Обзор моделей

и одной методологии

разработки ПО

Лекция 2. Программная инженерия

Управление программными проектами

Основные понятия программной инженерии

Продукт = Программная система

Основными составляющими программной инженерии являются

- продукты,
- процессы, обеспечивающие создание продуктов

Ключевой параметр продукта — сложность

Программа ≠ ПО

ГОСТ 19781-90 и ISO IEC 2382/1-93

А что же тогда такое «программный продукт»?

В чем отличие от «программы» или «ПО»?

А что же тогда такое «программный **проект**»?

Программный проект (project) - это временное предприятие, предназначенное для создания уникальных продуктов, услуг или результатов.

N.B.

Проектирование vs. Design

FYI. Трудности перевода

Проектирование — это один из шагов деятельности программного проекта и работа на этом шаге (создание эскизов концептуального решения требуемого, представляемого обычно с помощью рисунков и пояснения к ним)
В английской терминологии - design.

А что входит в состав *программного проекта?*

Программная инженерия

ПИ или SE

Программная инженерия или инженерия ПО (software engineering) впервые был использован в 1968 году

ПИ характеризовалось как система инженерных принципов для создания экономичного ПО, которое надежно и эффективно работает в реальных компьютерах.

Более современное определение

ISO/IEC 2382/1-93 дает более развернутую формулировку:

систематическое применение научных и технологических знаний, методов и практического опыта к проектированию, реализации, тестированию и документированию ПО в целях оптимизации его производства, поддержки и качества

ПИ обеспечивает управляемый, комплексный подход к созданию сложного программного продукта командой разработки.

Эволюция

Эволюция подходов в программной инженерии:

- классический, процедурный (70-80 годы)
- объектно-ориентированный;

Программная инженерия состоит из

- Методов
- Средств
- Процессов

Методы

- планирование и оценка программного проекта;
- анализ требований к компьютерной системе в целом и к программному обеспечению в частности;
- проектирование структур программ (и структур данных), входящих в состав ПО;
- конструирование программного текста (кодирование, программирование);
- тестирование (выявление ошибок в созданных программах);
- сопровождение ПО уже используемого заказчиками.

Средства

Для удобства они объединяются в системы автоматизированной разработки ПО. Такие системы называется CASE-системами (Computer Aided Software Engineering - программная инженерия с компьютерной поддержкой).

Примеры CASE-систем:

- инструменты редактирования программного кода;
- генераторы кода;

Процессы

Соединяют методы и средства, обеспечивая непрерывную технологическую цепочку разработки.

Определяют:

- порядок применения методов и утилит;

- формирование отчетов, форм;

- контроль, обеспечивающий координацию изменений;

- формирование “вех”, необходимых для оценки прогресса.

Классификация процессов ПИ

Определяется:

- международным стандартом ISO/IEC 12207-95 “Information Technology - Software Life Cycle Processes”
- ГОСТ Р ИСО/МЭК 12207-99

Связывается с понятием жизненного цикла программного обеспечения.

Жизненный цикл ПО определятся как период времени, который начинается с момента принятия решения о необходимости создания ПО и заканчивается в момент его полного изъятия из эксплуатации

Работы в ЖЦ ПО распределяются по:
5 основным,
8 вспомогательным и
4 организационным процессам.

Основные процессы

Всего 5:

1. Заказ.
2. Поставка.
3. Разработка.
4. Эксплуатация.
5. Сопровождение.

Вспомогательные процессы

Всего 8:

1. Документирование.
2. Управление конфигурацией.
3. Обеспечение качества.
4. Верификация.
5. Аттестация.
6. Совместная проверка.
7. Аудит.
8. Решение проблемы.

Организационные процессы

Всего 4:

1. Управления.
2. Создания инфраструктуры.
3. Усовершенствования.
4. Обучения.

Базис процессов

Основывается на трех основных сущностях:

1. Деятельность.

2. Действие.

3. Задача.

Процесс разработки должен быть адаптированным для конкретного проекта, быть гибким и для другого проекта может не повториться в том же виде.

Цель - создавать ПО за приемлемое время и с достаточным качеством, удовлетворяющим заказчика.

Базис по Прессману

1. Подготовка.
2. Планирование.
3. Моделирование.
4. Конструирование.
5. Развертывание.

Виды защитной деятельности

1. Отслеживание (трассировка).
2. Управление риском.
3. Обеспечение качества ПО.
4. Технические проверки.
5. Измерение.
6. Управление конфигурацией.
7. Управление повторной используемостью.
8. Подготовка и производство рабочего продукта.

Невыносимая *гибкость* бытия

Модель разработки

Описание подхода к структуре, согласно которой будет разрабатываться программное обеспечение (ПО) в виде задач и/или деятельности, которые имеют место в ходе процесса.

Методология

Это система принципов, а также совокупность идей, понятий, методов, способов и средств, определяющих стиль разработки программного обеспечения

Примеры моделей

- итеративная
- спиральная
- каскадная
- v-model
- dual vee model

Примеры методологий

- Agile (XP, Lean, Scrum, FDD)
- RUP
- MSF
- RAD
- TDD, BDD
- Cleanroom
- OpenUP

Процесс разработки

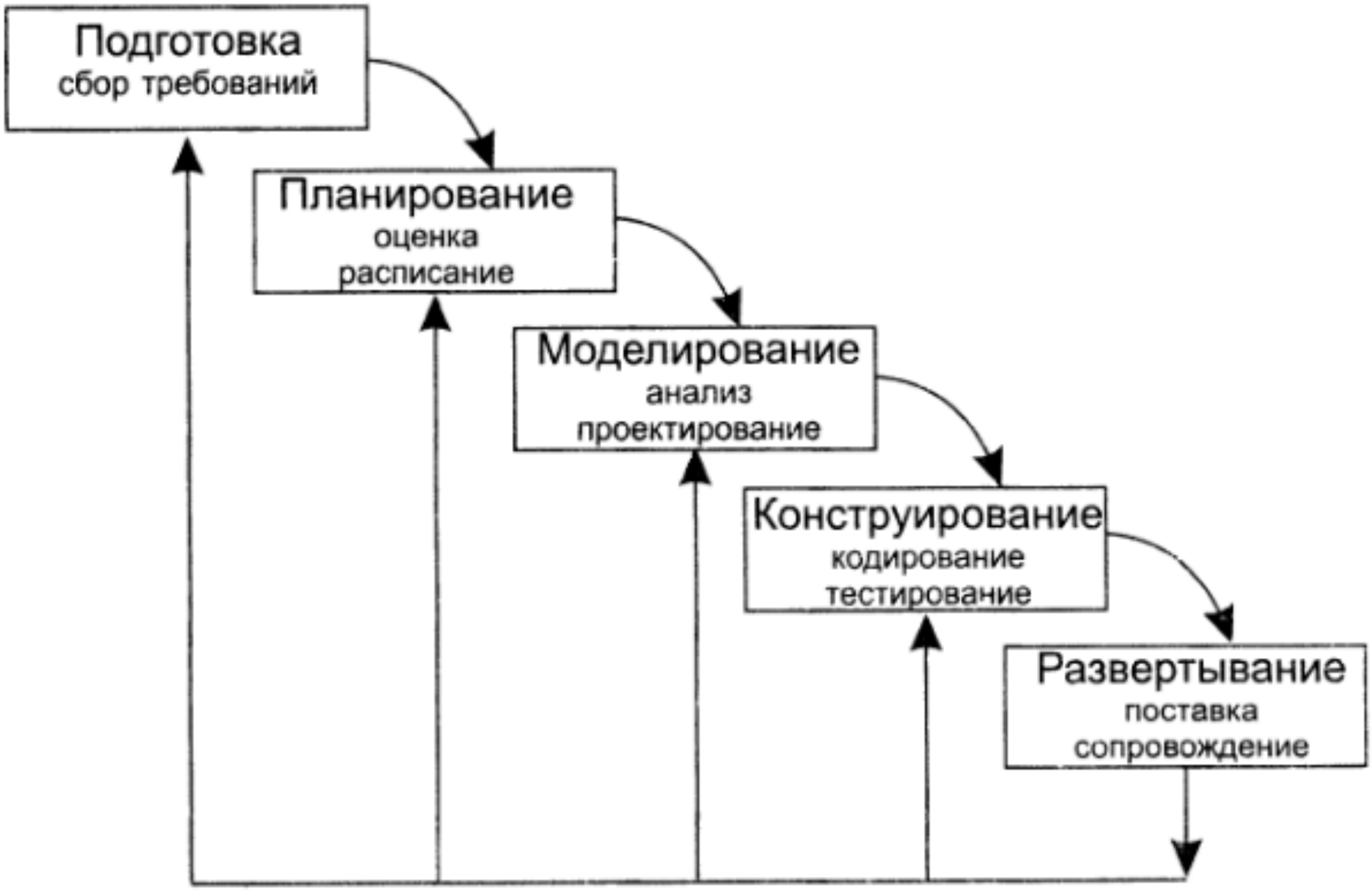
Вспомним стадии разработки ПО:

- анализ требований
- проектирование
- программирование
- тестирование
- документирование

Каскадная модель

Модель «Водопад». Водопадная модель.
Waterfall model

Каскадная модель. Схема



Каскадная модель. Этапы

Каскадная модель:

- Формирование требований.
  - Проектирование.
    - Реализация.
      - Тестирование (верификация).
        - Внедрение.
          - Эксплуатация и сопровождение.

Каскадная модель. Особенности

- последовательное выполнение стадий разработки (строгая очередность);

- невозможность перехода к последующей стадии без полного выполнения текущей;

- требования к продукту четко зафиксированы на всё время выполнения проекта;

- каждая стадия завершается выпуском полного набора документации (позволяющей продолжать/ передавать проект другим разработчикам);

Каскадная модель. Особенности

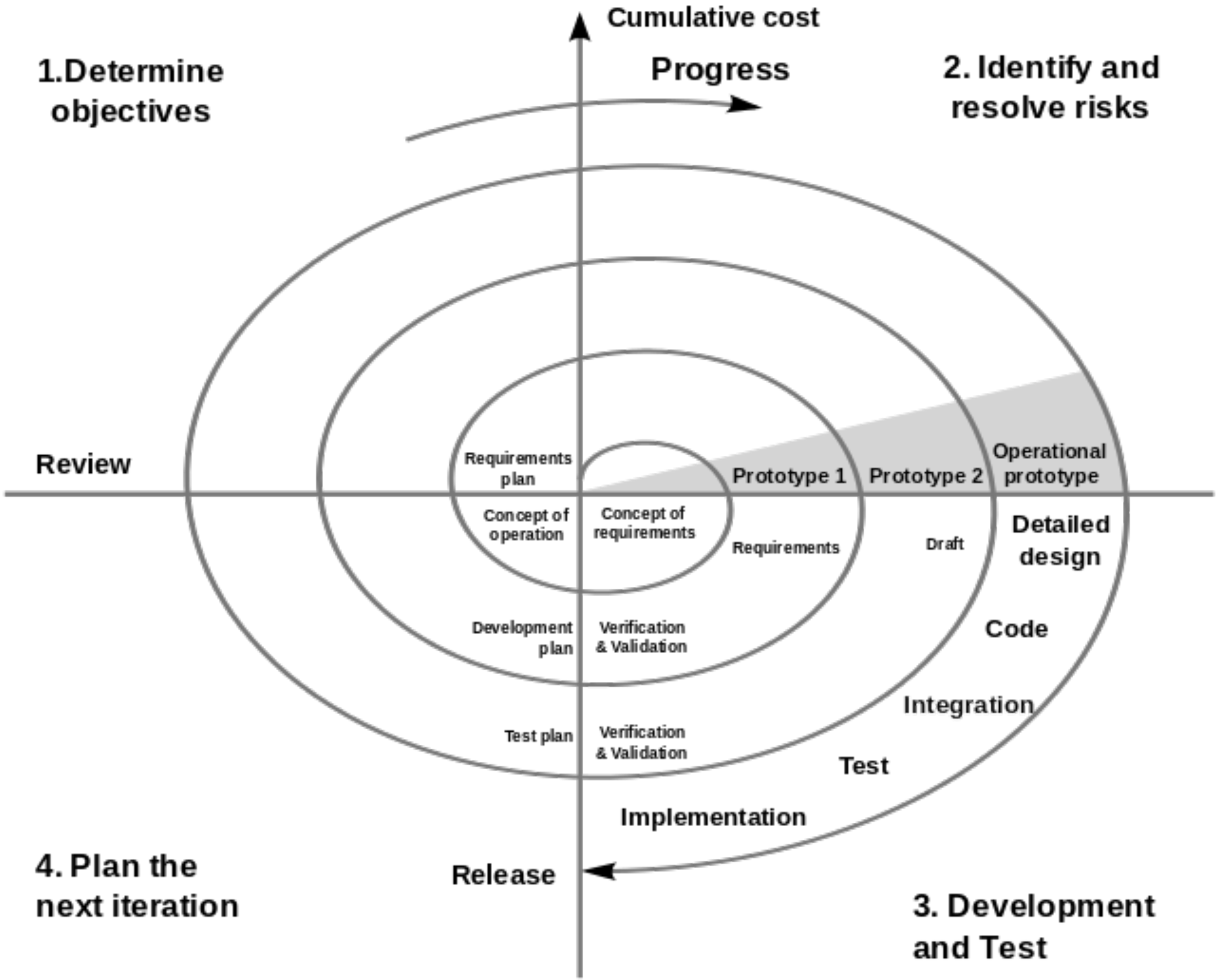
- невозможность перехода назад без последствий;
- полная согласованность документации на каждом этапе выполнения проекта;
- легко определить срок и затраты на проект;
- большая «стоимость» неточности в требованиях приводит к отбрасыванию на предыдущие этапы (результаты: срыв сроков, возрастание стоимости проекта и.д.);

Каскадная модель. Особенности

- Хорошо подходит для небольших проектов (низкие риски).
- Чем больше проект, тем больше риски.
- Результат - только в конце работы.

Спиральная модель

промежуточная модель между итеративной и каскадной моделью


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
Verification & Validation
Code
Integration
Test plan
Verification & Validation
Test
Implementation
Release
4. Plan the next iteration
3. Development and Test

Спиральная модель

- Сочетает в себе итеративность и этапность работы.

- Упор на риски, возникающие при реализации продукта.

- На **каждом витке** может применяться своя модель разработки;

Риски в спиральной модели

К рискам в спиральной модели относят:

1. Дефицит специалистов.
2. Нереалистичные сроки и бюджет.
3. Реализация несоответствующей функциональности.
4. Разработка неправильного пользовательского интерфейса.
5. «Золотая сервировка», перфекционизм, ненужная оптимизация и оттачивание деталей.

Риски в спиральной модели

6. Непрекращающийся поток изменений.

7. Нехватка информации о внешних компонентах, определяющих окружение системы или вовлечённых в интеграцию.

8. Недостатки в работах, выполняемых внешними (по отношению к проекту) ресурсами.

9. Недостаточная производительность получаемой системы.

10. Разрыв между квалификацией специалистов и требованиями проекта

Каждый виток спирали — новый фрагмент или версия ПО.

На каждом витке уточняются задачи (возможно, цель), характеристики проекта, определяется качество, планируются работы следующего витка.

Постепенное углубление, последовательная конкретизация деталей.

На последних витках — обоснованный, «отточенный» вариант, в большей степени реализованный.

Этапы спиральной модели

- определение целей;
- оценка и разрешение рисков;
- разработка и тестирование;
- планирование следующей итерации.

Особенности

- Неполное завершение работ на каждом этапе позволяет переходить на следующий этап, не дожидаясь полного завершения работы на текущем.

- Эту работу можно будет выполнить на следующей итерации.

- Для успешной реализации, требуется вводить временные ограничения на каждый из этапов жизненно цикла разработки.

- Переход к следующему этапу осуществляется, даже если текущий этап незавершен (*как составляется план?*).

На каждой итерации оцениваются

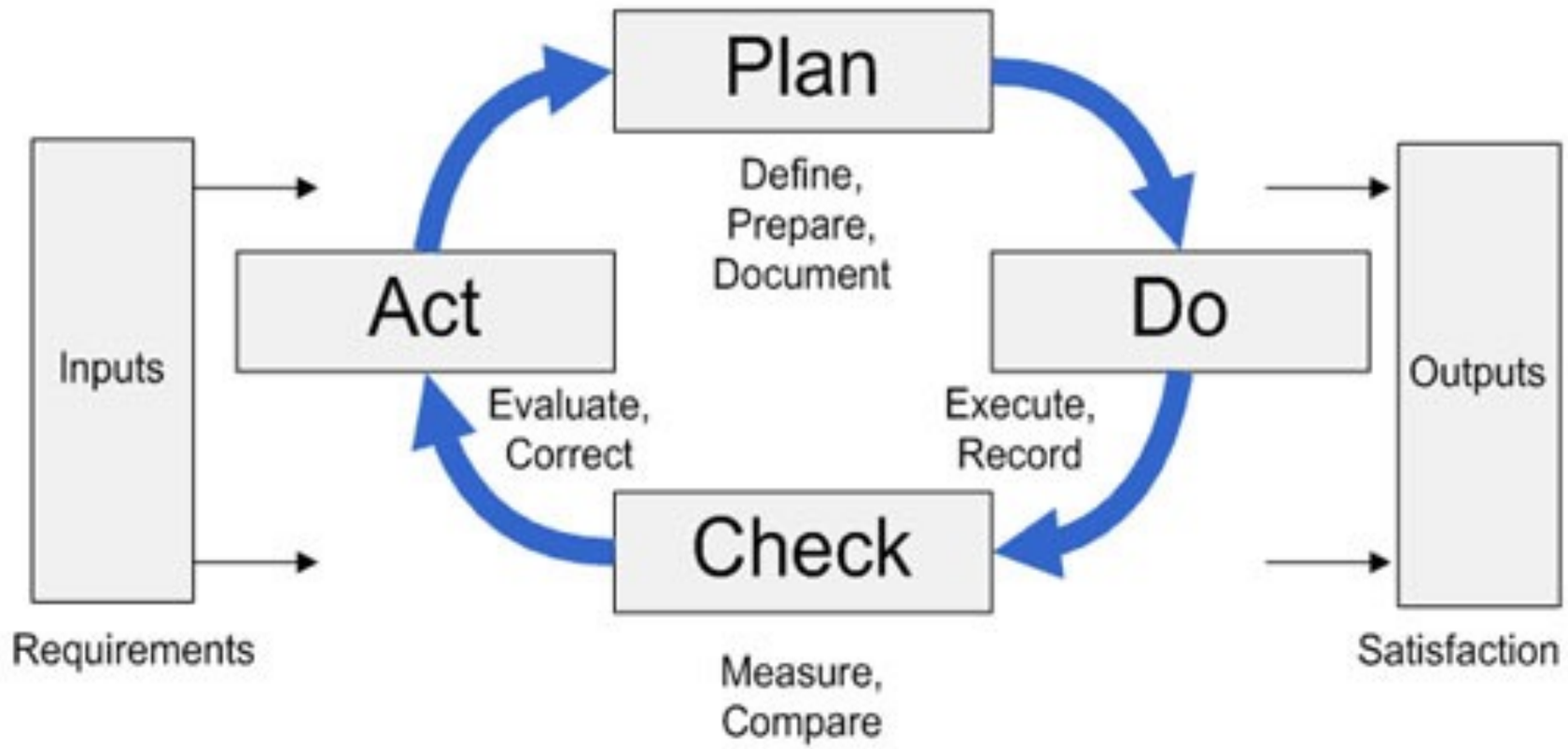
- риск превышения сроков и стоимости проекта;
- необходимость выполнения ещё одной итерации;
- степень полноты и точности понимания требований к системе;
- целесообразность прекращения проекта.

Итеративная модель

Итеративный подход к разработке ПО

Итеративная разработка

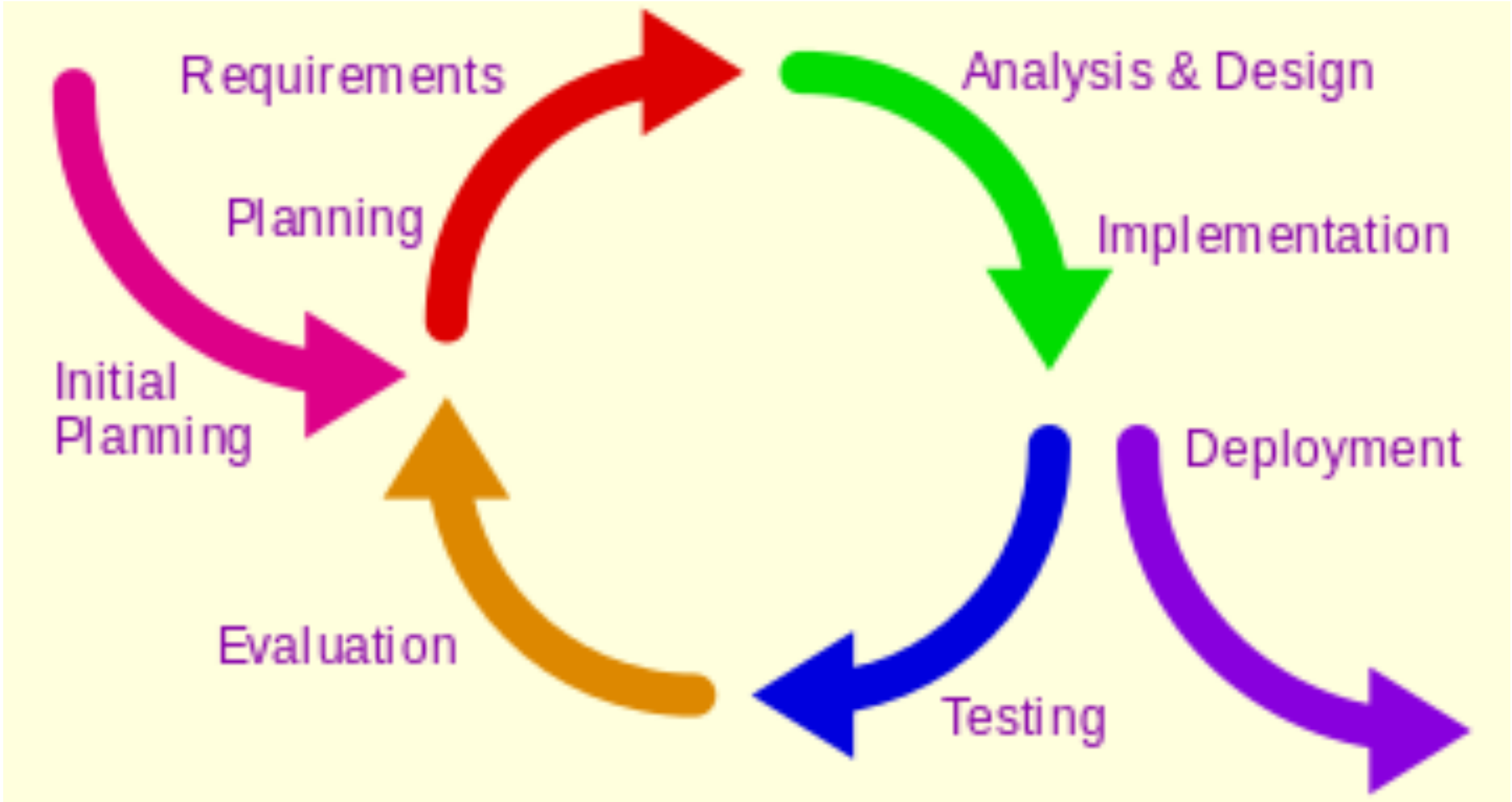
Выполнение работы над проектом с непрерывным и асинхронным анализом полученных результатов и корректировкой предыдущих этапов работы (PDCA).



Разработка проекта делится на фазы, каждая фаза включает:

- планирование (plan) / Requirements;
- реализацию (do) / Analysis, Design, Implementation;
- проверку (check) /Testing, Verification;
- оценку (act) / Evaluation.

Перед фазами идет первоначальное планирование, (Initial planning), после всего идет — внедрение (deployment).


Requirements
Analysis & Design
Planning
Implementation
Initial
Planning
Deployment
Evaluation
Testing

Итеративная модель. Особенности

- минимизация затрат для рисков (особенно на ранних этапах проекта);

- эффективная обратная связь с заказчиками, более динамичное обнаружение нестыковок требований и реализованного в продукте;

- фокусировка на наиболее приоритетные направления в развитии продукта;

- итеративное тестирование и проверка, позволяют выпускать более качественный продукт;

Итеративная модель. Особенности

- повторяемость фаз позволяет накапливать опыт;
- мониторинг реализации продукта заказчиком в реальном времени (заказчику видна реальное состояние разработки продукта);
- равномерное распределение по всему процессу реализации проекта (а не сосредотачивание затрат в его начале или конце).

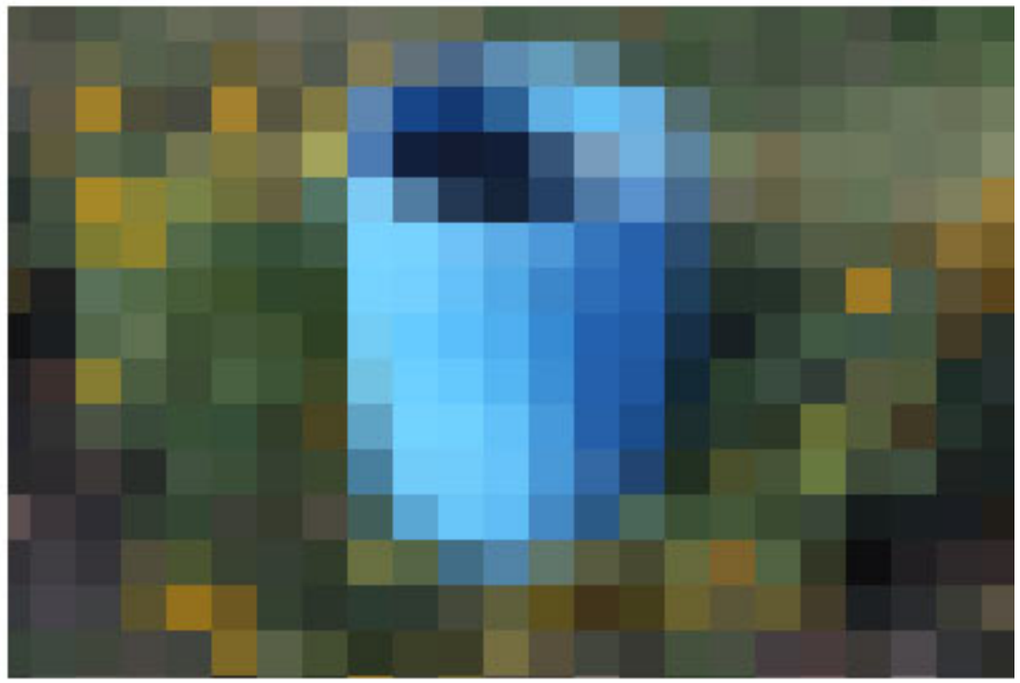
Метод прогрессивного джипега

составлен Артемием Лебедевым

Обычный джипег

30% выполнения

Прогрессивный джипег



70% выполнения