Программная инженерия

Программа курса

- 17 лекций
- Аттестация: экзамен в виде теста

Основные темы курса

- Жизненный цикл ПО
- Составление требований к ПО и варианты использования
- Объектно-ориентированный анализ ПО
- Объектно-ориентированное проектирование ПО
- Кодирование и тестирование
- Архитектура программных систем
- Метрики и оценка качества ПО

Литература

- Ian Sommerville. **Software Engineering** (8th Edition). 2006. 864 p.
- Арлоу Д., Нейштад А. **UML 2 и Унифицированный процесс.** Практический объектно-ориентированный анализ и проектирование, 2-е издание. 2007. 624 с.
- Брукс Ф. **Мифический человеко-месяц**, или Как создаются программные системы. 2007. 304 с.
- Marsic I. **Software Engineering**. 2009. 430 p.

Программная инженерия

Инженерия

- ***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.

Орфография

Инжен’ерия

Engin’eering

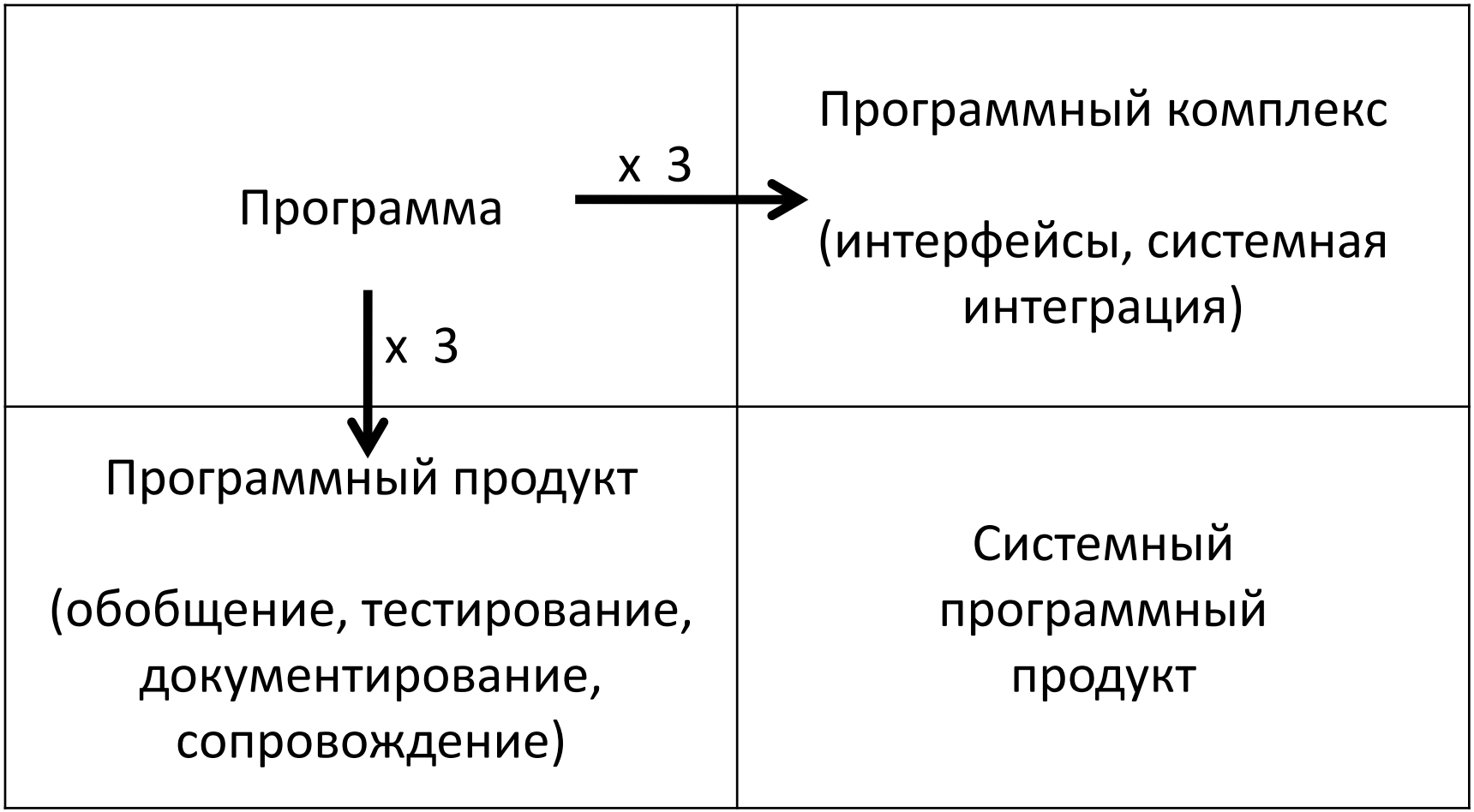
Программная инженерия

- Термин был предложен в 1968 г. на конференции посвященной «***Кризису ПО***», возникшего в результате появления интегральных схем и катастрофического усложнения ПО:
  - Реализация проектов задерживалась на годы
  - Стоимость проектов в десятки раз превышала прогнозируемую
  - Необходимы были методы разработки и контроля таких сложных программных систем

Программное обеспечение

- ***Программное обеспечение (программный продукт)*** – это компьютерная программа *и соответствующая документация*. Программные продукты могут быть разработаны как для конкретного заказчика, так и для всего рынка в целом.

Программа и программный продукт



Программная инженерия

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

Программная инженерия VS компьютерные науки

Программная инженерия

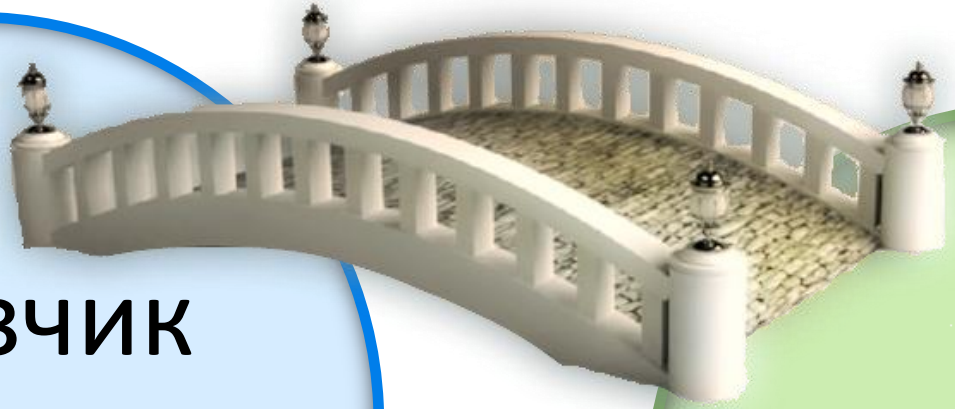
Практика и подходы к разработке полезных для конечного пользователя программных продуктов

Компьютерные науки

Теория и фундаментальные основы по созданию алгоритмов и компьютерных программ

В идеале, каждый программный инженер должен знать компьютерные науки (как каждый электрик должен знать физику)

Программная инженерия



Заказчик

Задача, требования, ограничения

Программист

Хорошее программное обеспечение

Можно выделить следующие важные признаки хорошего ПО:

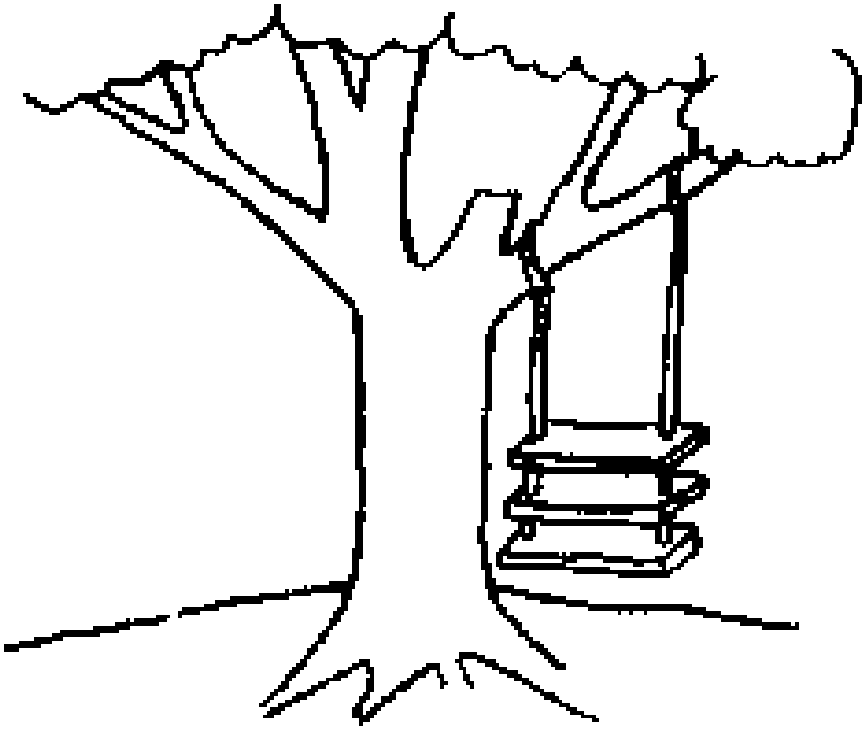
- Удобство сопровождения
- Функциональная надежность
- Эффективность
- Применимость

Современные проблемы программной инженерии

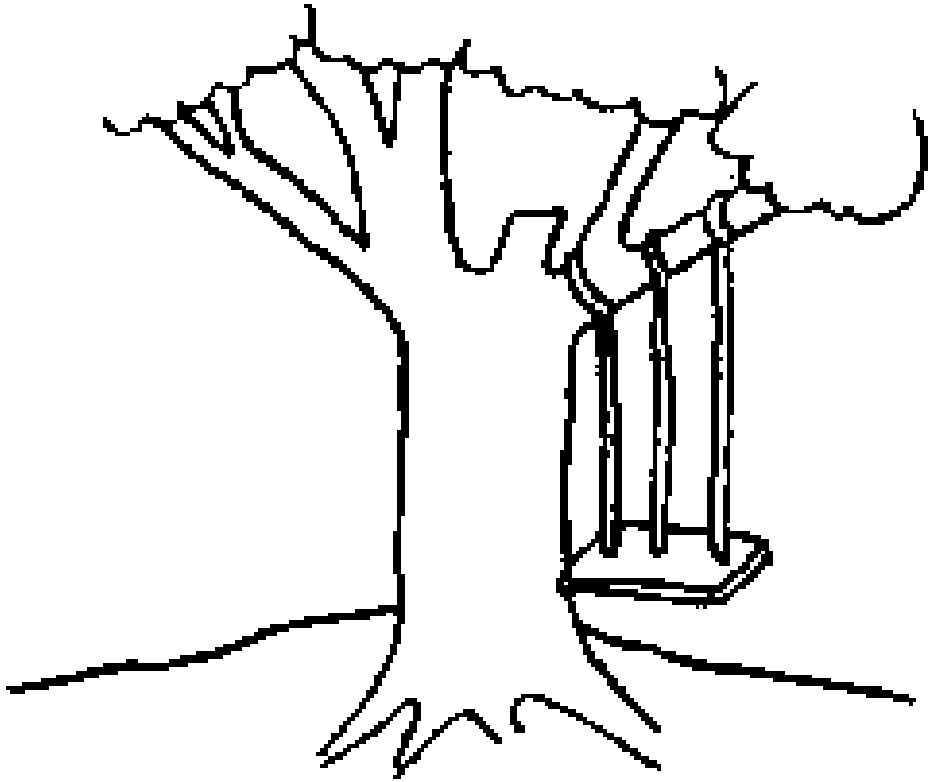
- Проблема гетерогенности
- Проблема своевременного представления результатов
- Проблема доверия

Процесс разработки ПО (Жизненный цикл ПО)

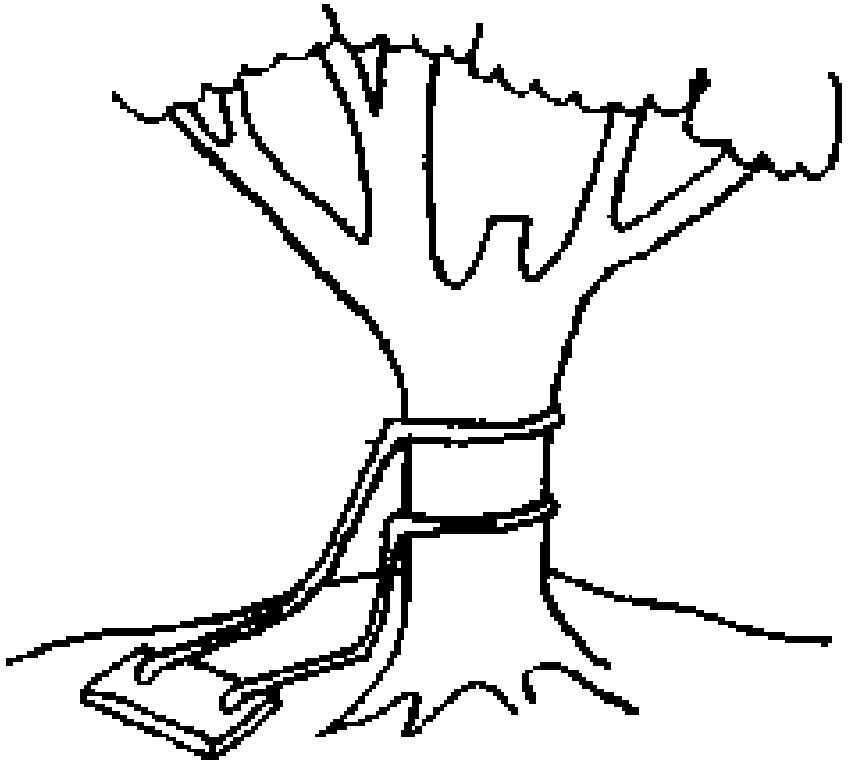
«Качели» - как проектируются программы (1975 !)



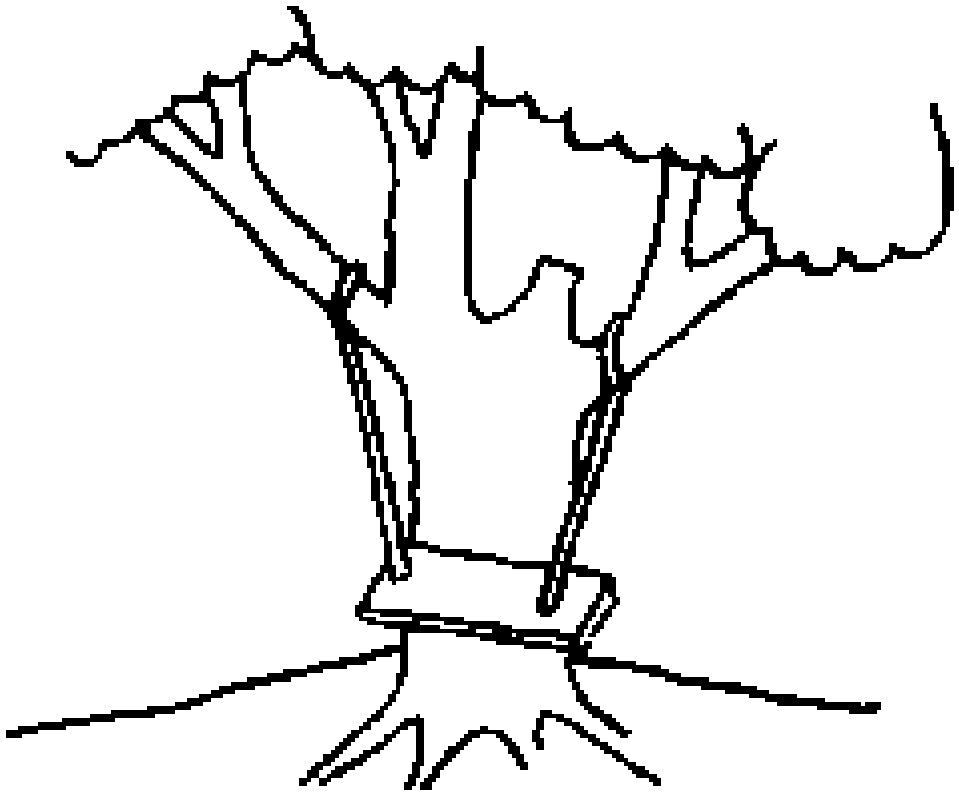
Как было предложено организатором разработки



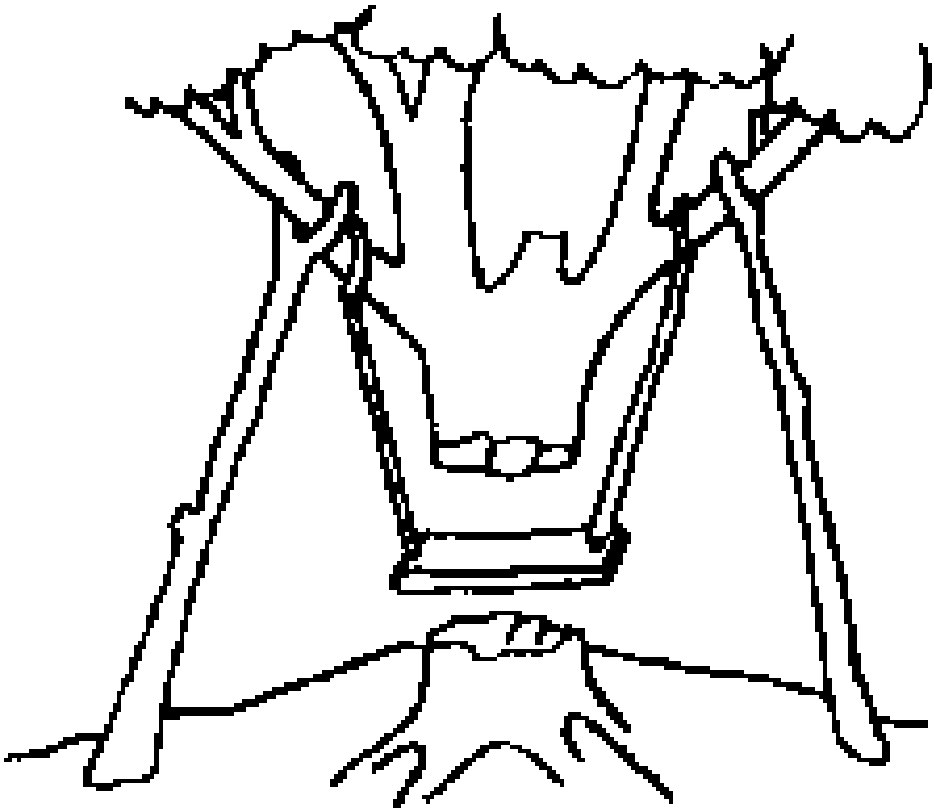
Как было описано
в техническом задании



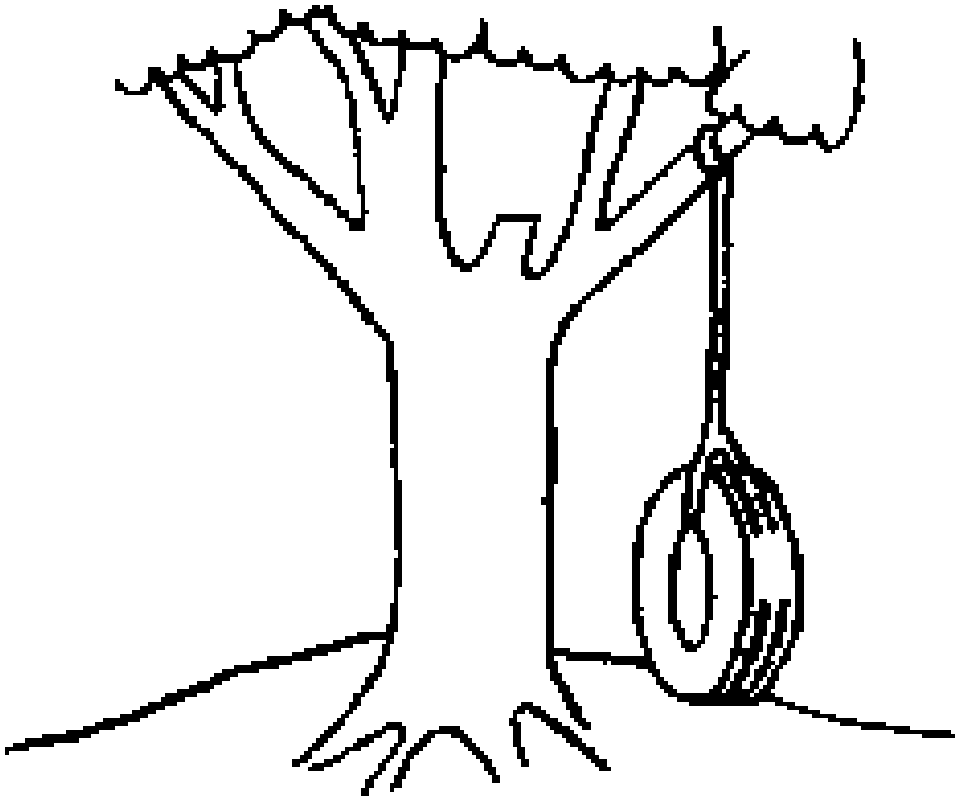
Как было спроектировано
ведущим системным специалистом



Как было реализовано программистами



Как было внедрено



Чего хотел пользователь

Процесс разработки ПО

- ***Процесс разработки ПО (жизненный цикл ПО)*** – это набор действий и связанных с ними результатов, направленных на разработку и/или развитие программного продукта:

1. Спецификация требований
2. Разработка
3. Валидация
4. Развитие

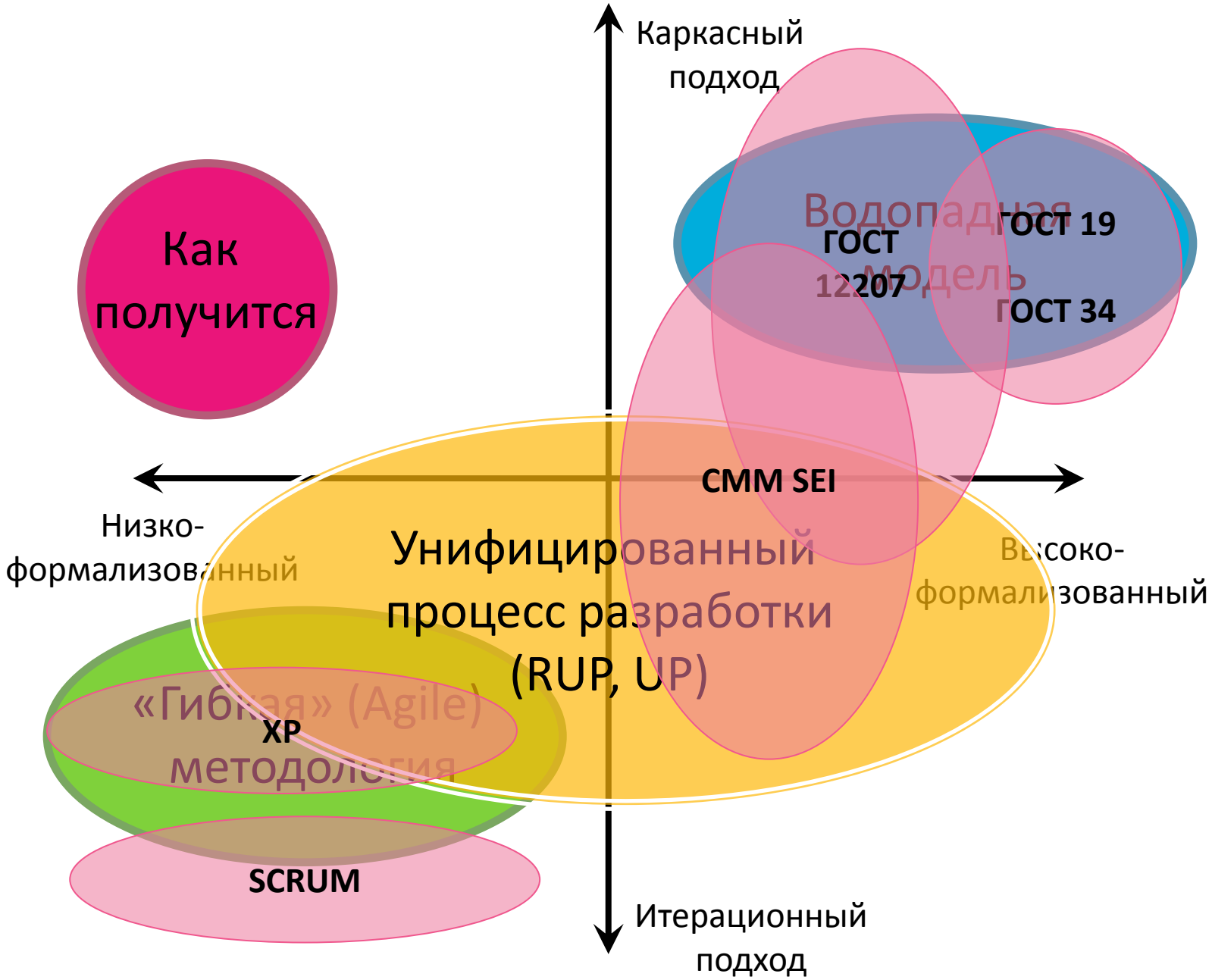
Процесс разработки ПО

- Не существует «идеального процесса разработки ПО»

НЕ СУЩЕСТВУЕТ!!!!

Модель процесса разработки ПО

- ***Модель процесса разработки ПО*** – это абстрактная репрезентация процесса разработки ПО, представляющая данный процесс в определенной перспективе.
- Модель – это не всеобъемлющее описание процесса разработки ПО. Это скорее абстракция, которая позволяет описать различные подходы к процессу разработки.


Каркасный подход
Как получится
Водопадная модель
ГОСТ 12207
ГОСТ 19
ГОСТ 34
CMM SEI
Низко-формализованный
Высоко-формализованный
Унифицированный процесс разработки (RUP, UP)
«Гибкая» (Agile) методология
XP
SCRUM
Итерационный подход

Модели разработки ПО

Можно выделить 3 основных модели разработки ПО:

 Водопадная модель

 Поэтапная (эволюционная) разработка

 Компонентно-ориентированная разработка

Часто эти модели объединяются в рамках одного проекта (подсистемы могут создаваться посредством различных подходов)

Водопадная модель

Фазы водопадной модели

- Результатом каждой фазы в водопадной модели является один или несколько утвержденных документов.
- Последующая фаза не может начаться пока предыдущая не завершена.
- Но только в идеальном мире процесс разработки линеен и не обладает обратными связями
- => в реальном мире в водопадной модели подразумевается несколько итераций, после которых результат «замораживается» и происходит переход ко следующей фазе.

Достоинства и недостатки

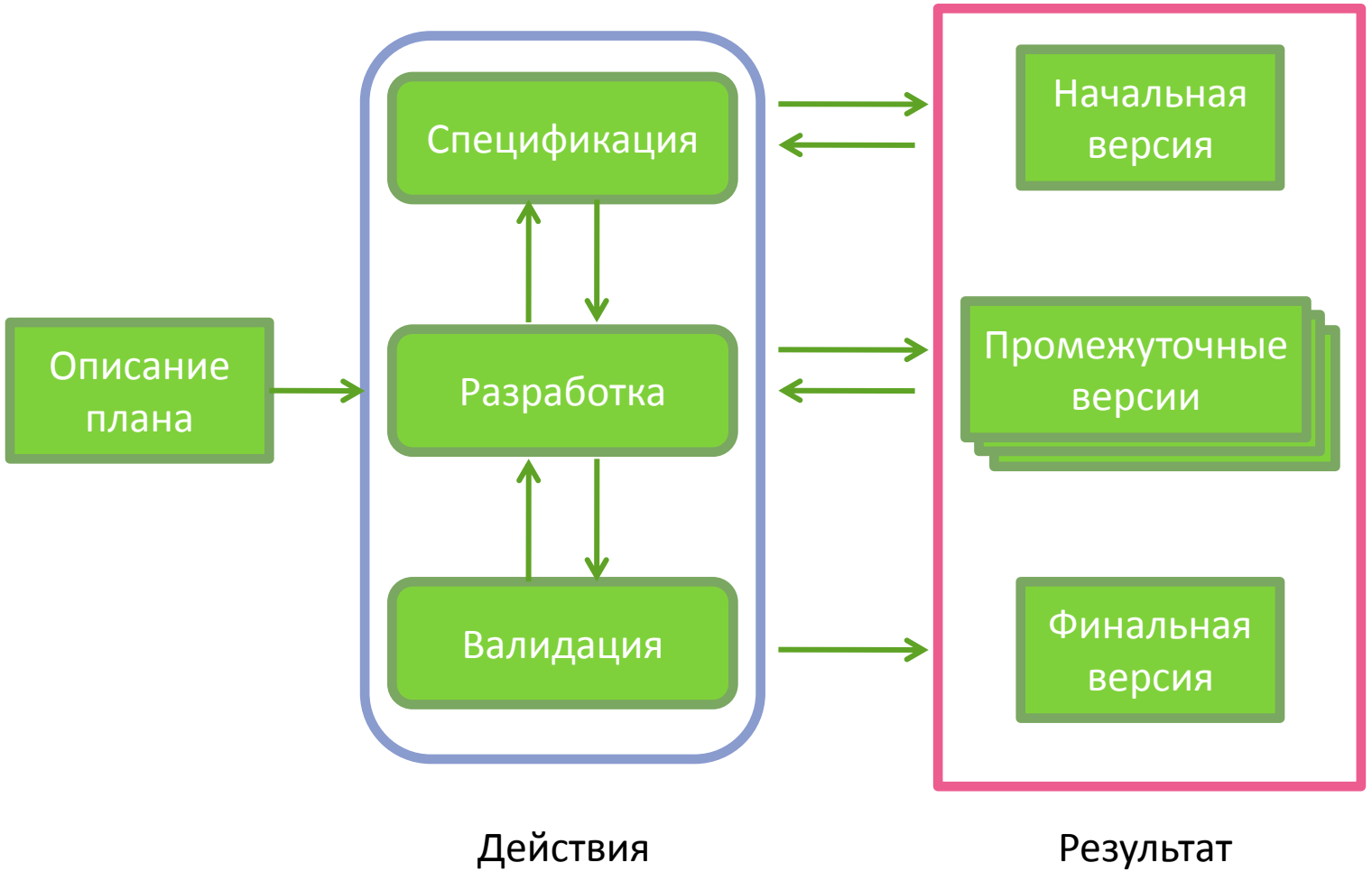
Достоинства:

- Очень формализованная, в результате каждой фазы формируется утвержденный документ
- Соответствует стандартным моделям инженерной разработки.

Недостатки:

- Отсутствие гибкости
- Все договоренности – на ранней стадии, соответственно нет возможности подстроиться под изменяющиеся требования пользователя.

Поэтапная разработка



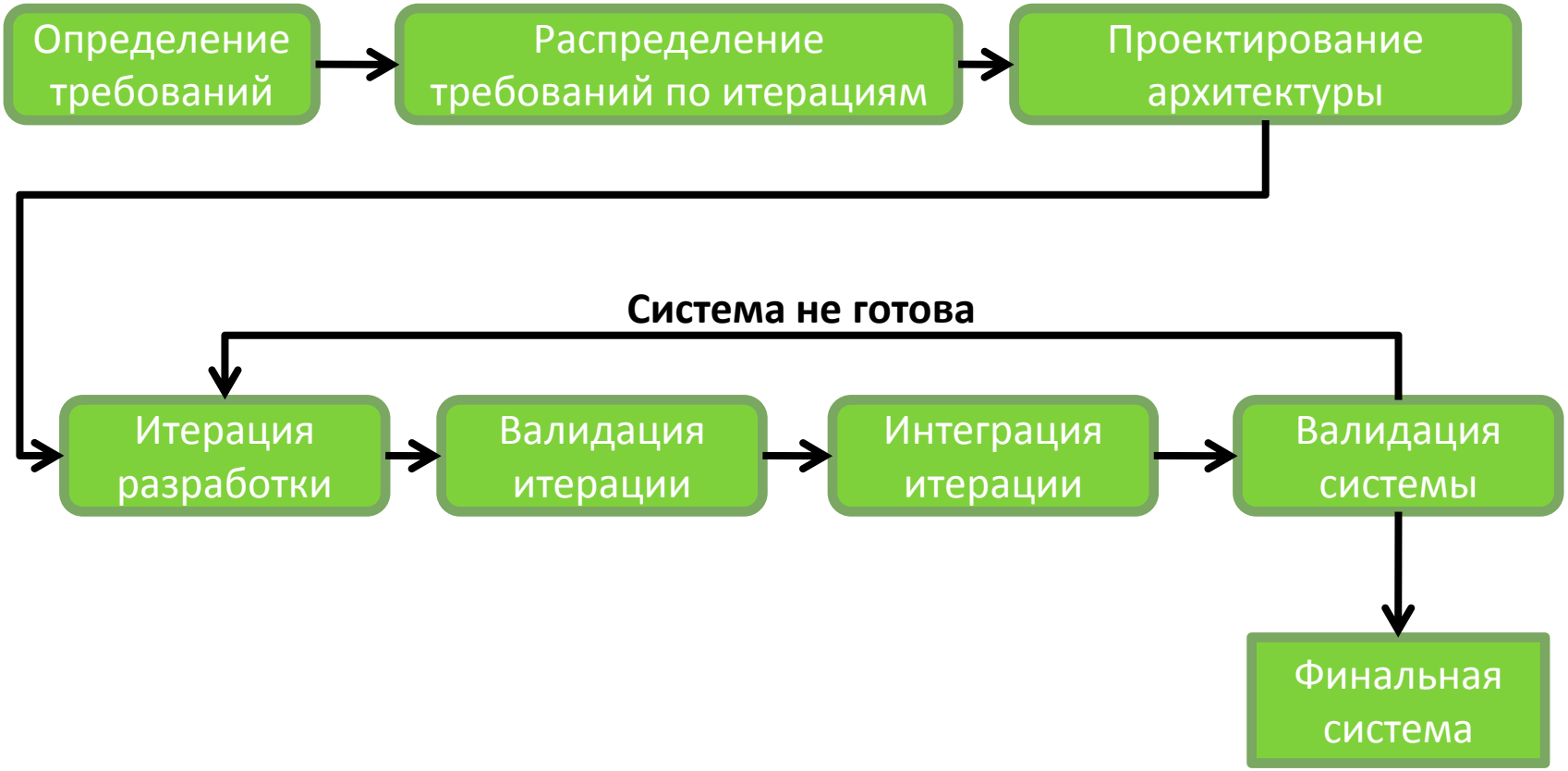
Поэтапная разработка

- ***Постепенная (эволюционная) разработка***: тесная работа с клиентом, для выяснения требований и постепенное создание финальной версии.
  - В начале – те части, которые понятно как делать. Развитие системы: добавление новых возможностей, предлагаемых клиентом.

Поэтапная разработка

- ***Разработка на основе прототипов***: постепенно создаются прототипы разрабатываемой системы, для того чтобы понять конкретные требования заказчика.
  - Как только прототип выполнил свою задачу (требования заказчика стали понятны), он выбрасывается. Его код не используется в конечном продукте.

Инкрементальная разработка



Инкрементальная разработка

- В начале описываются и ранжируются по значимости базовые сервисы, которые должна предоставлять система.
- Определяется количество итераций, за которое должны создать готовую систему.
- В начале каждой итерации формируются детальные требования к набору сервисов, которые будут созданы в ее рамках (дальнейшее их изменение невозможно).

Достоинства

- Не надо ожидать, пока вся система будет полностью готова. Результат первой итерации может быть использован сразу.
- Ранние итерации могут служить прототипами и давать основу для построения требований ко следующим итерациям.
- Наиболее важные сервисы создаются в первую очередь, соответственно обеспечивается их всестороннее тестирование на более поздних этапах.

Недостатки

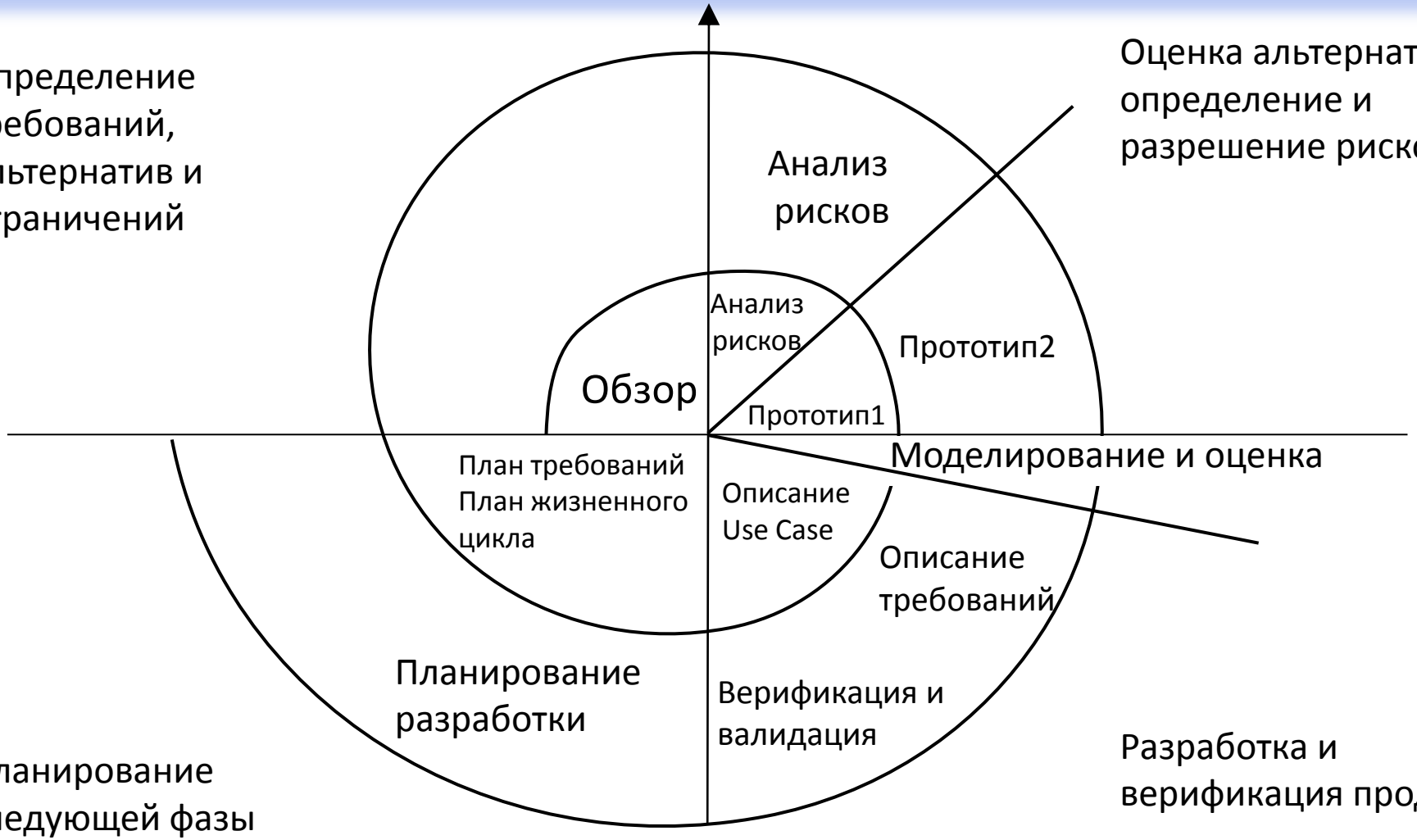
- Необходимо обеспечить малый объем работ в рамках итерации (не более 20 000 строк кода). При этом, должна обеспечиваться определенная функциональность.

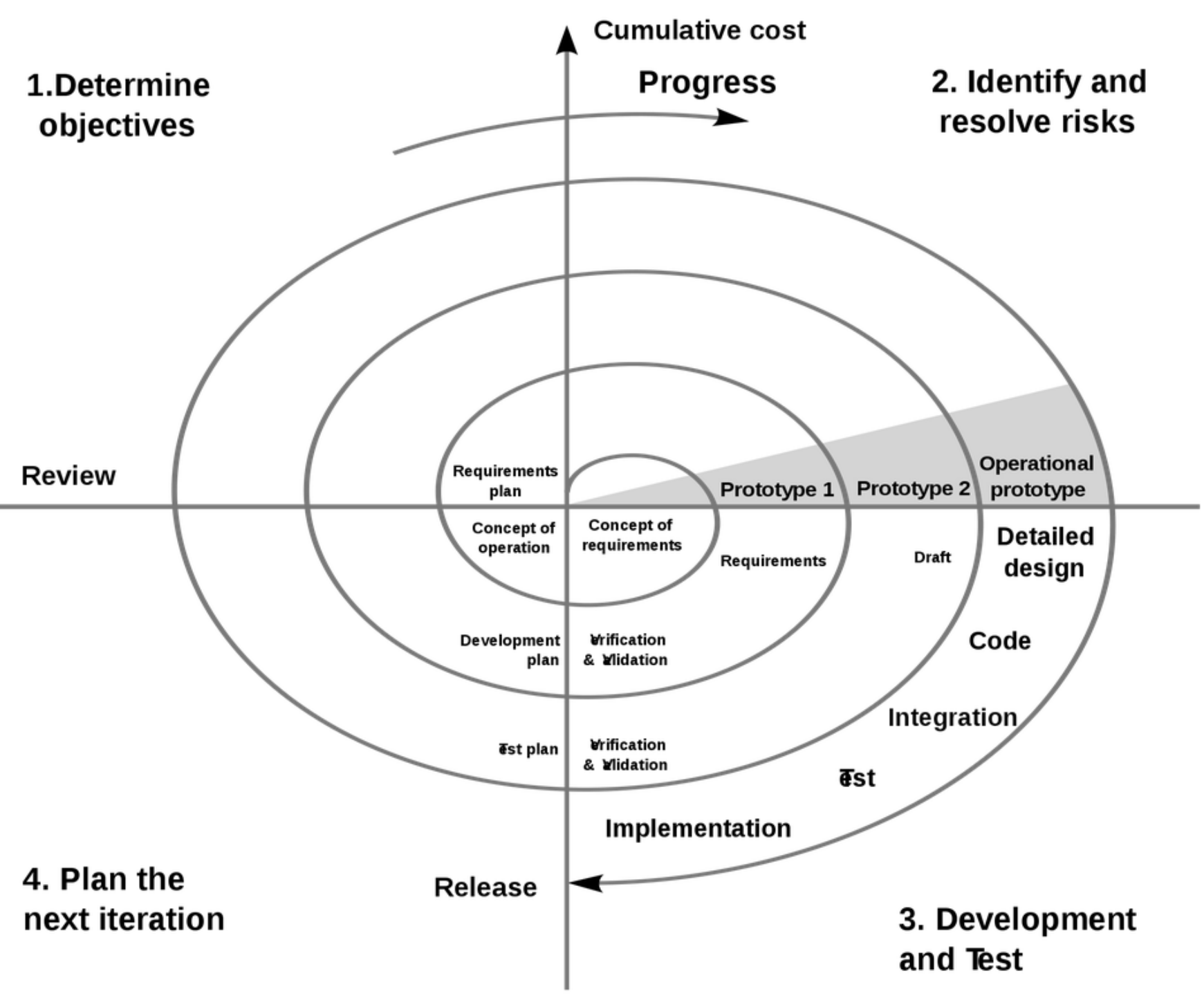
- Но множество систем обладают большим набором базовых сервисов, необходимых всем дальнейшим надстройкам и разделить их на итерации не всегда легко.

Спиральная разработка

- Каждый шаг спирали – фаза процесса разработки ПО (постановка задачи, определение требований, дизайн архитектуры и т.д.)
- В каждом шаге выделяют 4 сектора:
  - Определение требований
  - Оценка и уменьшение рисков
  - Разработка и валидация
  - Планирование

Спиральная разработка




Cumulative cost
Progress
1.Determine objectives
2. Identify and resolve risks
Review
Requirements plan
Prototype 1
Prototype 2
Operational prototype
Concept of operation
Concept of requirements
Requirements
Draft
Detailed design
Development plan
Code
Integration
Test plan
Test
Implementation
Release
4. Plan the next iteration
3. Development and Test

Риски

- **Основное отличие спиральной разработки** от других методов разработки ПО – это явная оценка рисков.
- Риск – это вероятность того, что что-то может пойти не так как хотелось бы (например, при использовании нового языка программирования есть риск, что существующие компиляторы **не** будут создавать высокоэффективный код).

Применение поэтапной разработки

- Поэтапная разработка более гибкая, чем водопадная модель, позволяет легче подстраиваться под изменяющиеся требования заказчика
- Хорошо подходит для небольших и средних по размерам проектов (порядка 500 000 строк кода)

Итерации позволяют

- контролировать и корректировать ход выполнения проекта
- эффективнее работать с изменяющимися требованиями;
- эффективнее работать с рисками
- на ранних этапах оценивать потенциальные характеристики системы

Проблемы поэтапной разработки

- **Процесс разработки не виден**
  - Ради скорости разработки в жертву приносится формальность: практически отсутствует документация, производимая на каждом этапе водопадной модели
- **Системы часто слабоструктурированы**
  - Постоянные изменения приводят к повреждению структуры ПО. Поддержка и дальнейшее изменение становится дорогой и сложной процедурой.

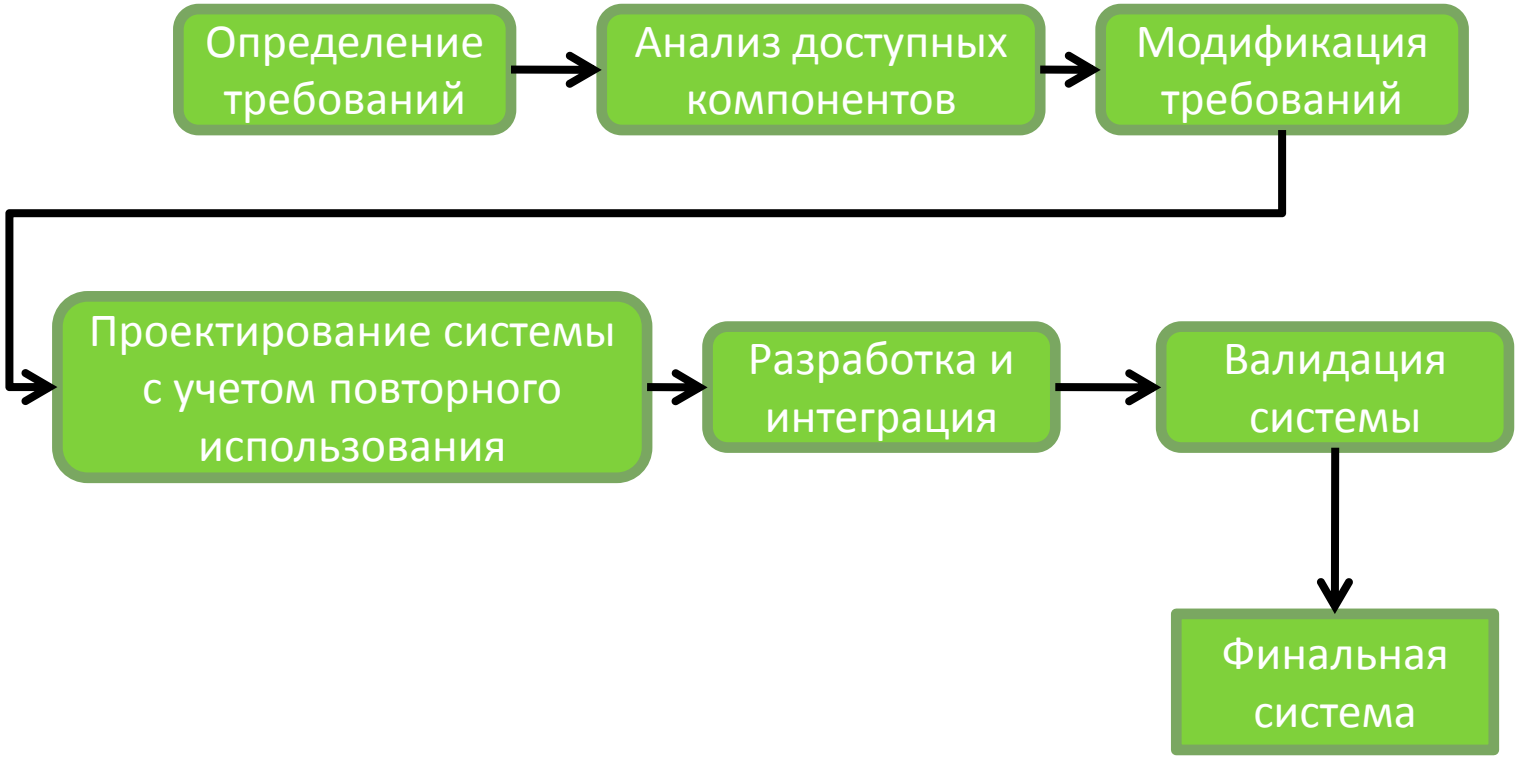
Компонентно-ориентированная разработка

- Компонентно-ориентированная разработка основывается на формальном закреплении повторного использования кода.
- **Программный компонент** – это автономный элемент программного обеспечения, предназначенный для *многократного использования*, который может распространяться для использования в других программах в виде скомпилированного кода.

Компонентно-ориентированная разработка

- Компонентно-ориентированный подход – развитие объектно-ориентированного. Создан для проектирования и реализации *крупных и распределенных программных систем (корпоративных приложений)*
- ***С точки зрения КОП*** программная система – это набор компонентов с четко определенным интерфейсом.
  - Изменения в систему вносятся путем создания новых компонентов или изменения старых.
  - **Наследование реализации запрещено. Наследуется только интерфейс.**

Компонентно-ориентированная разработка



Достоинства и недостатки компонентно-ориентированной разработки

- Достоинства
  - Уменьшается объем ПО, которое необходимо разработать => уменьшается цена и риски.
  - Уменьшается время разработки
- Недостатки
  - Компромиссы при выработке требований могут привести к тому, что реальные требования пользователей не будут учтены.
  - Контроль над системой может быть потерян при появлении новых версий используемых компонентов

Сколько формализма необходимо?

- С 1970 по 1995 считалось, что чем тщательней оформлена документация, тем лучше
- Далее перешли на «компактность» документации: диаграммы и схемы
- Но может быть лучше пусть будет специальный человек, который знает всю документацию и может объяснить?

Факторы, влияющие на оптимальный уровень формализма

- Масштаб проекта
- Критичность проекта
- Распределение участников
- Новизна проекта
- Требования заказчика
- Ожидаемая долговечность проекта

Вопросы

- Программная инженерия - это дисциплина, отражающая все грани разработки программного продукта рамках существующих **…**
- Качественный программный продукт требует в **…** раз больше трудозатрат чем программа с тем же функционалом
- Процесс разработки ПО – это…
- Идеальный процесс разработки ПО – это …

Вопросы

- Мы с вами разобрали 3 модели разработки ПО. Опишите эти модели:
  - Название
  - Достоинства
  - Недостатки

Вопросы

- Какую модель разработки вы выберете для каждой предложенной системы и почему:
  - Игрушка для iPhone;
  - Корпоративное приложение для работы с кадрами предприятия;
  - ПО для управления рентгеновским аппаратом;
  - Драйвер для Microsoft Kinect

Вопросы

- Опишите основные критерии, определяющие уровень формализма, необходимый для разработки ПО

Резюме

- ***Программная инженерия*** – это дисциплина, отражающая все грани разработки программного продукта рамках существующих *организационных*, *финансовых* и *временных* ***ограничений.***
- Качественный программный продукт требует в 10 раз больше трудозатрат чем программа с тем же функционалом

Резюме

- ***Процесс разработки ПО (жизненный цикл ПО)*** – это набор действий и связанных с ними результатов, направленных на разработку и/или развитие программного продукта**.**

- Идеального процесса разработки не существует!

Резюме

- Можно выделить 3 класса моделей разработки ПО:
  - Водопадная модель:
    - самая первая;
    - самая формальная;
  - Модель поэтапной разработки:
    - самая гибкая;
    - результат – после первой итерации;
  - Компонентно-ориентированная модель:
    - повторное использование кода;
    - ускоренная разработка.