анализ риска на основе начальных требований;
4 – планирование и анализ риска на основе реакции заказчика;
5- переход к комплексной системе;
6 – начальный макет системы;
7 – версия системы следующего уровня;
8- разработанная система;
9 – оценивание заказчиком

Компонентно-ориентированная модель



Продолжение схемы



Достоинства:

- Уменьшает на 30% время разработки программного продукта;
- Уменьшает стоимость программной разработки до 70%;
- Увеличивает в 1,5 раза производительность разработки.

СПАСИБО ЗА ВНИМАНИЕ!

Если остались вопросы, задавайте их в ЭИОС МТУСИ или пишите на почту