Software Engineering

Methods and Tools
to
build the right product & build it right

软件工程

方法与工具
to
构建正确的产品，并正确地构建它

Software Life Cycle

Requirements Phase

What to develop? Involves requirements elicitation activities.

Output of this phase are the Requirements Specs & Test Cases

Design Phase

How to develop? A blueprint to translate specs into code is created.

Output of this phase are the Architecture & Detail Design

Implementation Phase

Product Manifestation takes place.

Output of the phase includes Code, Unit Test Results & Documentation

Testing & QA

Verification (right product) & Validation (build right) takes place

QA & Test Results are captures

Maintenance

Bug reports are handled and above phases are repeated to add new features

软件生命周期

需求阶段

要开发什么？涉及需求获取活动。

该阶段的输出是需求规格说明书和测试用例

设计阶段

如何开发？创建将规格说明转化为代码的蓝图。

该阶段的输出为架构设计和详细设计。

实施阶段

产品实现在此阶段进行。

该阶段的输出包括代码、单元测试结果和文档。

测试与质量保证

在此阶段进行确认（正确的产品）和验证（正确地构建）。

QA 和测试结果已捕获

维护

处理缺陷报告，并重复上述阶段以添加新功能

Common patterns of software life cycle are referred to as Software Process Models.

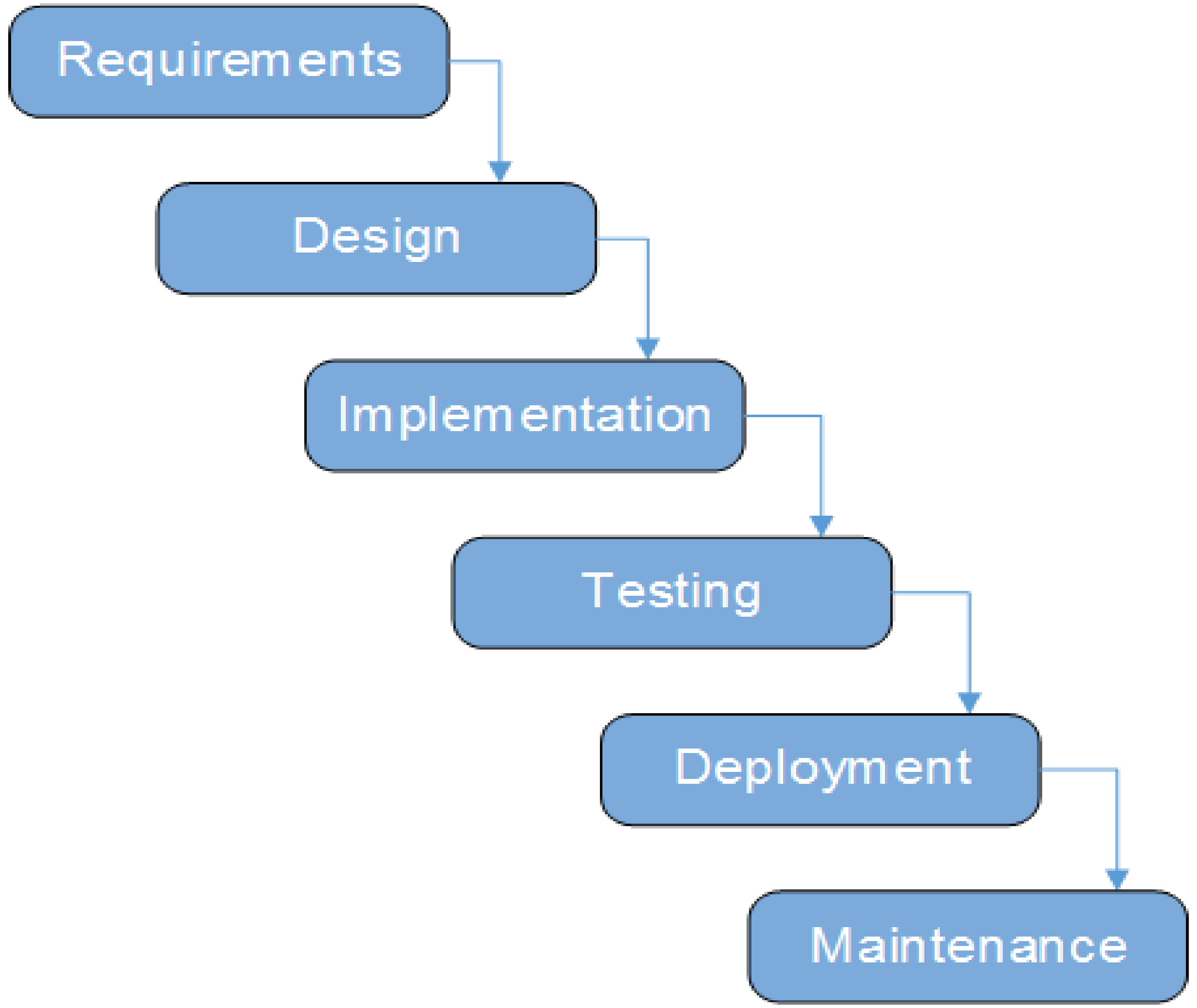
软件生命周期的常见模式被称为软件过程模型。

Several such process models have been identified or defined.

已经识别或定义了多种此类过程模型。

Software Process Models

WaterFall



Requirements

Design

Implementation

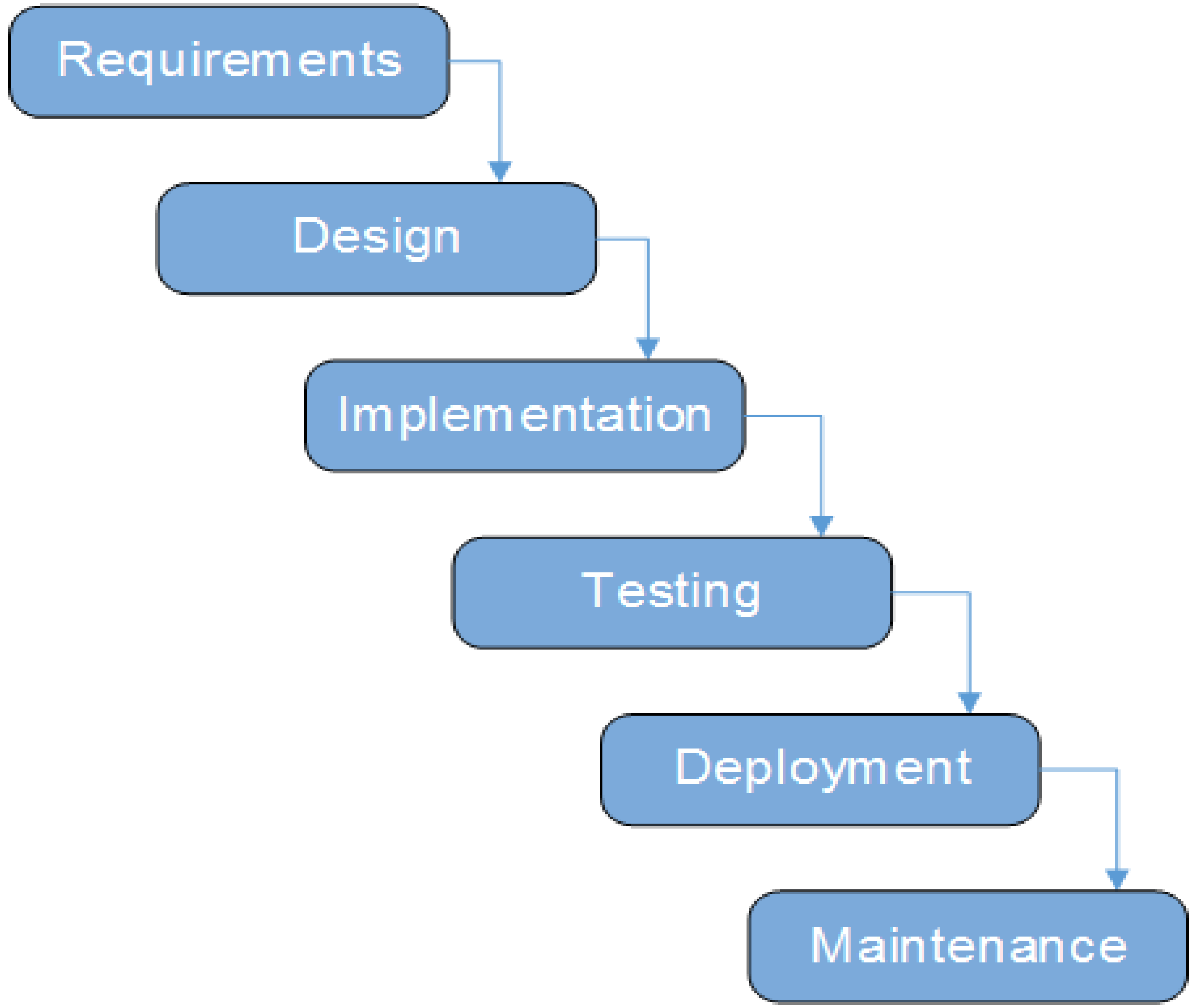
Testing

Deployment

Maintenance

软件过程模型

瀑布模型



Requirements

Design

Implementation

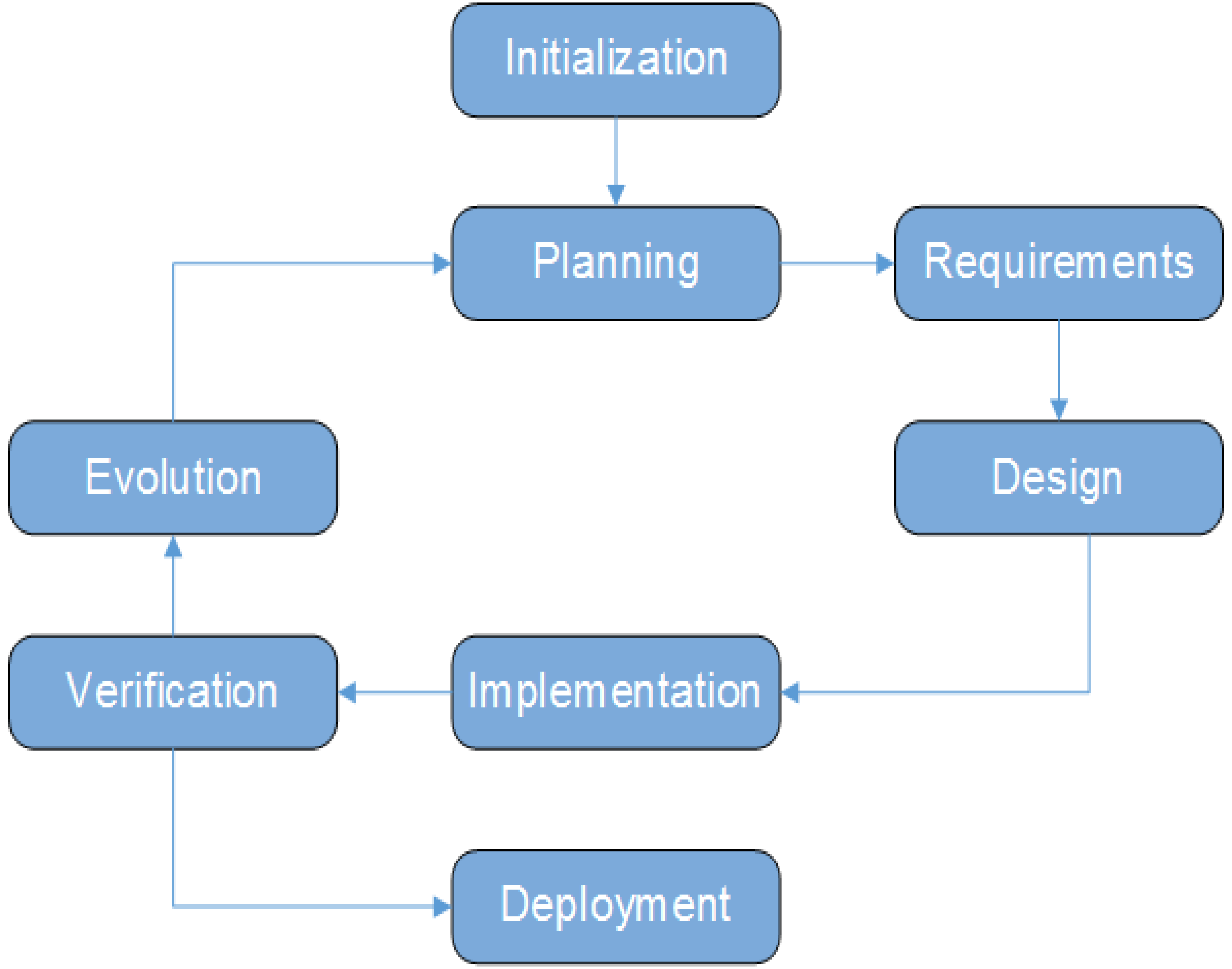
Testing

Deployment

Maintenance

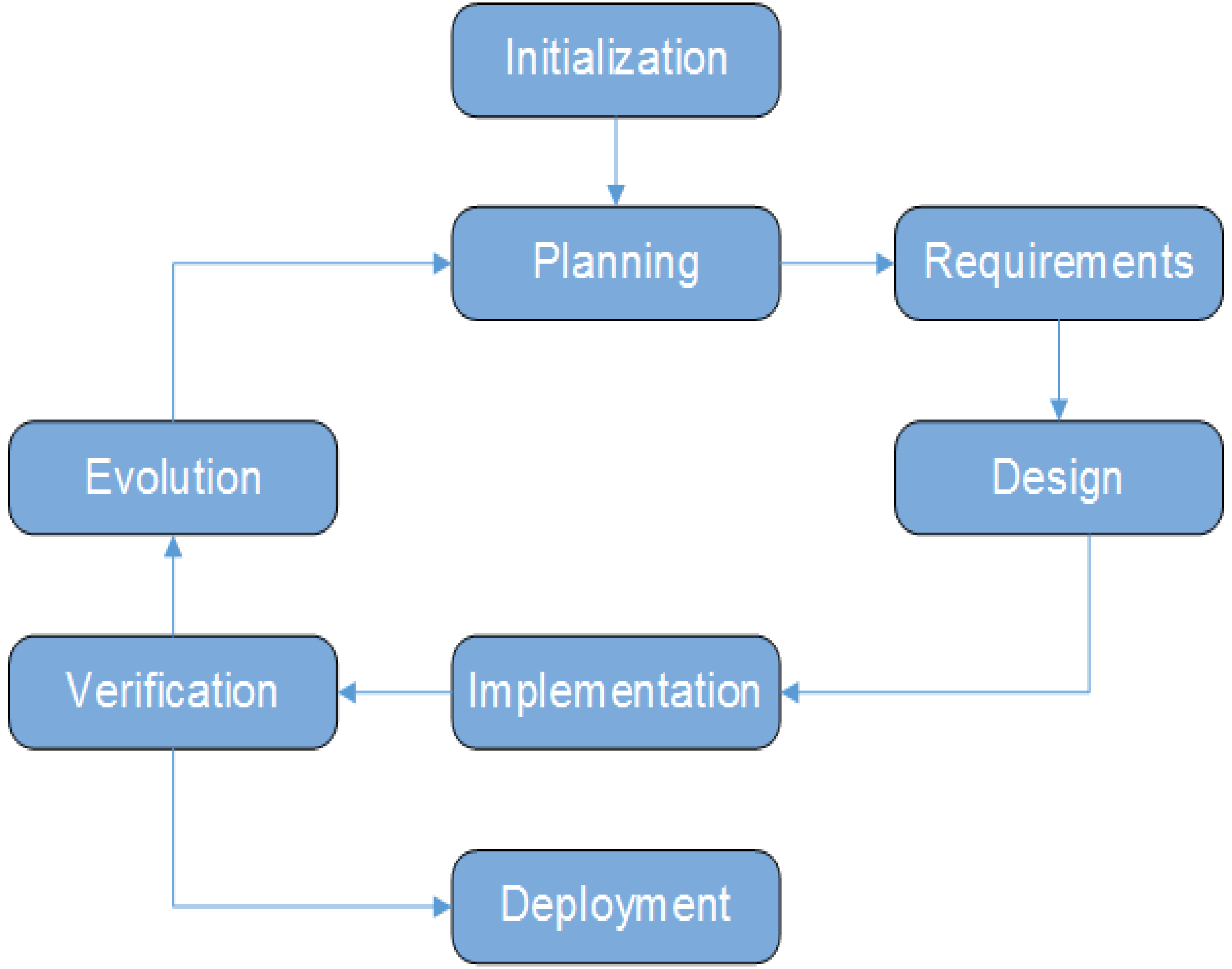
Software Process Models

Iterative

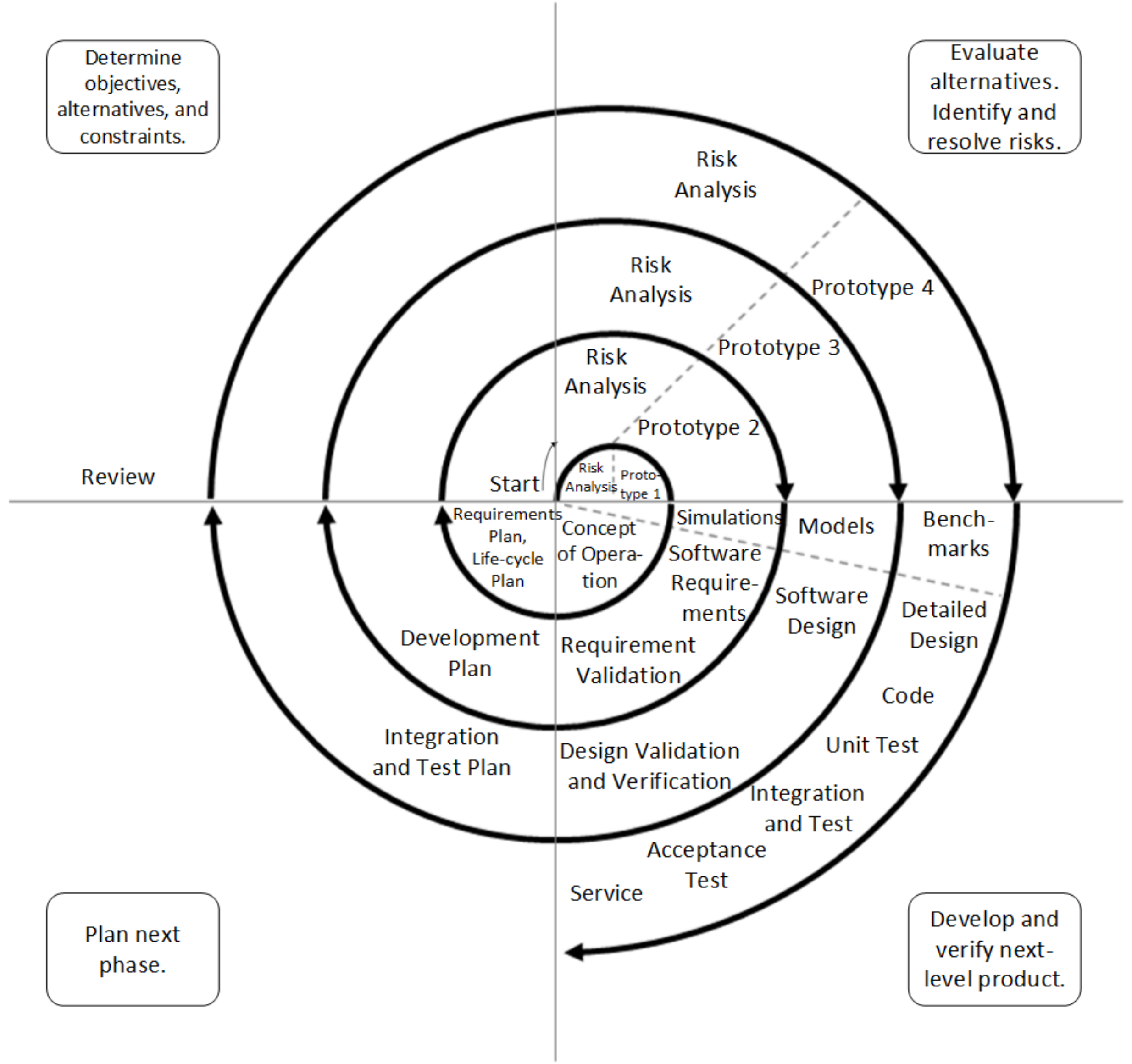


软件过程模型

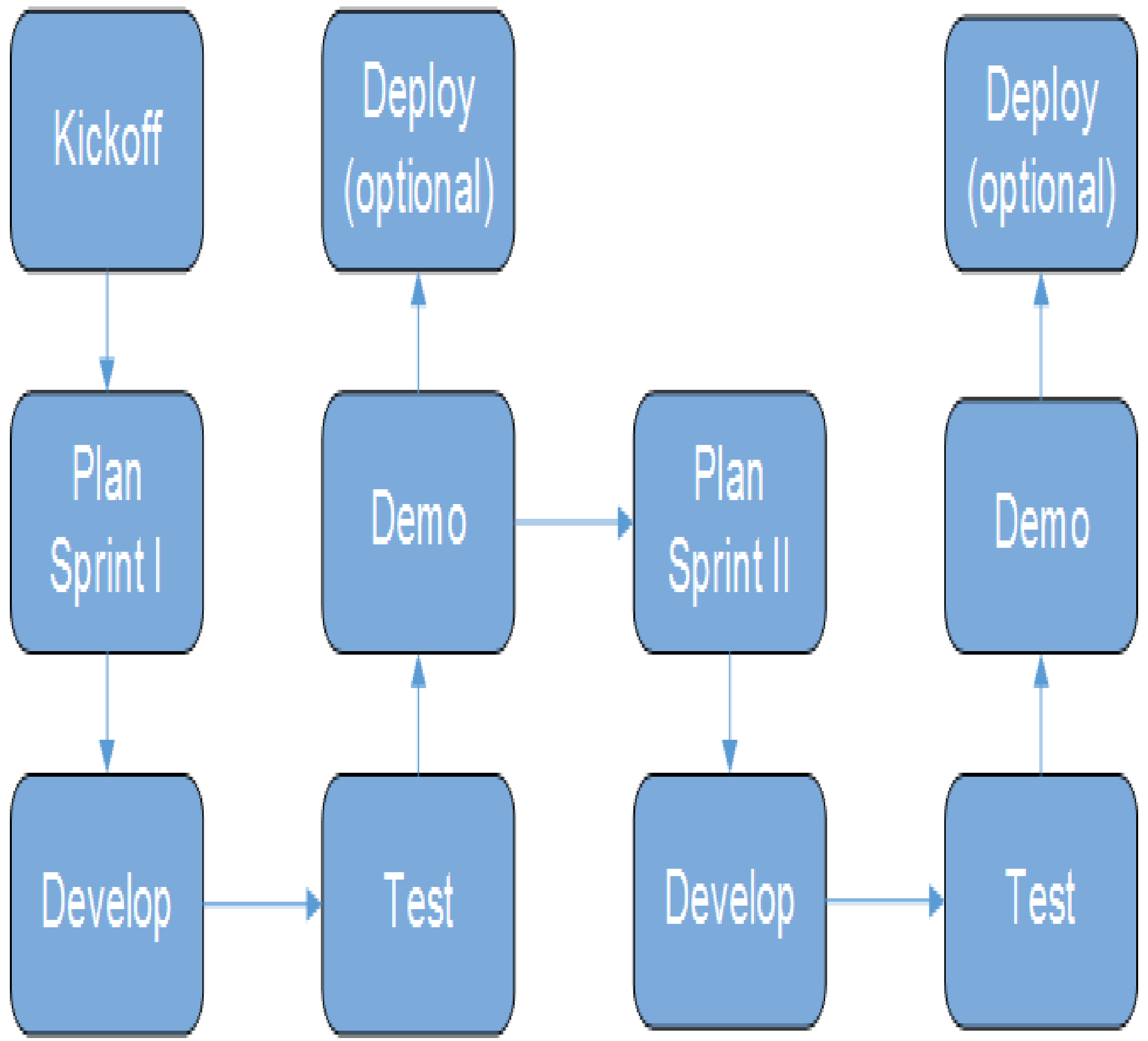
迭代



Software Process Models

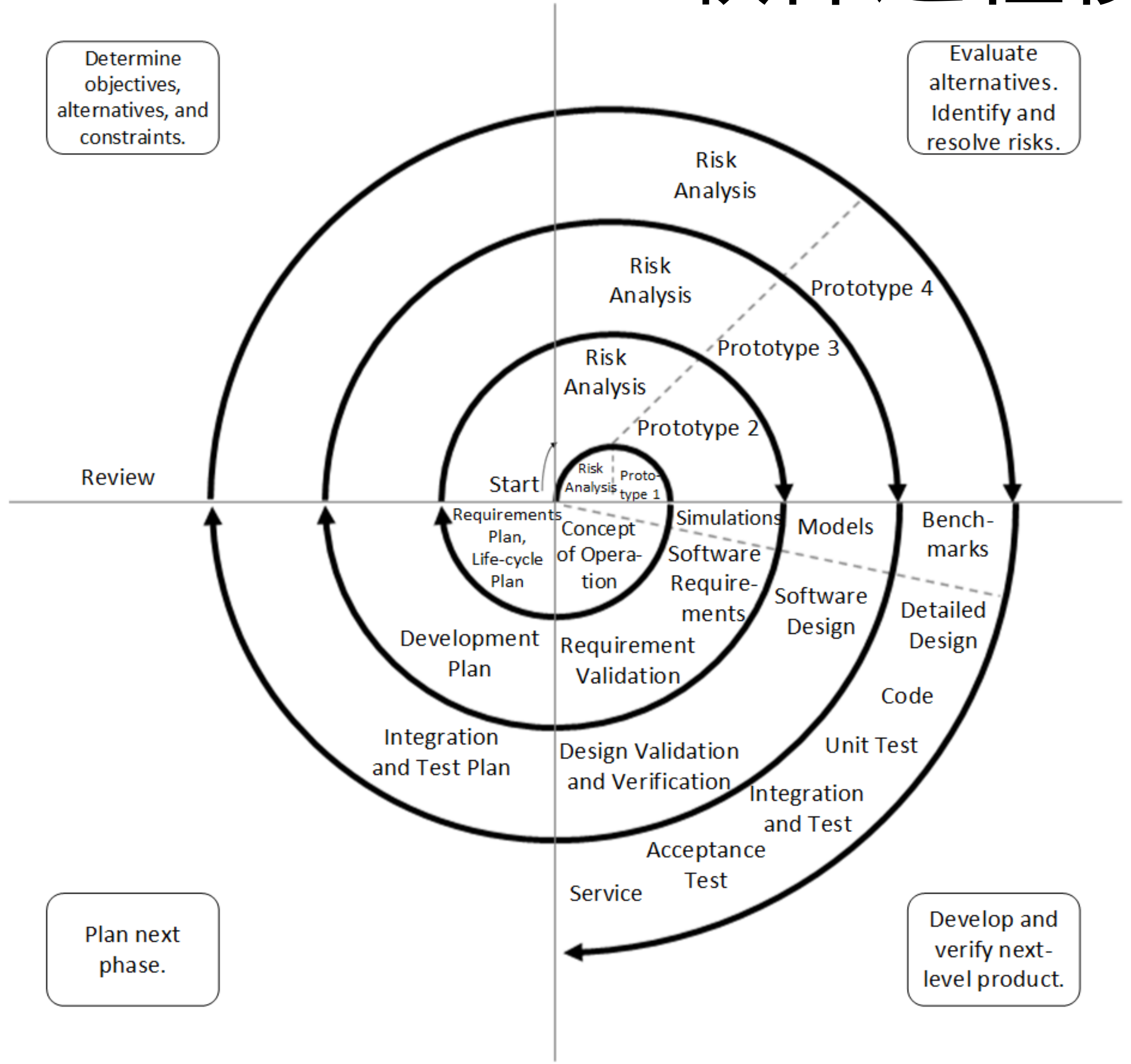


Spiral

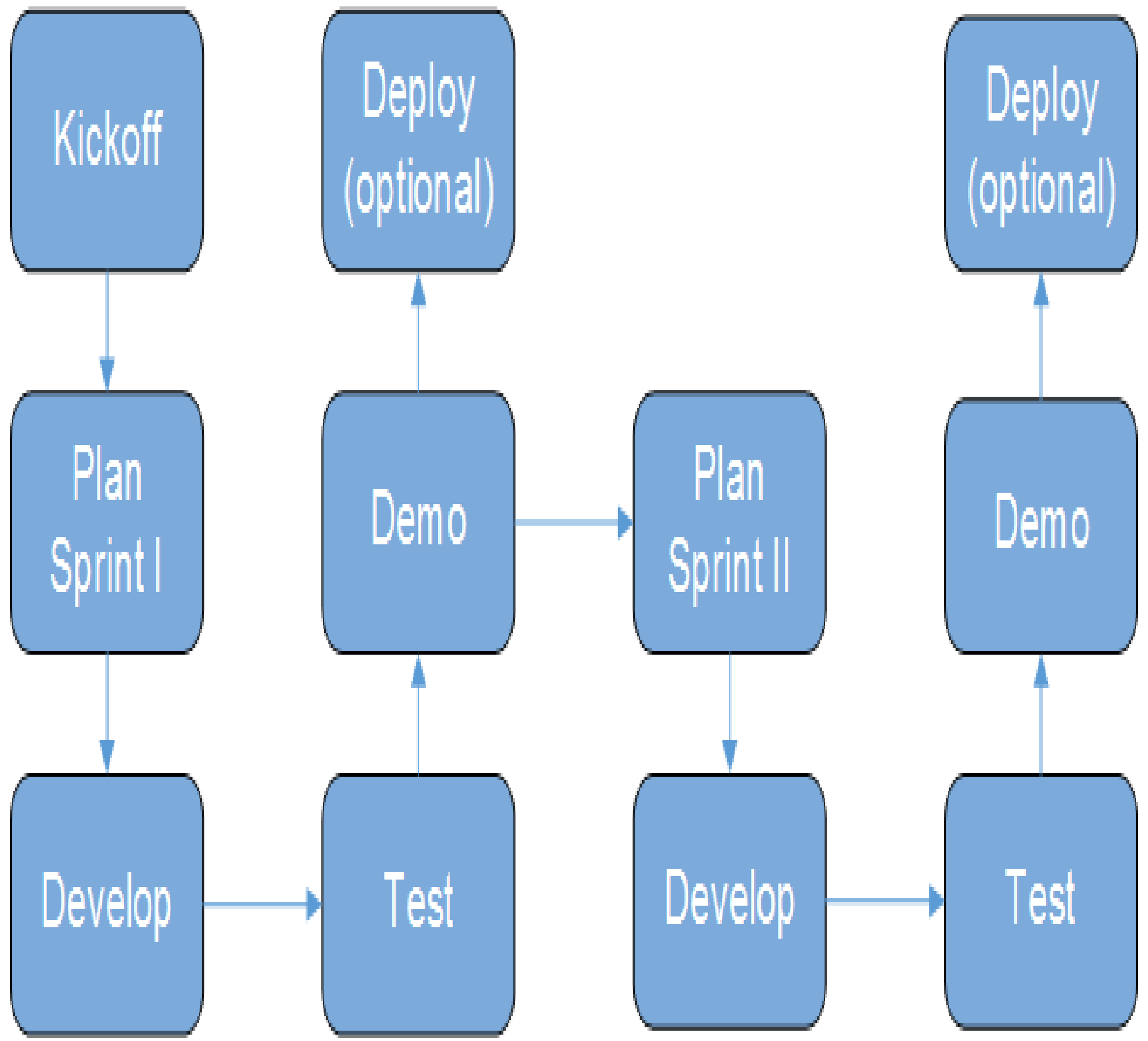


Agile

软件过程模型



螺旋



敏捷

Irrespective of the underlying process model

- a requirements phase to define the product, and
- either a formal design phase upfront or refactoring/redesign after the code is written to address maintainability and other software quality

will take place

无论底层的流程模型如何

- 一个用于定义产品的需求数阶段，以及
- 要么在前期进行正式的设计阶段，要么在代码编写后通过重构/重新设计来解决可维护性及其他软件质量问题

Software Definition

- Functional Requirements
  - What functionality the software system provides
    - Expressed as
      - User Stories (Agile Process Model)
      - “shall” statements (traditional IEEE/ISO based process models)
      - UML Use Case Diagram
    - Verified using
      - Acceptance Testing
- Non Functional Requirements
  - How well the functionality is provided
    - Expressed as
      - performance, scalability, reliability, availability, security, maintainability, usability, accessibility measures or metrics
    - Verified using
      - performance, scalability/load, [regression, black box, reliability], stress, security, maintainability, usability, accessibility tests
      - Static code analysis
      - Models

Software Design

- Expressed as
  - Architecture (UML Deployment, Component and/or Package Diagram)
  - Detailed Design (UML Class, and Sequence, State-Machine, Activity and/or Timing Diagrams)

软件定义

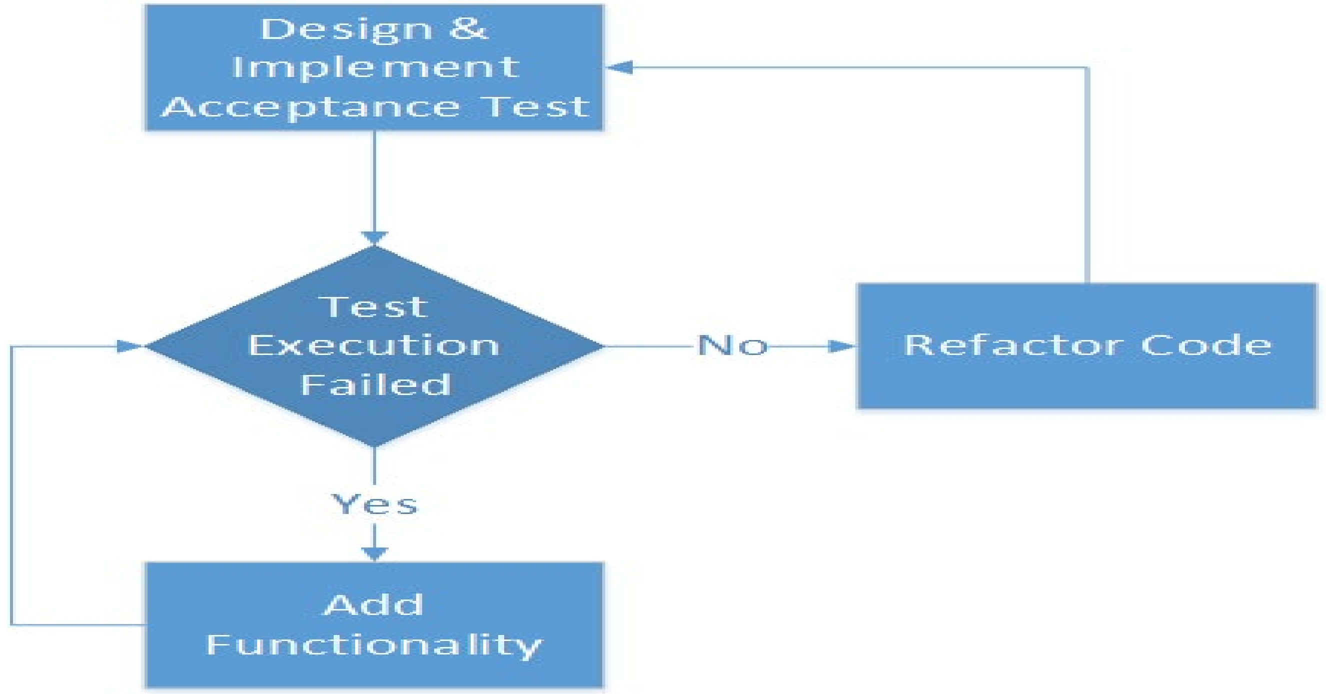
- 功能需求
  - 软件系统提供的功能
    - 表达方式为
      - 用户故事（敏捷过程模型）
      - “应”语句（传统的基于IEEE/ISO的过程模型）
      - UML用例图
    - 通过以下方式验证
      - 验收测试
- 非功能性需求 q求
  - 功能提供的完善程度
    - 表示为
      - 性能、可扩展性、可靠性、可用性、安全性、可维护性、可用性、可访问性方面的衡量标准或指标
    - 通过以下方式验证
      - 性能、可扩展性/负载、[回归、黑盒、可靠性], 压力、安全、可维护、可用性、可访问性测试
      - 静态代码分析
      - 模型

软件设计

- Expressed as
  - 架构（UML 部署图、组件图和/或包图）
  - 详细设计（UML 类图、序列图、状态机图、活动图和/或时序图）

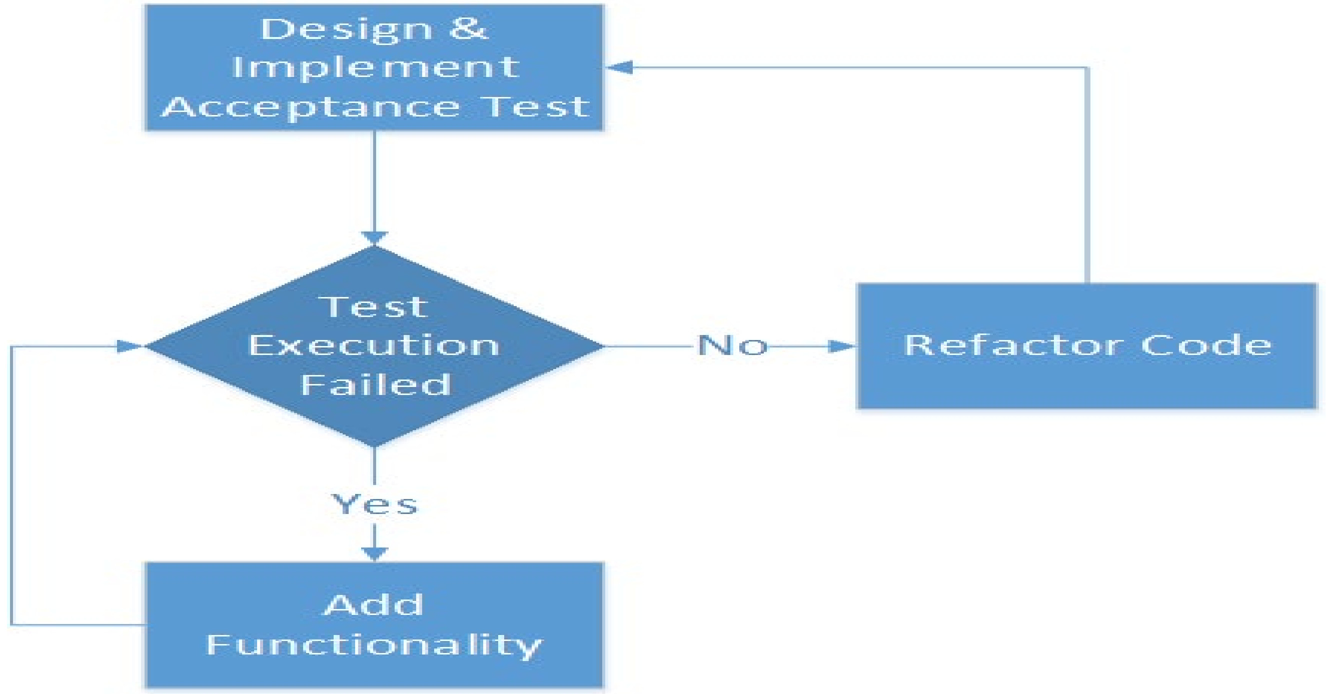
Behaviour (Test) Driven Development
(Android & Java, Espresso, JUnit, Jenkins & Git)

Test Driven development



行为（测试）驱动开发（Android 与 Java，Espresso，JUnit，Jenkins 和 Git）

测试驱动开发


Design & Implement Acceptance Test
Test Execution Failed
No
Refactor Code
Yes
Add Functionality