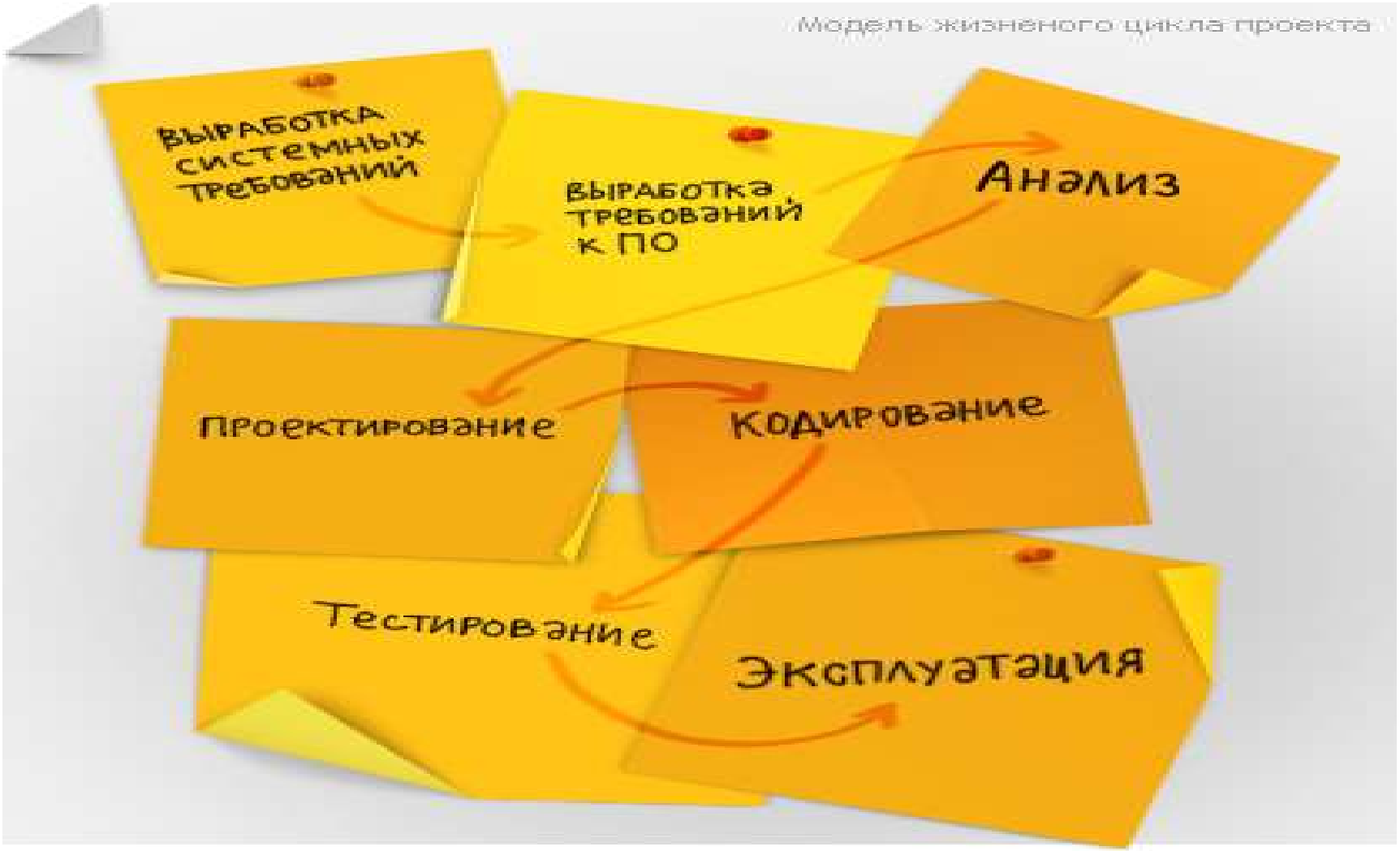
Модели жизненного цикла программного обеспечения

Часть 2



Модель ЖЦ ПО

- Под моделью ЖЦ ПО понимается структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач на протяжении ЖЦ.
- Модель ЖЦ зависит от специфики, масштаба и сложности проекта и специфики условий, в которых система создается и функционирует.

Технология проектирования ПО

- Технология проектирования ПО определяется как совокупность технологических операций проектирования в их последовательности и взаимосвязи, приводящая к разработке проекта ПО.

Технология проектирования ПО

Требования к технологии проектирования ПО:

- соответствие стандарту ISO/IEC 12207 ;
- гарантированное достижение целей разработки в рамках установленного бюджета, с заданным качеством и в установленное время;
- возможность декомпозиции проекта на составные части, разрабатываемые группами исполнителей ограниченной численности (3-7 человек), с последующей интеграцией составных частей;

Технология проектирования ПО

Требования к технологии проектирования ПО:

- минимальное время получения работоспособного ПО;
- независимость получаемых проектных решений от средств реализации (СУБД, операционных систем, языков и систем программирования);
- поддержка комплексом согласованных CASE-средств, обеспечивающих автоматизацию процессов, выполняемых на всех стадиях ЖЦ.

Технология проектирования ПО

Стандарты технологии проектирования:

- Стандарт проектирования;
- Стандарт оформления проектной документации;
- Стандарт интерфейса конечного пользователя.

Проект

- Одним из ключевых понятий технологии разработки программного обеспечения, как и многих других областей деятельности, является понятие проекта.
- Проект есть уникальное временное предприятие, направленное на создание определенного, уникального продукта и услуги.
- Технология управления проектом есть совокупность знаний, навыков, инструментов и методов для планирования и реализации действий, направленных на достижение поставленной в рамках проекта цели.
- Процесс разработки программного обеспечения является плохо определенным и динамичным.

Четыре «П» разработки ПО

- Персонал
  (кто это делает)
- Процесс
  (способ, которым это делается)
- Проект
  (выполнение необходимых действий)
- Продукт
  (артефакты)

Продукт

Артефакт – любой вид информации, создаваемый, изменяемый и используемый сотрудниками при создании системы

Артефакты:

- Само приложение
- Спецификация требований
- Проектная модель
- Исходный и объектный код
- Тестовые процедуры
- …

Проект

Совокупность действий, необходимых для создания артефакта:

- контакт с заказчиком
- написание документации
- проектирование
- программирование
- тестирование
- …

Процесс

- Процесс создания ПО – определение полного набора видов деятельности, необходимых для преобразования требований пользователя в продукт.
- Процесс служит шаблоном для создания проекта.
- Процесс определяет:
  - кто делает
  - что делает
  - когда делает
  - как достичь цели

Семейства процессов разработки ПО

- тяжеловесные (heavyweight)
  - применяются при фиксированных требованиях и многочисленной группе разработчиков разной квалификации
- облегченные (lightweight, agile)
  - применяются при малочисленной группе квалифицированных разработчиков и грамотном заказчике, который имеет возможность участвовать в процессе

Модель ЖЦ ПО

Крайний случай – модель «черного ящика»
(отсутствие какой-либо модели, “code and fix”)

Модель ЖЦ ПО

Стадии в модели «черного ящика»

- Начало проекта
- Безудержный энтузиазм
- Разочарование
- Хаос
- Поиски виновных
- Наказание невиновных
- Награждение непричастных
- Определение требований к системе

Модель ЖЦ ПО

Стадии полного ЖЦ ПО:

- Анализ осуществимости проектных решений
- Планирование и формирование требований к ПО
- Проектирование системы
- Детальное проектирование
- Кодирование
- Интеграция
- Внедрение
- Эксплуатация и сопровождение

Состав модели ЖЦ ПО

В состав модели ЖЦ ПО входят следующие стадии:

1. Формирование требований к ПО.
2. Проектирование.
3. Реализация.
4. Тестирование.
5. Ввод в действие.
6. Эксплуатация и сопровождение.
7. Снятие с эксплуатации.

Формирование требований к ПО

Стадия формирования требований к ПО включает следующие этапы:

- планирование работ;
- проведение обследования деятельности автоматизируемого объекта (организации);
- построение моделей деятельности организации:
- модели "AS-IS" ("как есть");
- модели "TO-BE" ("как должно быть").

Стадия проектирования

Стадия проектирования включает следующие этапы:

- разработка системного проекта;
- разработка технического проекта.

Стадия реализации

- Стадия реализации включает разработку и изготовление продукта с требуемыми характеристиками, в требуемом объеме и в требуемые сроки.

Стадия тестирования

- Стадия тестирования включает проверку соответствия разработанной программы заявленным требованиям:

➢ тестирование

➢сертификация

Стадия ввода в действие

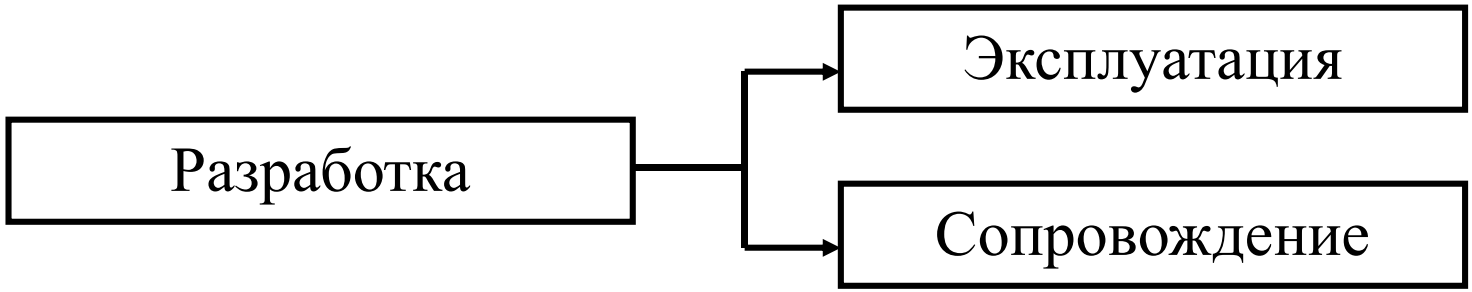
Стадия ввода в действие включает следующие этапы:

- подготовка производства и персонала к использованию программы;
- адаптация программы к особенностям ее использования.

Стадия эксплуатации и сопровождения

Стадия эксплуатации и сопровождения определяет следующие действия:

- Эксплуатация программы в производстве
- Помощь пользователям, выявление и разрешение проблем, связанных с использованием программы

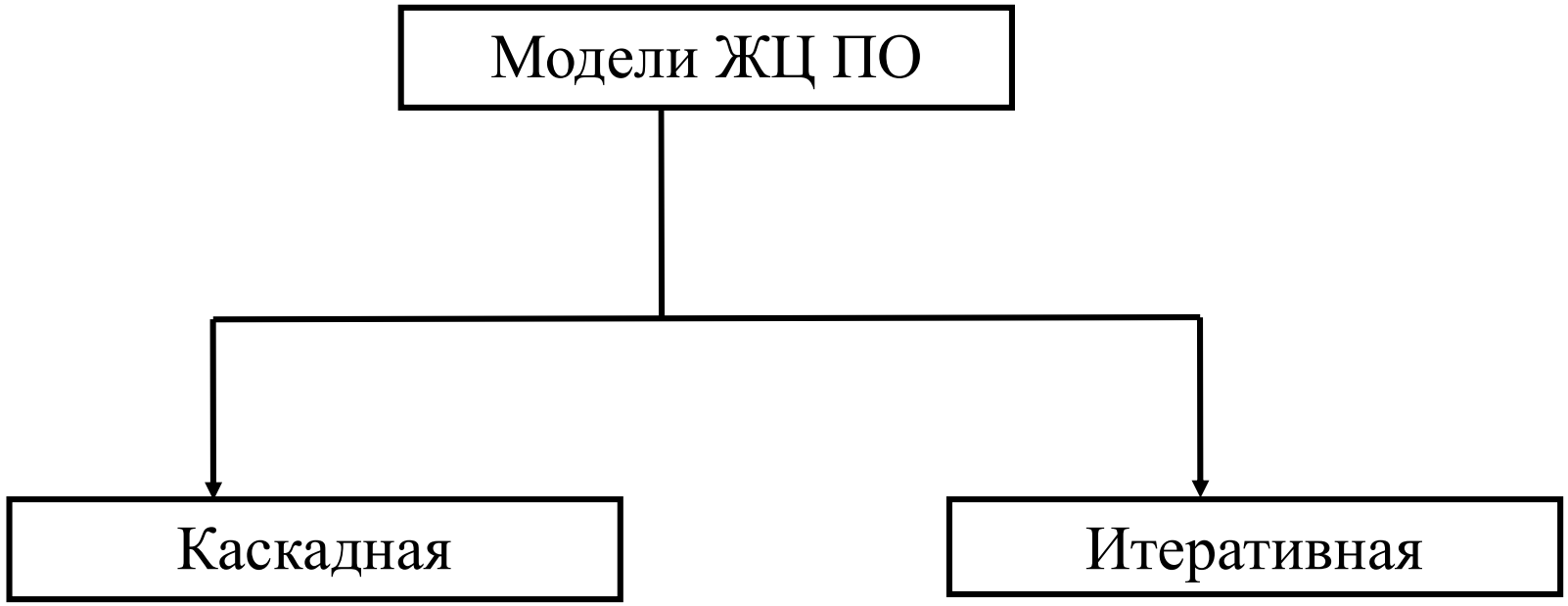


Стадия снятия с эксплуатации

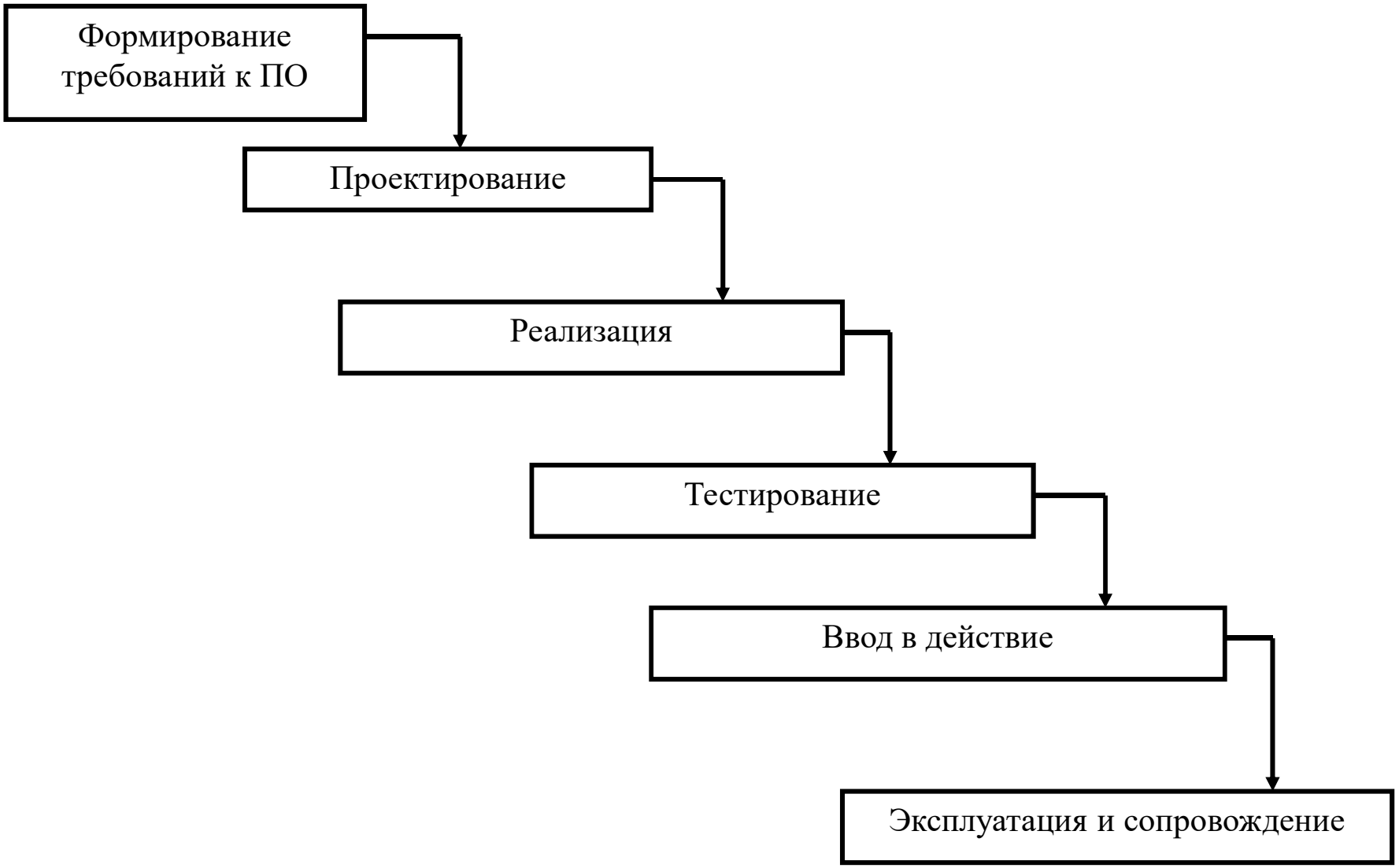
Стадия снятия с эксплуатации определяет следующие действия:

- прекращение использования ПС;
- архивация или утилизация системы и ее компонентов

Классификация моделей ЖЦ ПО



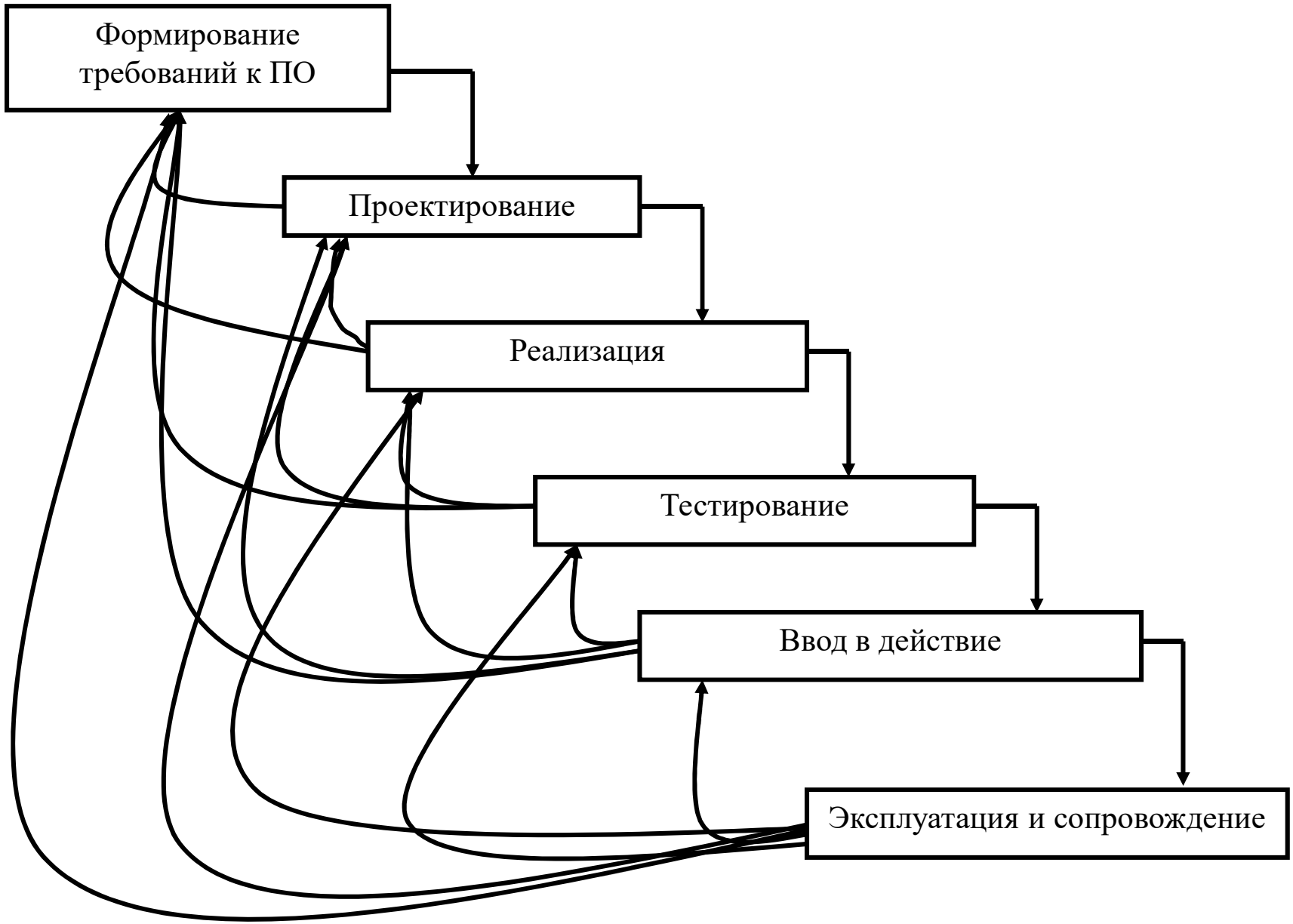
Каскадная модель (waterfall model)



Каскадная модель

- Особенности каскадной модели:
  - Фиксированный набор стадий
  - Каждая стадия -> законченный результат
  - Стадия начинается, когда закончилась предыдущая.
- Недостатки: **негибкость**
  - фаза должна быть закончена, прежде чем приступить к следующей
  - Набор фаз фиксирован
  - Тяжело реагировать на изменения требований
- **Использование**: там, где требования хорошо понятны и стабильны.

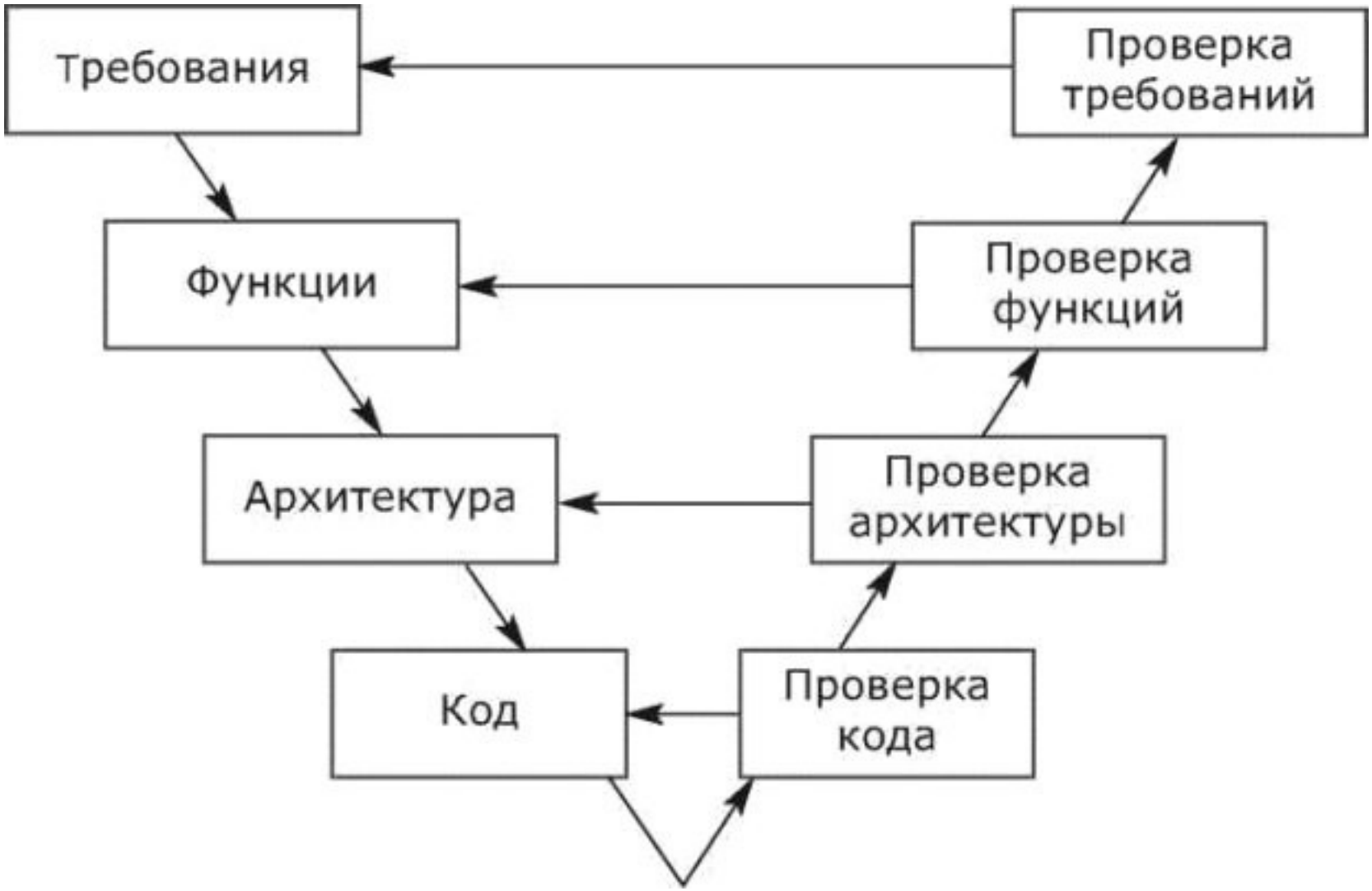
Поэтапная (каскадная) модель с промежуточным контролем



Поэтапная (каскадная) модель с промежуточным контролем

- Разработка ИС ведется итерациями с циклами обратной связи между этапами. Межэтапные корректировки позволяют учитывать реально существующее взаимовлияние результатов разработки на различных этапах; время жизни каждого из этапов растягивается на весь период разработки.
- Самый весомый недостаток: 10-ти кратное увеличение затрат на разработку.

V-образная модель



V-образная модель

V- образная модель как дальнейшее развитие каскадной модели была создана с целью помочь работающей над проектом команде в планировании с обеспечением дальнейшей возможности тестирования системы. В этой модели особое значение придается действиям, направленным на верификацию и аттестацию ИС. Она демонстрирует, что тестирование производится на ранних этапах ЖЦ разработки.

Использование V-образной модели наиболее эффективно в следующих случаях:

1) при разработке проектов, для которых требования максимально четко определены и доступны заранее

2) при разработке проектов, для которых определены и понятны методы реализации решения и технология, а разработчики имеют опыт в работе с данной технологией

3) при разработке систем, в которых требуется высокая надежность.

V-образная модель

Преимущества:

1) планирование на ранних стадиях разработки системы ее тестирования

2) обеспечение аттестации и верификации всех промежуточных результатов разработки

3) упрощение отслеживания хода процесса разработки

4) простота в использовании

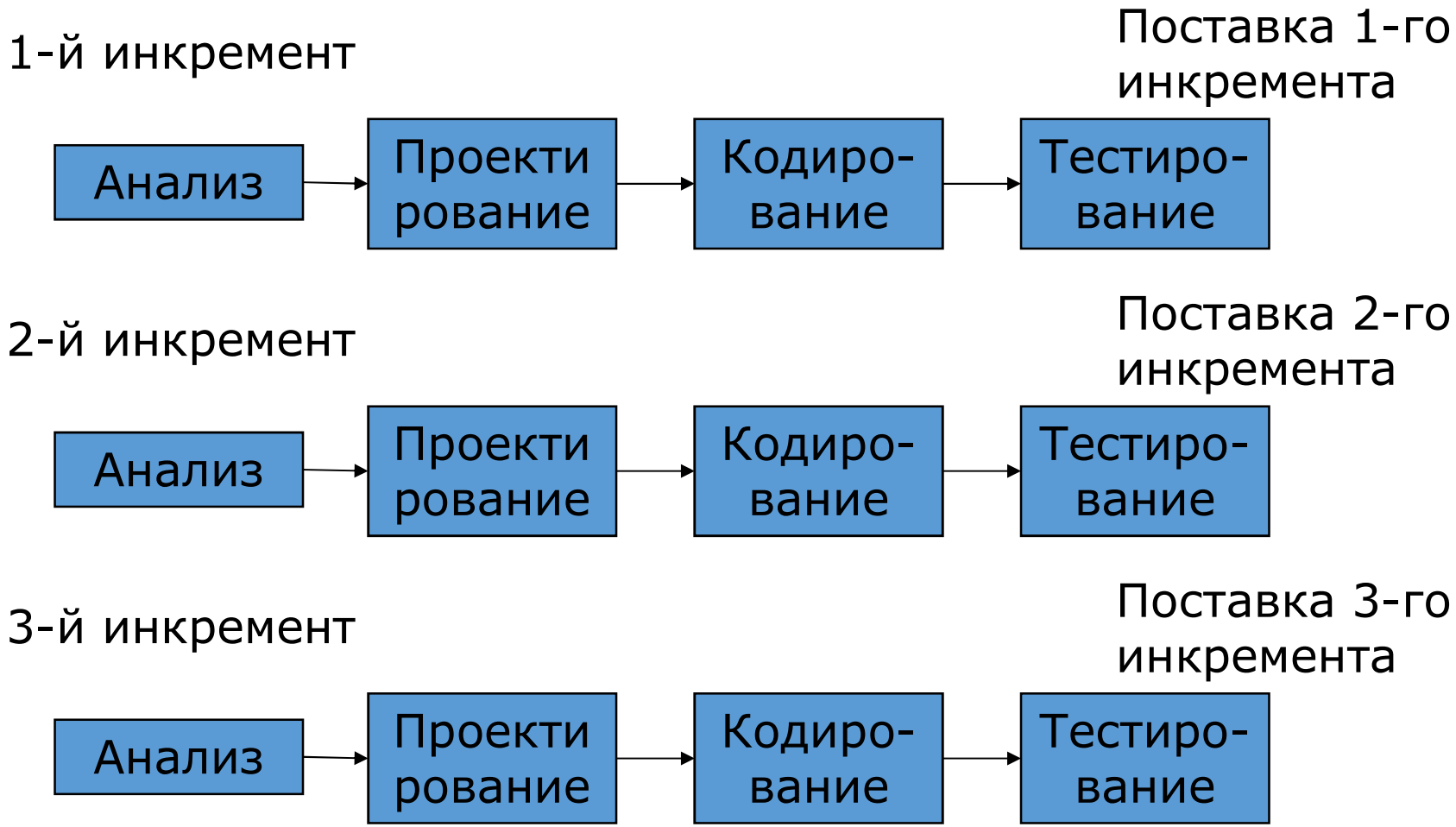
Недостатки:

1) сложность поддержки параллельных событий

3) невозможность внесения динамических изменений в требования на разных этапах жизненного цикла

5) отсутствие в модели действий, направленные на анализ рисков

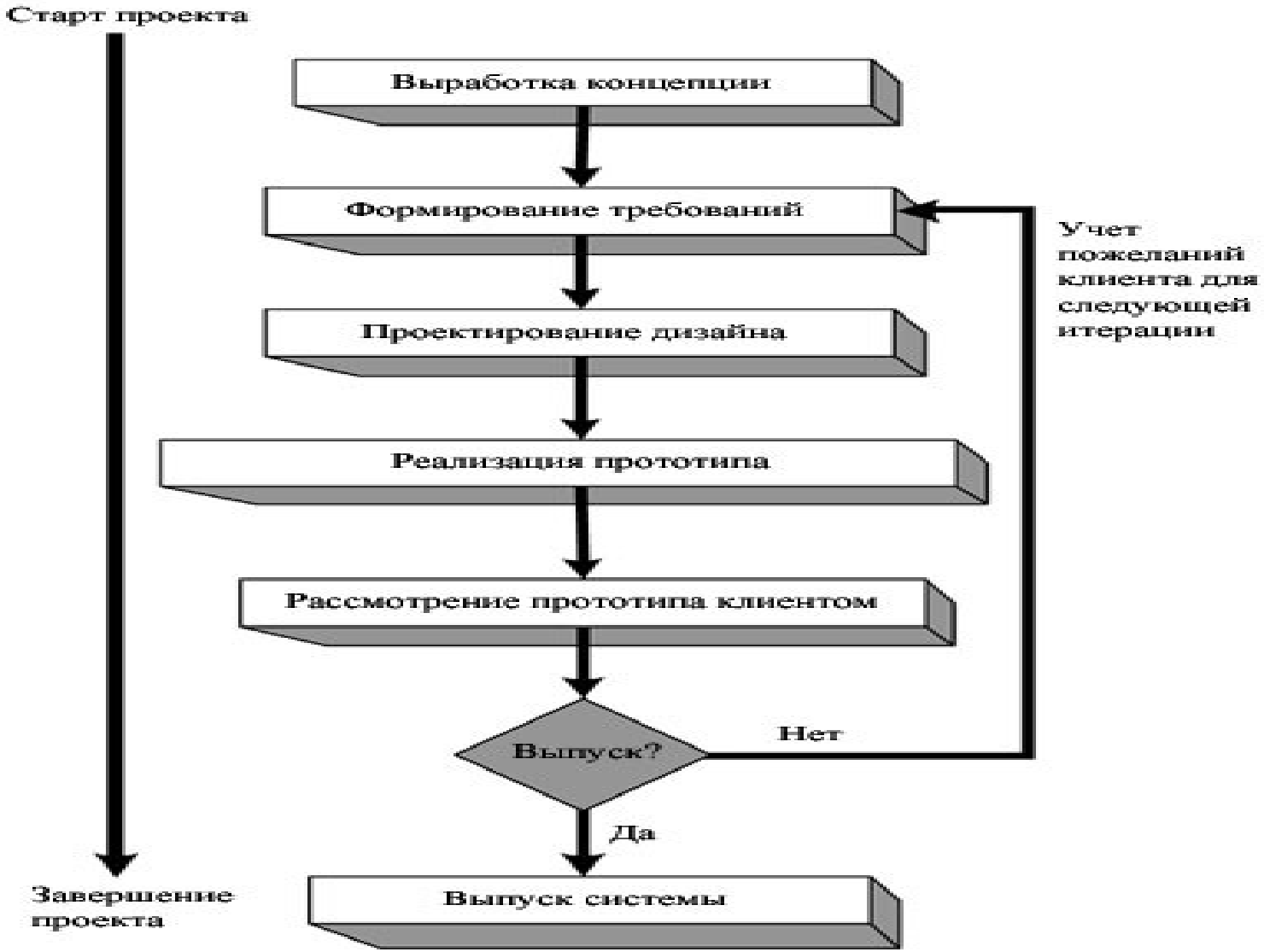
Инкрементная модель



Инкрементная модель

В инкрементной модели полные требования к системе делятся на различные сборки. Процедура разработки по инкрементной модели предполагает выпуск на первом большом этапе продукта в базовой функциональности, а затем уже последовательное добавление новых функций, так называемых «инкрементов». Процесс продолжается до тех пор, пока не будет создана полная система.

Эволюционная модель



Эволюционная модель

Особенности эволюционной модели:

- Стадии повторяются неоднократно.

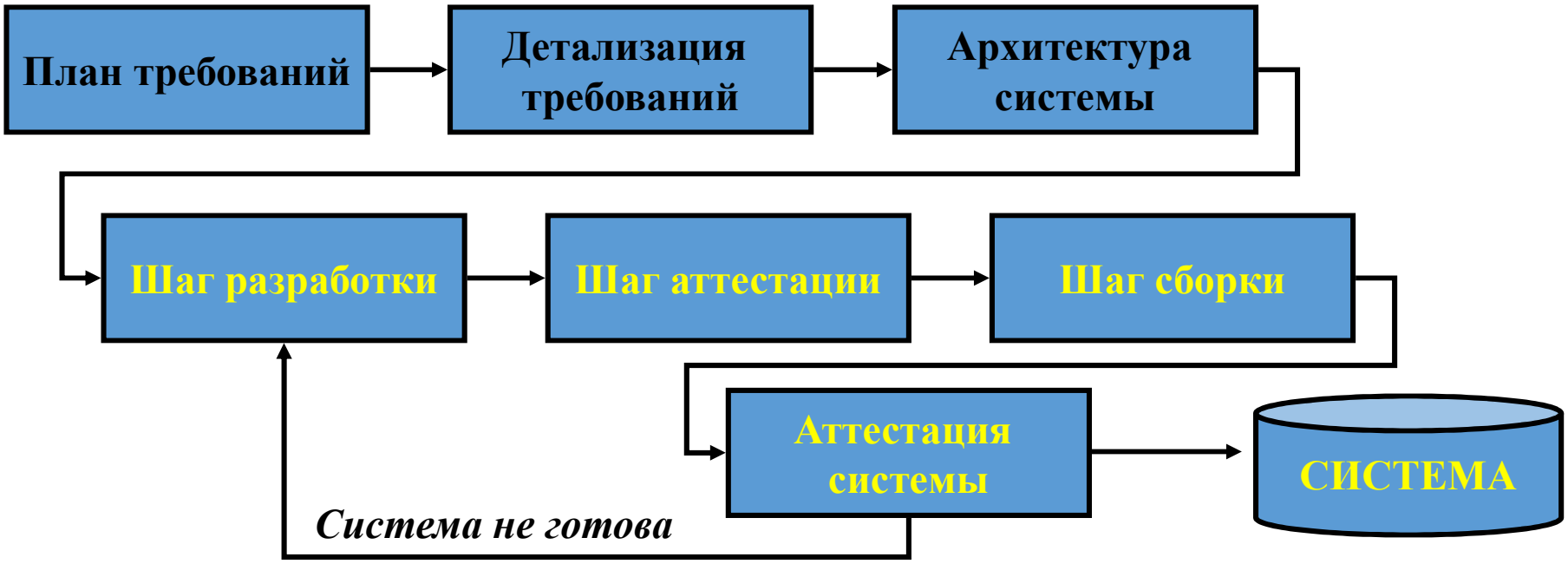
  Сначала для плохо сформулированных требований выполняется весь цикл работ по созданию работающего прототипа. Потом уточняются требования и все повторяется... На выходе – продукт, отвечающий потребностям пользователей.

- Недостатки:
  - Система часто плохо структурирована
  - Проект «не прозрачен»
  - Требуются средства для быстрой разработки
  - Подходит для малых и средних проектов

Модель пошаговой разработки

Модель пошаговой разработки (Миллс):

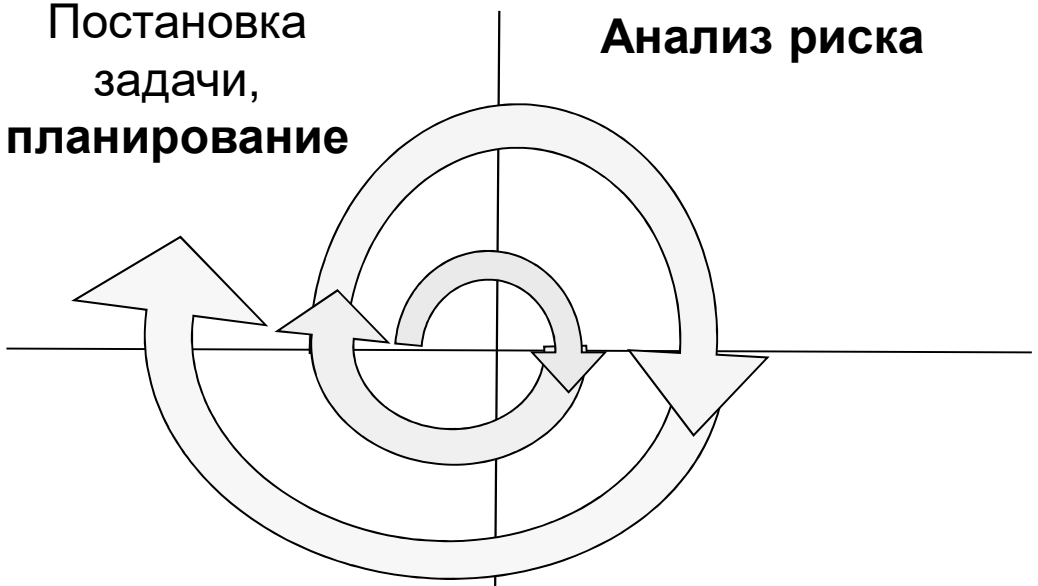
- Шаги. Каждый шаг – работающий прототип.
- Наиболее важные для заказчика компоненты – в начале.
- Требования фиксированы во время шага.
- Для шага можно применять каскадную или эволюционную модель.



Спиральная модель

- Программное обеспечение создается итерационно с использованием метода прототипирования.
- Прототипом обычно называют действующий программный продукт, реализующий отдельные функции и внешние интерфейсы разрабатываемого программного обеспечения.

На 1-й итерации может использоваться макет, который оценивается заказчиком.



Спиральная модель

Особенности спиральной модели:

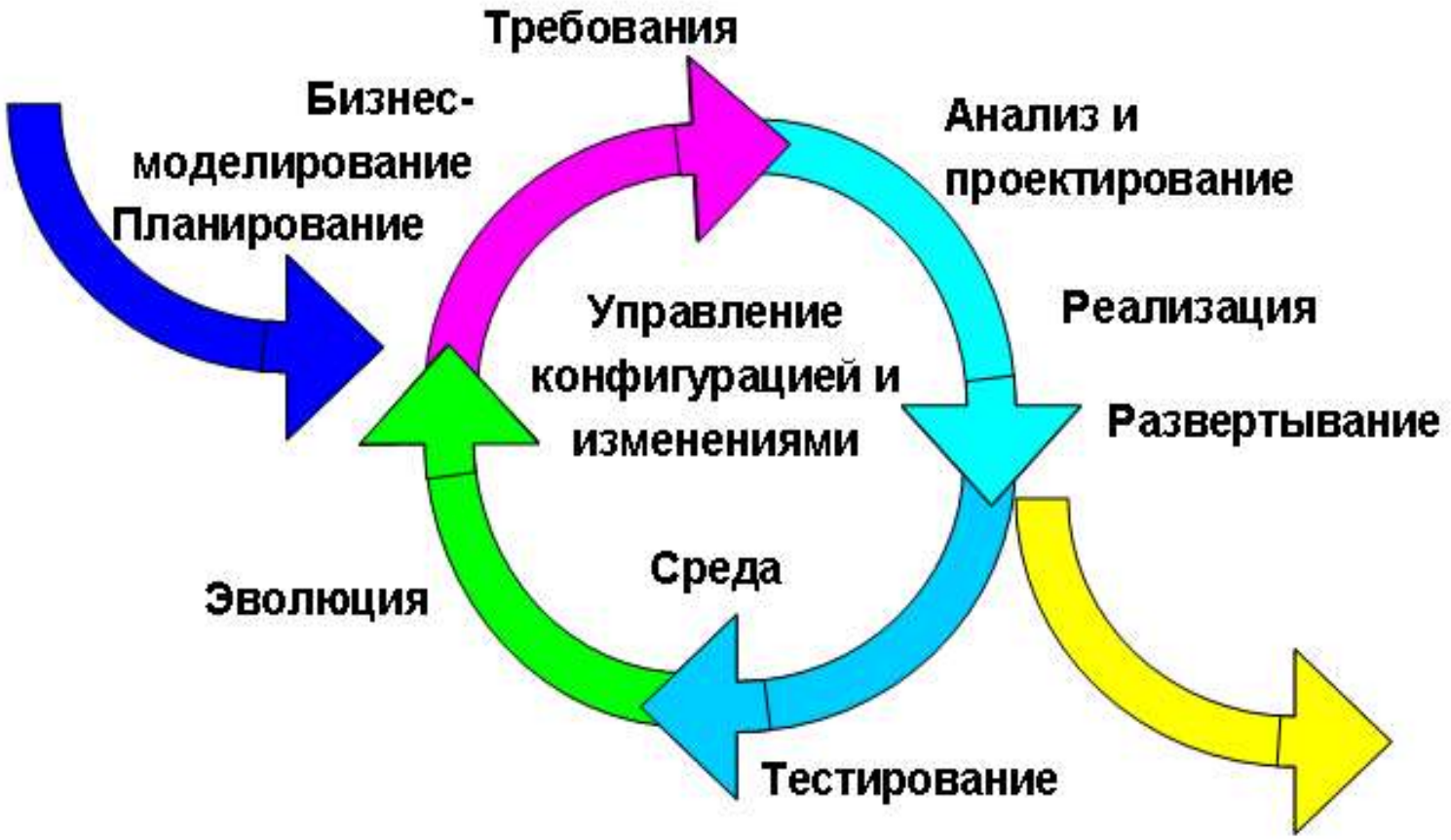
- Вместо действий с обратной связью – спираль.
- Каждый виток спирали соответствует 1 итерации.
- Нет заранее фиксированных фаз. В зависимости от потребностей.
- Каждый виток разбит на 4 сектора:
  - Определение целей
  - Оценка и разрешение рисков
  - Разработка и тестирование
  - Планирование
- **Главное отличие: акцент на анализ и преодоление рисков.**
- На каждом витке могут применяться разные модели процесса разработки ПО.

Итеративная (итерационная) модель

Итеративная разработка ПО — это процесс создания программного обеспечения, который осуществляется небольшими этапами, в ходе которых ведется анализ полученных промежуточных результатов, выдвигаются новые требования и корректируются предыдущие этапы работы.

Цель каждой итерации — это получение версии ПО, включающей в себя как новые возможности, реализованные в ходе текущей итерации, так и функциональность всех предыдущих итераций. Результат финальной итерации содержит всю требуемую функциональность продукта.

Итеративная модель



Итеративная модель

Преимущества:

- Снижение рисков
- Организация эффективной обратной связи проектной команды с потребителем, создание продукта, реально отвечающего его потребностям
- Быстрый выпуск минимально ценного продукта (MVP) и возможность вывести продукт на рынок и начать эксплуатацию гораздо раньше

Недостатки:

- Проблемы с архитектурой и накладные расходы
- Нет фиксированного бюджета и сроков

Методология

- **Методология** – принципы и способы организации теоретической и практической деятельности. Совокупность методов, применяемых в какой-либо науке.

Для проектирования ПО:

Методология есть методы, принципы и способы организации деятельности проектной группы для создания программного средства.

Методологии разработки ПО

- RAD (Rapid Application Development) Быстрая разработка
- Rational Unified Process (RUP).
- Microsoft Solution Framework (MSF)
- Extreme Programming (XP). Экстремальное программирование (самая новая среди рассматриваемых методологий)

Быстрая разработка RAD (Rapid Application Development).

Модель RAD (Rapid Application Development).

- Метод быстрой разработки приложений. Подходов к разработке прикладного ПО в рамках спиральной модели ЖЦ.
- Метод основан на последовательной итерации эволюционной системы или прототипов, которые анализируются совместно с заказчиком.

RUP (Rational Unified Process)

Rational Unified Process

- Фирма Rational Software предложила свою модель жизненного цикла, которая называется Rational Objectory Process.

Основные свойства данной технологии:

- процесс итеративный, т.е. происходит последовательное уточнение результатов,
- действия процесса направлены на создание моделей, а не других элементов проекта, например, текстовых документов,
- действия жизненного цикла определяются в первую очередь блоками использования (use case).

Основные принципы RUP

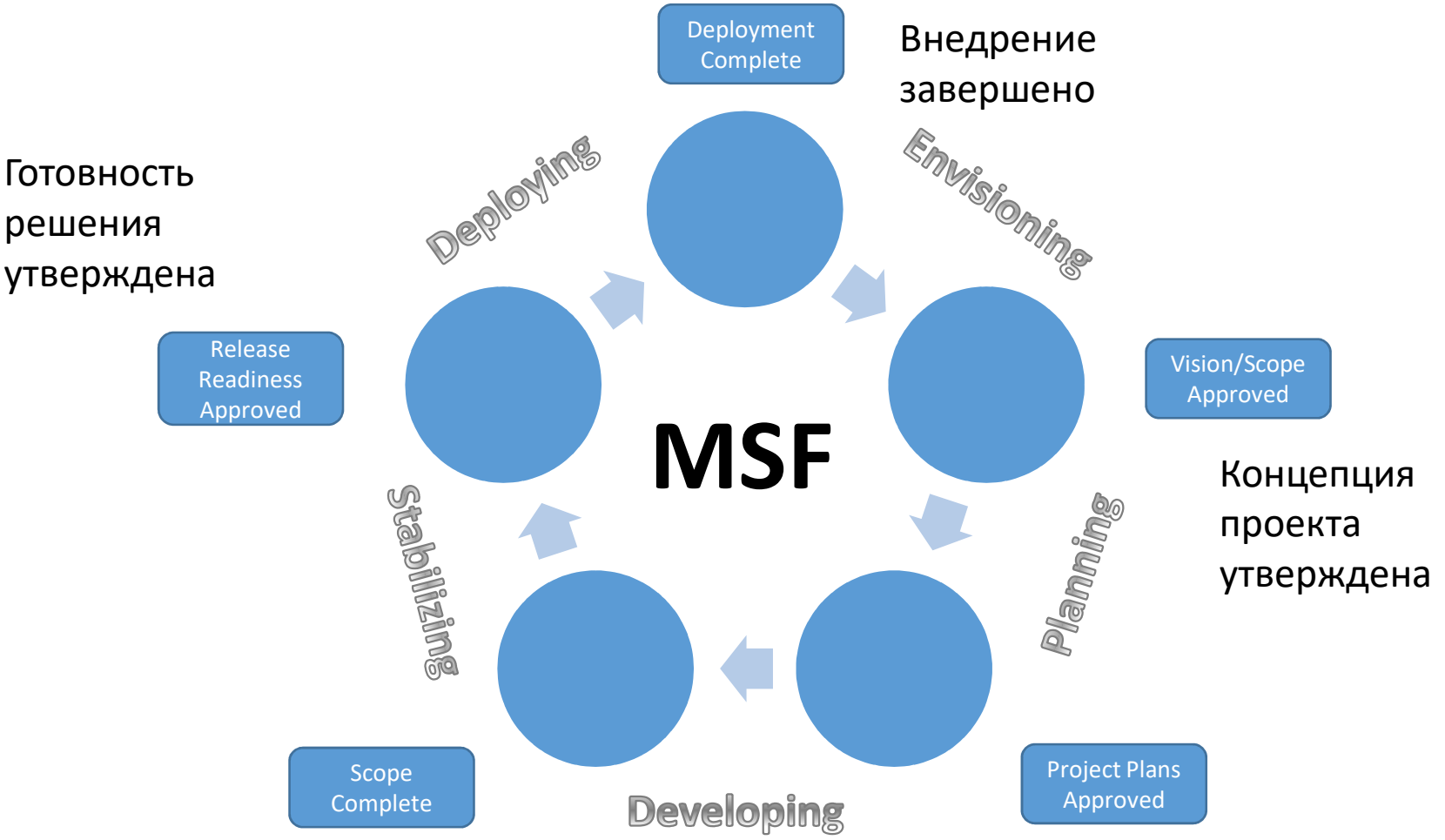
- Снижение риска (итерационный подход к созданию ПО)
- Выполнению требований заказчиков (планирование и управление проектом на основе требований)
- Построение системы на базе компонентной архитектуры ПО
- Визуальное моделирование
- Обеспечение высокого качество (упреждающее тестирование)
- Управление изменениями

MSF (Microsoft Solutions Framework)

Методология MSF

- MSF – методология разработки программного обеспечения от компании Microsoft, опирающаяся на практический опыт компании и описывающая управление людьми и управление процессами в ходе разработки решения.

Microsoft Solutions Framework (MSF)



Экстремальное программирование

Терминология

- **Экстрема́льное программи́рование (XP)**— одна из гибких методологий разработки программного обеспечения.
- **Гибкая методология разработки** — это концептуальный каркас, в рамках которого выполняется разработка программного обеспечения.

Основные приёмы XP

Двенадцать основных приёмов экстремального программирования могут быть объединены в **четыре группы**:

- Короткий цикл обратной связи
- Непрерывный, а не пакетный процесс
- Понимание, разделяемое всеми
- Социальная защищенность программиста

Основные приёмы XP

Короткий цикл обратной связи (Fine scale feedback):

- Разработка через тестирование (Test driven development)
- Игра в планирование (Planning game)
- Заказчик всегда рядом (Whole team, Onsite customer)
- Парное программирование (Pair programming)

Основные приёмы XP

Непрерывный, а не пакетный процесс:

- Непрерывная интеграция (Continuous Integration)
- Рефакторинг (Design Improvement, Refactor)
- Частые небольшие релизы (Small Releases)

Основные приёмы XP

Понимание, разделяемое всеми:

- Простота (Simple design)
- Метафора системы (System metaphor)
- Коллективное владение кодом (Collective code ownership) или выбранными шаблонами проектирования (Collective patterns ownership)
- Стандарт кодирования (Coding standard or Coding conventions)

Основные приёмы XP

Социальная защищенность программиста (Programmer welfare):

•40-часовая рабочая неделя (Sustainable pace, Forty hour week)