Introduction

Minsoo Ryu

Hanyang University

Topics covered

1. What is Software Engineering?
2. Software Crisis
3. Why is Software So Hard?
4. The Triad for Software Development

What is Software Engineering?

- ❒ Software engineering is <u>an engineering discipline that is concerned with all aspects of software production</u>
  - Software engineers should adopt <u>a systematic and organized approach</u> to their work and use <u>appropriate tools and techniques</u> depending on the problem to be solved, the development constraints and the resources available

- ❒ The term software engineering was <u>popularized by F.L. Bauer</u> during the NATO Software Engineering Conference in 1968
  - Software engineering was proposed to discuss 'software crisis'
  - <u>Software crisis</u>: orders of magnitude larger and more complex

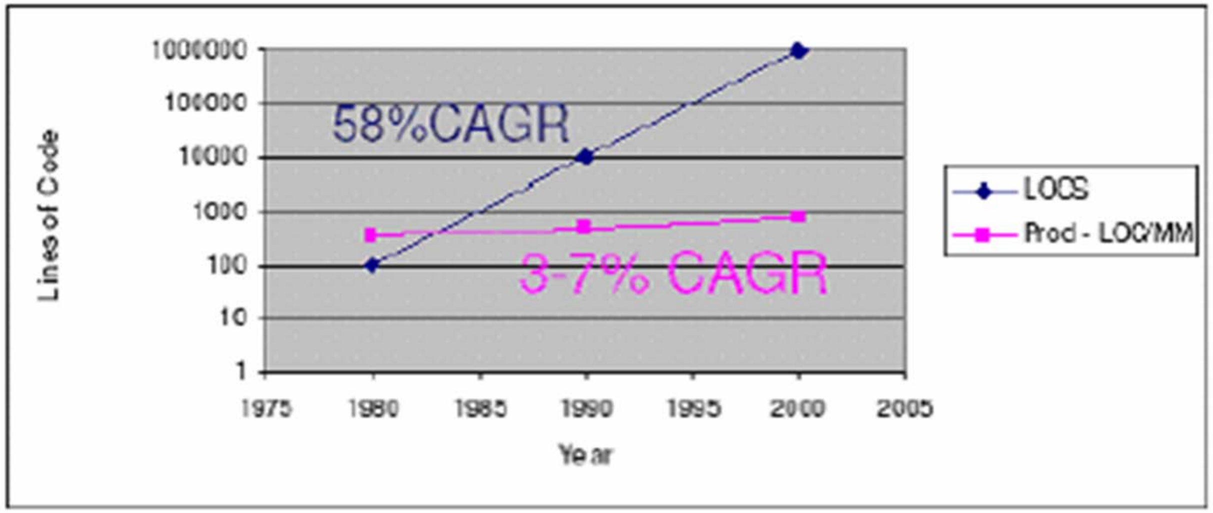
Software Crisis

- ❒ Software crisis was used to describe the impact of rapid increases in computer power and the complexity of the problems which could be tackled
  - In essence, it refers to the difficulty of writing correct, understandable, and verifiable computer programs
  - The roots of the software crisis are complexity, expectations, and change
- ❒ The software crisis manifested itself in several ways:
  - Projects running over-budget
  - Projects running over-time
  - Software was of low quality
  - Software often did not meet requirements
  - Projects were unmanageable and code difficult to maintain

Software Crisis

- ❒ Dominique Potier, Vice President at Thales, EU Information Society Technology, 2004
  - Embedded S/W Complexity: 58% annual increase
  - Embedded S/W Productivity: 3% – 7% annual increase

SW complexity & productivity growth



Embedded S/W Productivity by a single engineer is 0.5 – 5.0 LOC/Hour

Why is Software So Hard?

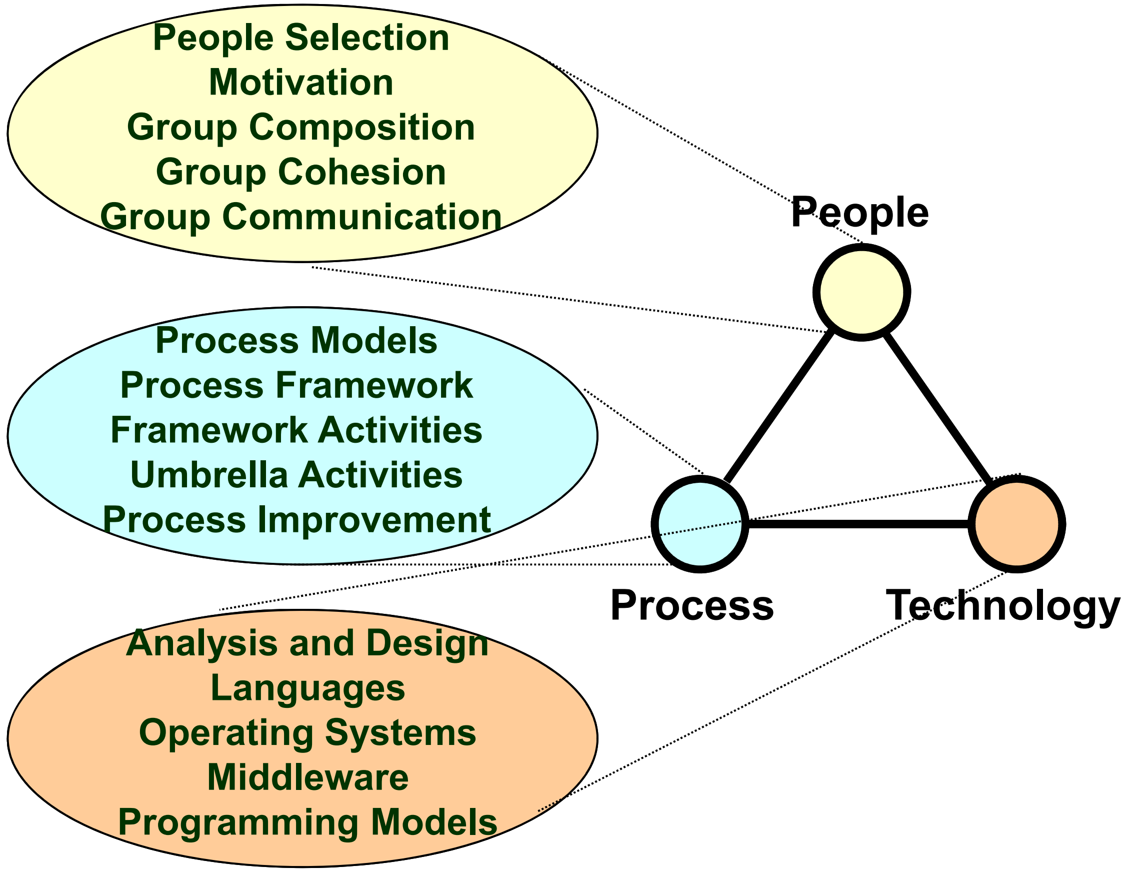
- **Software is intangible and prone to changes**
  - 프로젝트의 진행상황을 정확하게 파악하는 것이 어렵다
  - 표준화된 소프트웨어 개발 프로세스가 없다
  - 소프트웨어를 평가할 수 있는 정량화된 메트릭이 별로 없다
  - 소프트웨어는 재사용이 어렵다
  - 소프트웨어 기술은 너무 빨리 진화하며 예측이 어렵다
  - 요구명세가 자주 변경된다
  - 응용 도메인을 이해하는 소프트웨어 엔지니어를 찾기 어렵다

Any Solution to Software Crisis?

- Various processes and methodologies have been developed over the last few decades to "tame" the software crisis, with varying degrees of success

- However, it is widely agreed that there is no "silver bullet" — that is, no single approach which will prevent project overruns and failures in all cases
  - In general, software projects which are large, complicated, poorly-specified, and involve unfamiliar aspects, are still particularly vulnerable to large, unanticipated problems

The Triad for Software Development



"People are the most important aspect of the triad because with good people, the correct processes and technology can be developed, tested and implemented."

Tools are the fourth factor that affects the success of S/W project. With good tools, people, process, and technology can be best managed for better productivity and quality

Software Processes

Minsoo Ryu

Hanyang University

Topics covered

What is a Software Process

- Ivar Jacobson, Grady Booch, and James Rumbaugh
  - "A process defines who is doing what, when, and how to reach a certain goal"
- It is widely accepted that *the quality of the product depends on the quality of the process!*
- Synonyms include software life cycle and software development process
  - There are several models for such processes, each describing approaches to a variety of tasks or activities that take place during the process

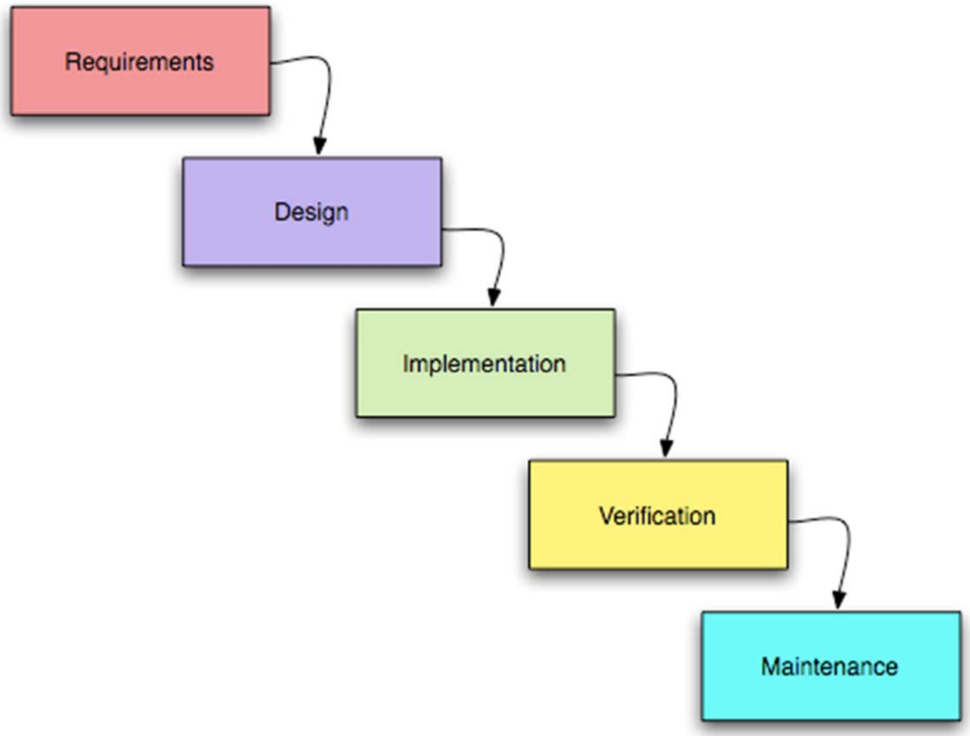
4 Major Software Processes

- ❒ Requirements engineering
- ❒ Design and implementation
- ❒ Verification and validation
- ❒ Evolution

Waterfall Development

- ❒ The waterfall model is a sequential software development model (a process for the creation of software)
  - ▪ in which development is seen as flowing steadily downwards (like a waterfall) through the phases of requirements analysis, design, implementation, testing (validation), integration, and maintenance
- ❒ The origin of the term "waterfall" is often cited to be an article published in 1970 by Winston W. Royce (1929–1995) although Royce did not use the term "waterfall" in this article
  - ▪ Ironically, Royce was actually presenting this model as an example of a flawed, non-working model (Royce 1970)

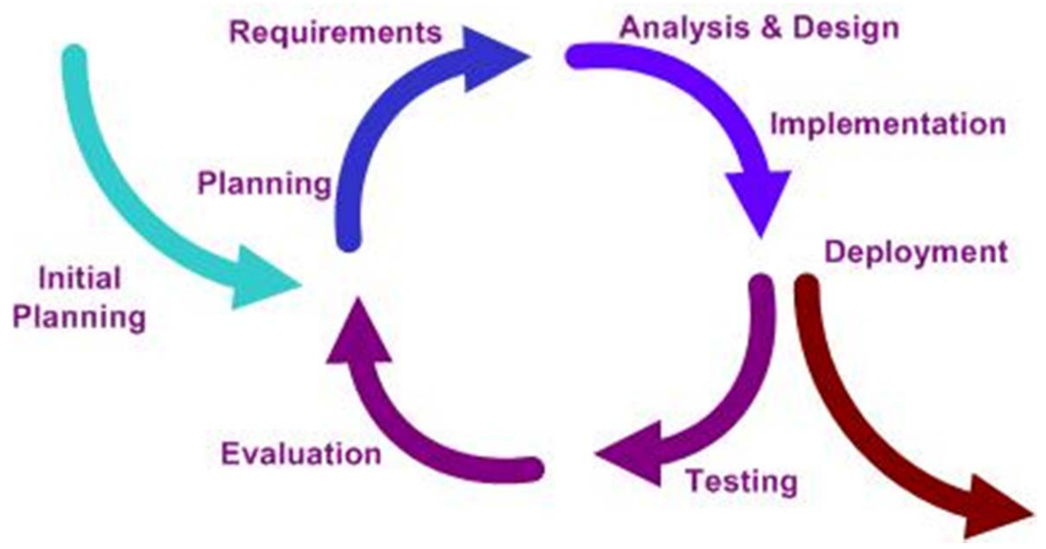
Waterfall Development



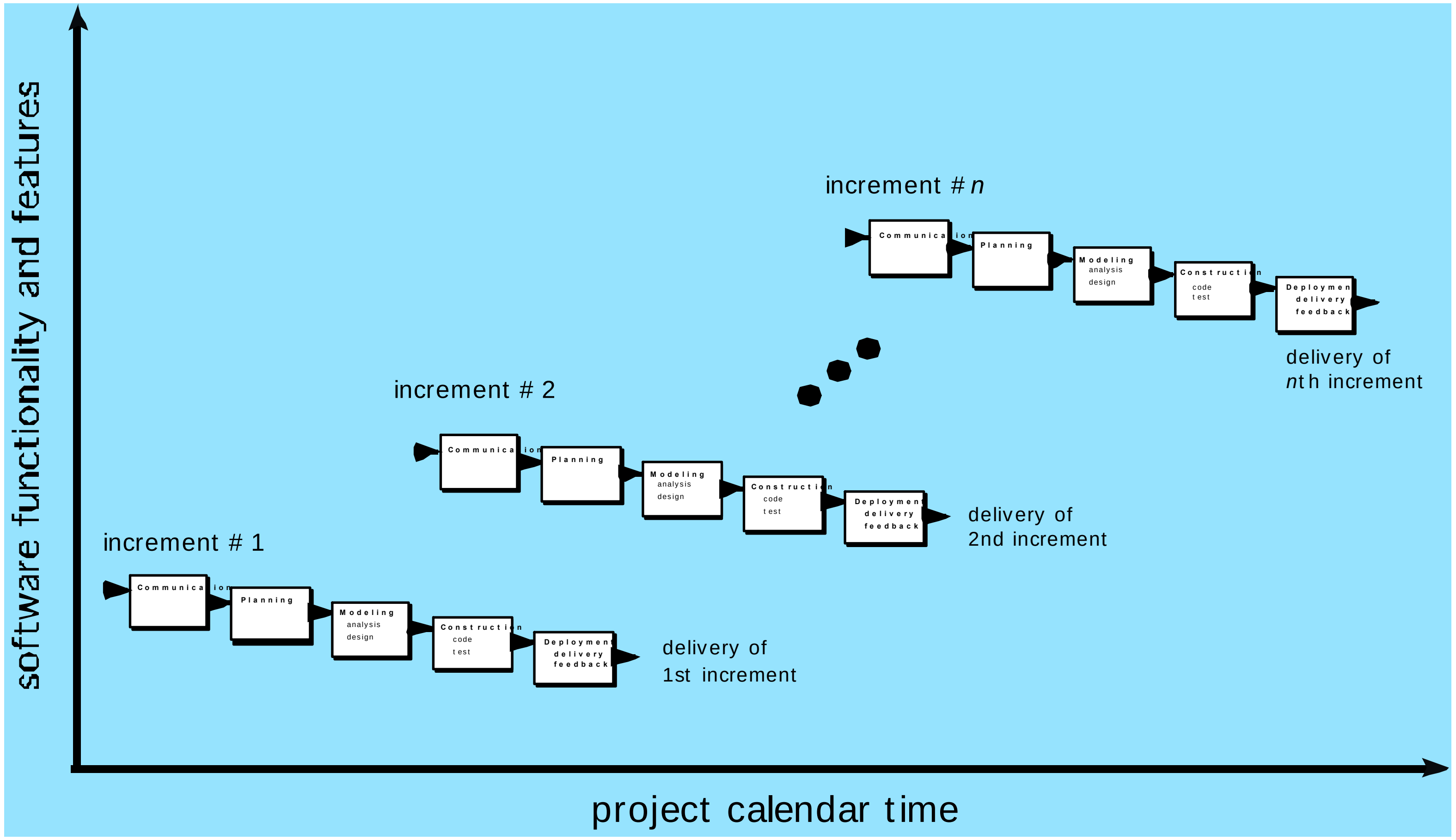
Criticism of the Waterfall Development

- Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements
- Therefore, this model is only appropriate when the requirements are well-understood and changes will be fairly limited during the design process
  - Few business systems have stable requirements
- The waterfall model is mostly used for large systems engineering projects where a system is developed at several sites

Iterative and Incremental Development



Iterative and Incremental Development


software functionality and features
increment # 1
Communication
Planning
Modeling analysis design
Construction code test
Deployment delivery feedback
delivery of 1st increment
increment # 2
delivery of 2nd increment
increment # n
delivery of nth increment
project calendar time

Advantages of Iterative and Incremental Development

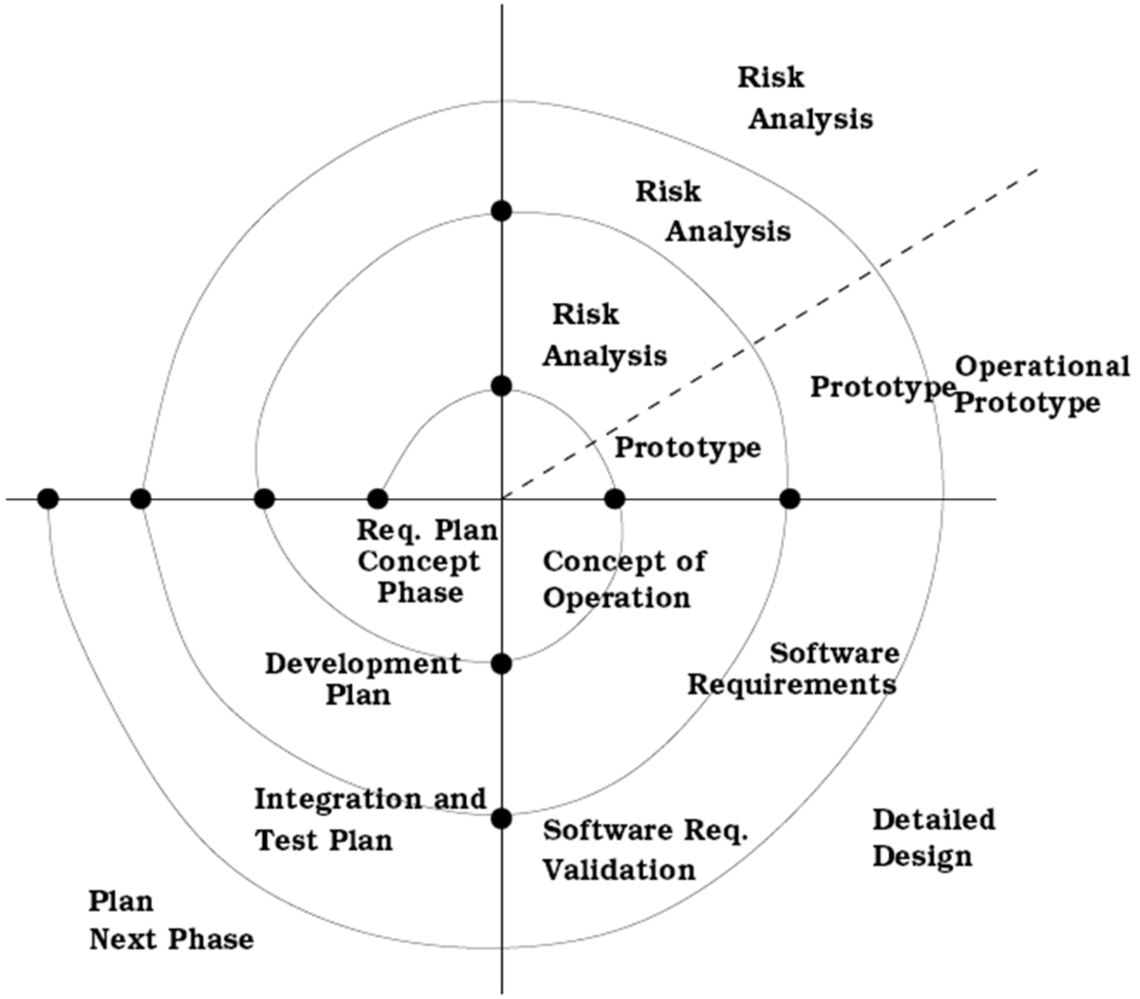
- ❒ Customer value can be delivered with each increment so system functionality is available earlier
- ❒ Early increments act as a prototype to help elicit requirements for later increments
- ❒ Lower risk of overall project failure
- ❒ The highest priority system services tend to receive the most testing

Spiral Development

❒ Proposed by Barry Bohem, 1988

- Process is represented as a spiral rather than as a sequence of activities with backtracking.
- Each loop in the spiral represents a phase in the process
- No fixed phases such as specification or design - loops in the spiral are chosen depending on what is required
- Risks are explicitly assessed and resolved throughout the process

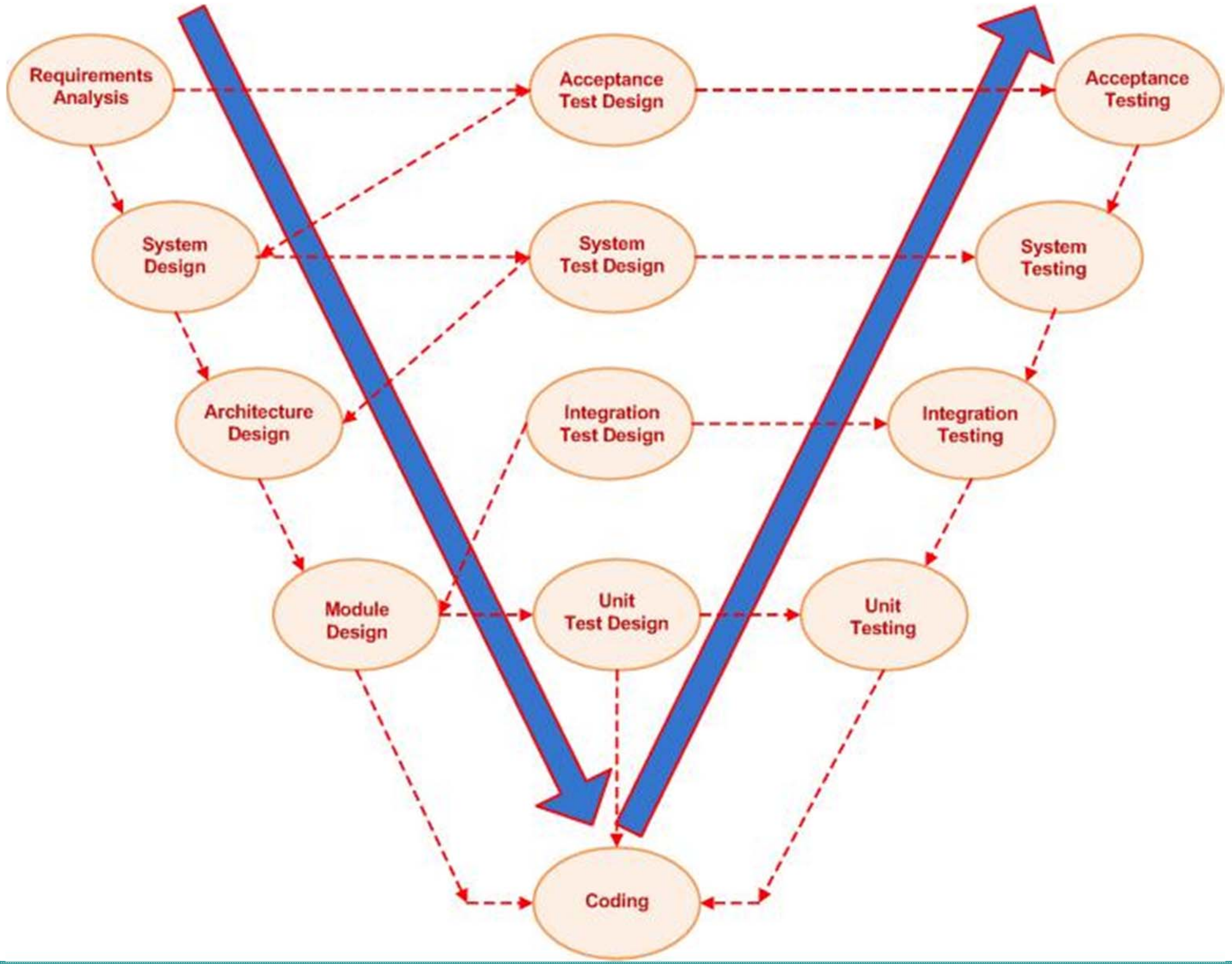
Spiral Development



V Model

- The V-model is a software development model which can be presumed to be the extension of the waterfall model
  - The V-Model demonstrates the relationships between each phase of the development life cycle and its associated phase of testing
- The V-model can be said to have developed as a result of the evolution of software testing
  - The tests in the ascending (Validation) hand are derived directly from their design or requirements counterparts in the descending (Verification) hand
  - The 'V' can also stand for the terms Verification and Validation

V Model


Requirements Analysis
Acceptance Test Design
Acceptance Testing
System Design
System Test Design
System Testing
Architecture Design
Integration Test Design
Integration Testing
Module Design
Unit Test Design
Unit Testing
Coding