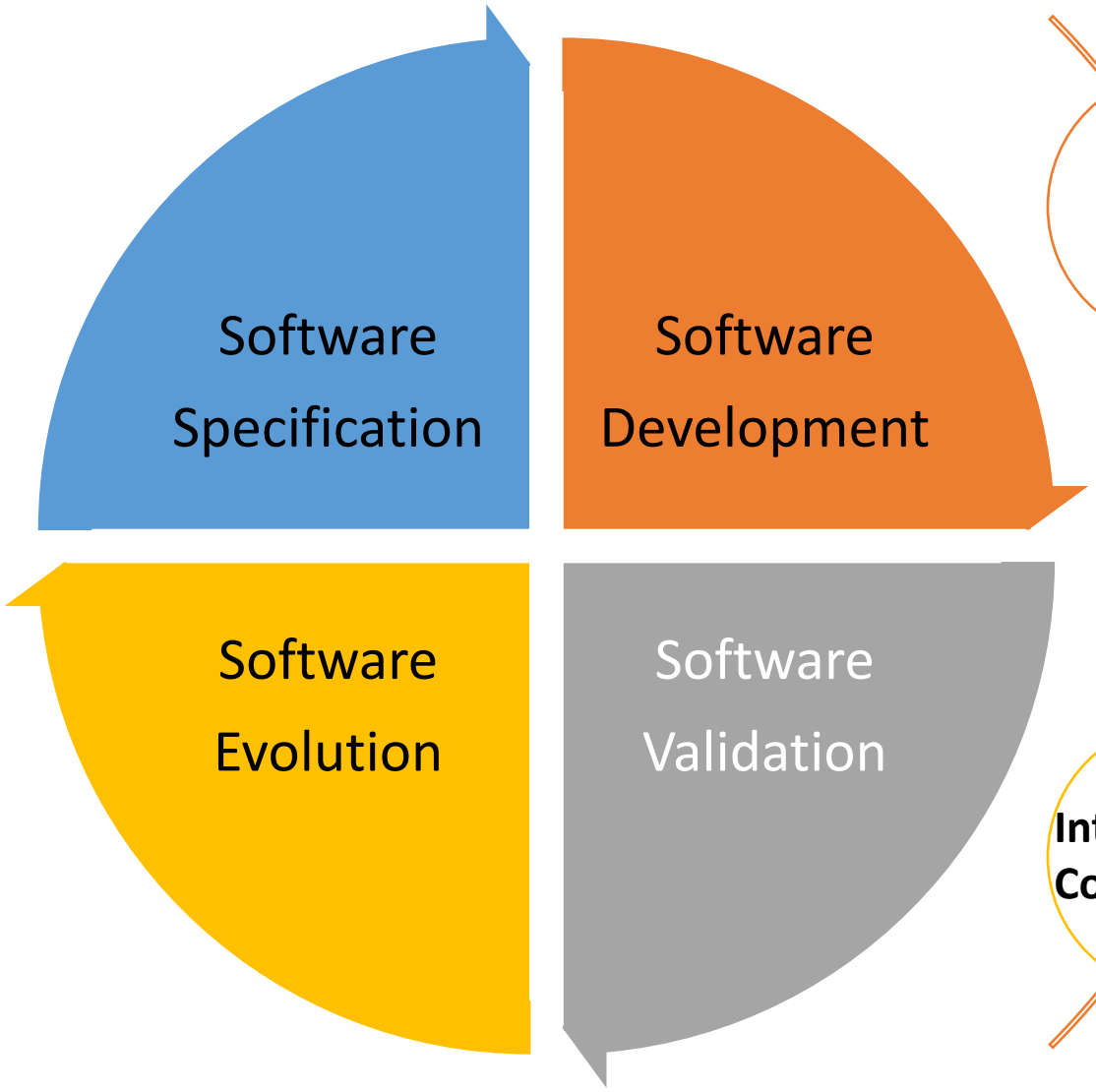
CS335, 2021-2022

Software Processes

Dapeng Dong

Fundamental Activities in General Process Models



The Waterfall Model

Represents the fundamental activities as separate process phases including requirements specification, design, implementation, testing and maintenance.

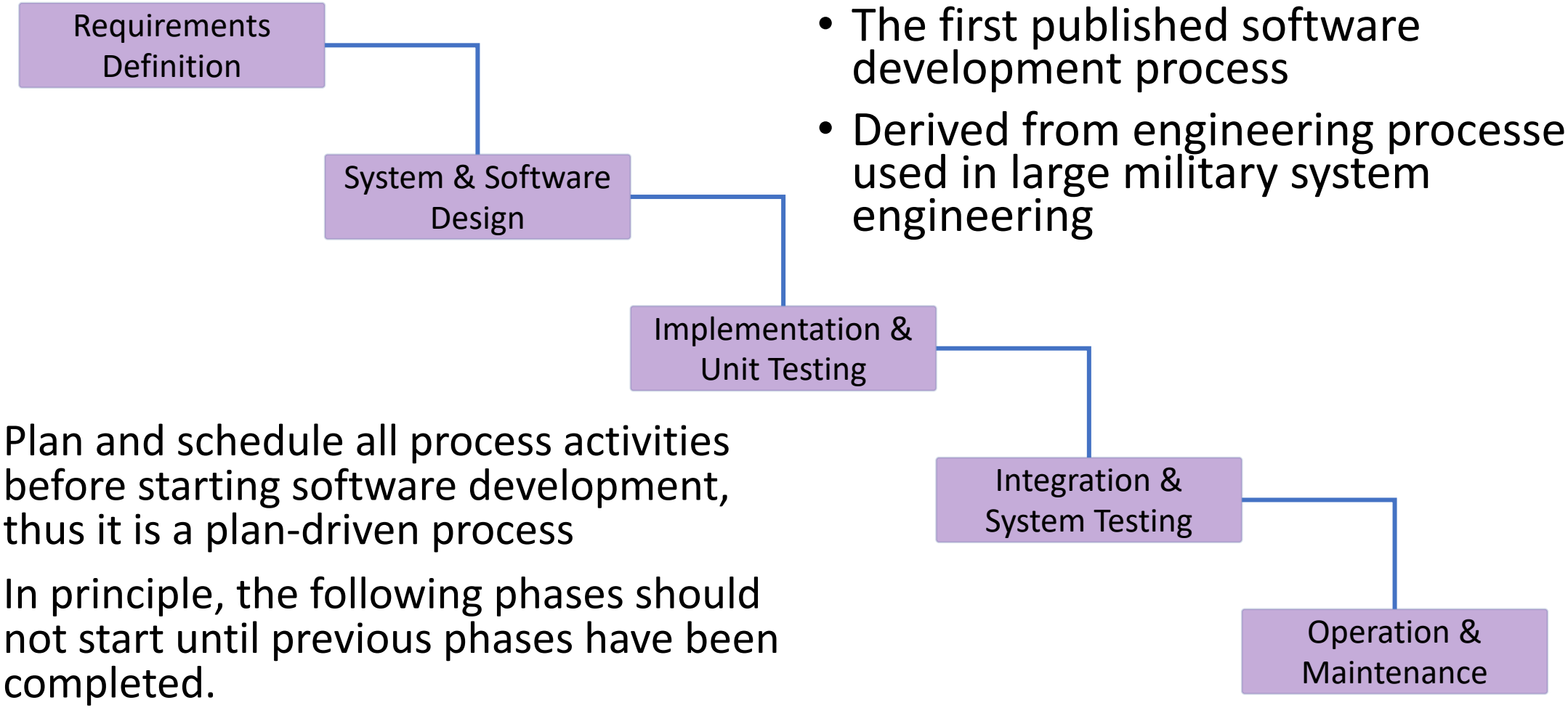
Incremental Development

Interleaves the fundamental activities; the system is developed as a series of 'increment', with each increment adding functionality to the previous version.

Integration & Configuration

Relies on the availability of reusable components; the system development focuses on configuration, composition and integration for use in a new setting.

The Waterfall Model



- The first published software development process
- Derived from engineering processes used in large military system engineering

- Plan and schedule all process activities before starting software development, thus it is a plan-driven process
- In principle, the following phases should not start until previous phases have been completed.

Reading (The Waterfall Model): *Royce, W.W., 1987, March. Managing the development of large software systems: concepts and techniques. In Proceedings of the 9th international conference on Software Engineering (pp. 328-338).*

The Stages of the Waterfall Model (1)

- Requirement analysis and definition
  - Gathers information about the required functionalities (e.g., the software system shall provide a login function) and constraints (e.g., the software system shall use PostgreSQL as backend database as it's compatible with the existing IT infrastructure of the organization)
  - Analyzes whether the software system is achievable, given a set of budget, time frame, resources, skills and available technologies
  - Creates detailed system specification
- System and Software Design
  - Allocates the requirements to hardware or software systems
  - Establishes an overall system architecture (e.g., identify and describe the interfaces of components and their relationships, etc.)

The Stages of the Waterfall Model (2)

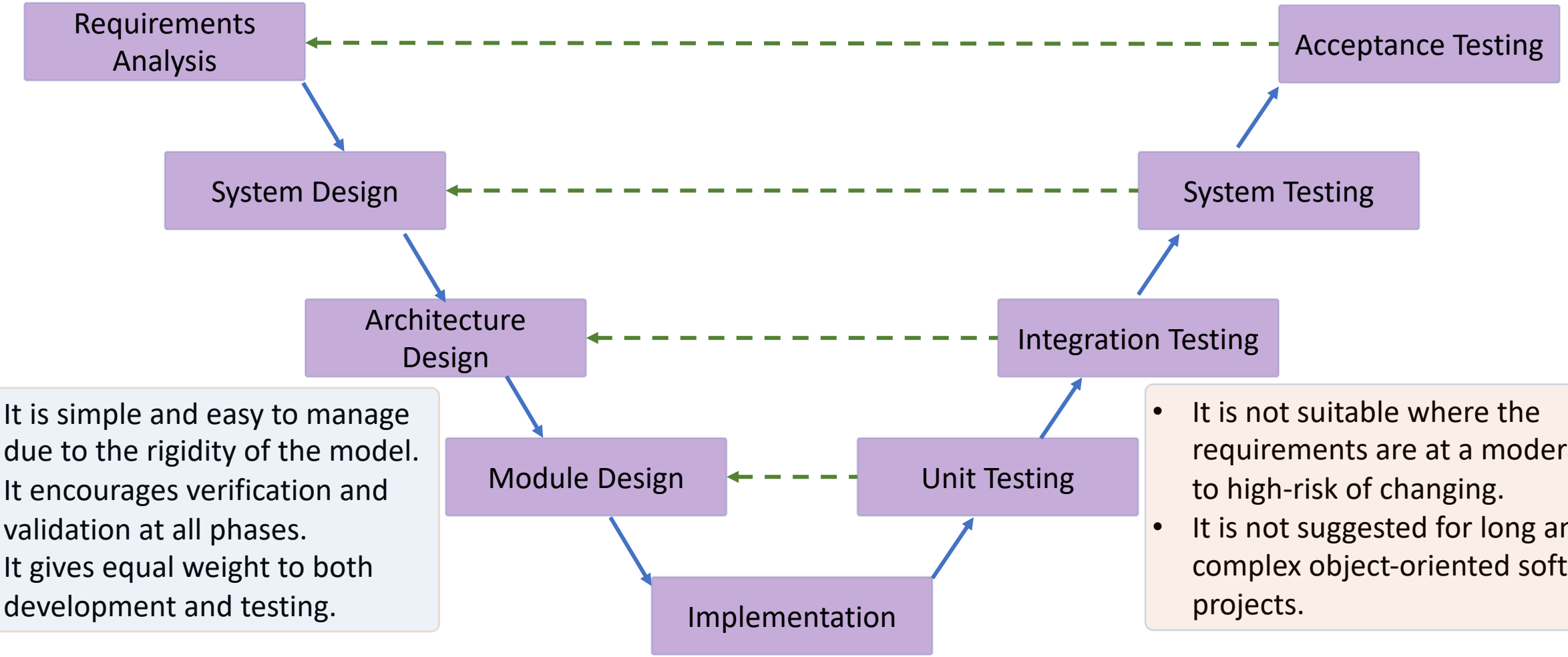
- Implementation and Unit Testing
  - The software design is realized as a set of programs or program units
  - Program units are tested to meet their specification
- Integration and System Testing
  - The individual program units or programs are integrated and tested as a complete system
  - The verified and validated software system is delivered to the user after the testing
- Operation and Maintenance
  - The system is deployed and put into practical use
  - The software system maybe continuously updated to patch for errors or security threats; to improve stability of the system; or to enhance the system’s functionality

Applications

- Embedded Systems
  - Software needs to interface with hardware systems
  - Usually, hardware is inflexible; it is often inconvenient to delay decisions on the software's functionality until it is being implemented
- Critical Systems
  - There is need for extensive safety and security analysis of the software specification and design
  - The software specification and the design documents must be complete
  - Safety related problems are usually expensive to correct
- Large Software Systems
  - Multiple partners jointly developing the software system, complete specifications may be needed to allow for the independent development of different subsystems.

- Software Systems that must cope with changes
  - Changes are inevitable in many software projects, the nature of the one-way process of the waterfall model can NOT cope with changes effectively.

The V-Model

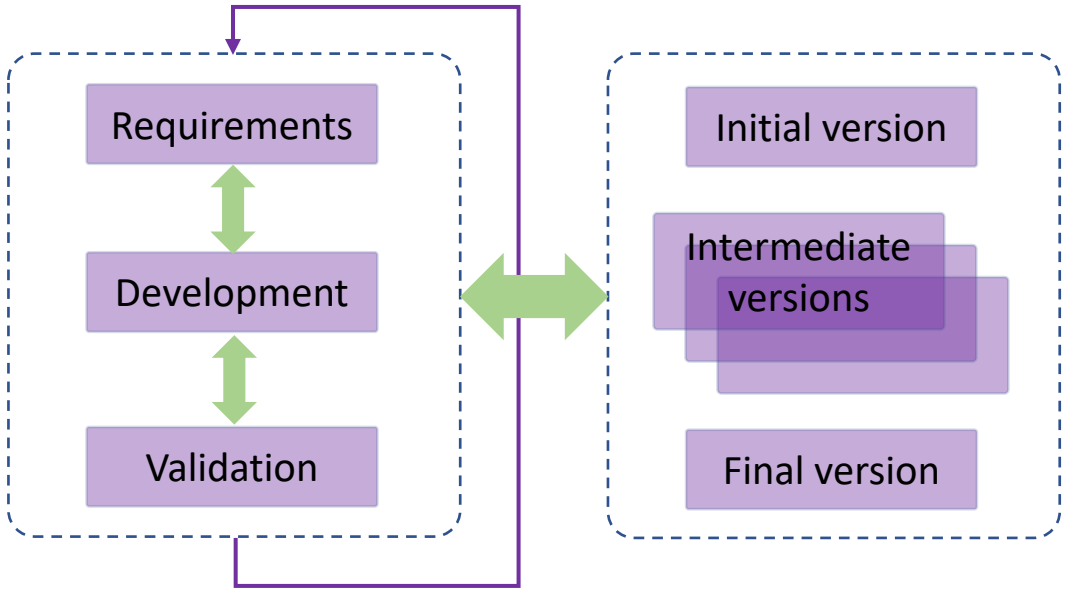


- It is simple and easy to manage due to the rigidity of the model.
- It encourages verification and validation at all phases.
- It gives equal weight to both development and testing.

- It is not suitable where the requirements are at a moderate to high-risk of changing.
- It is not suggested for long and complex object-oriented software projects.

Reading (The V-Model): *Forsberg, K. and Mooz, H., 1991, October. The relationship of system engineering to the project cycle. In INCOSE International Symposium (Vol. 1, No. 1, pp. 57-65).*

Incremental Development



- Begins with a simple implementation of a part of the software system.
- With each increment the product evolves with enhancements being added every time until the final version is completed.
- Code is written and tested in smaller pieces, thus reduces risks associated with the process.
- It allows changes to be included easily along the development process.

Advantages of Incremental Development

- Reduced cost for implementing requirements changes
  - The amount of analysis and documentation that need to be redone is significantly less then it is required with the Waterfall model
- Receiving feedback early
  - It is easier to get user feedback on the work done
  - Users can comment on the software and provide useful suggestions
- Early delivery of partially working product
  - Users can start to use and gain values from the early release of the software
  - Incremental development does not mean each increment needs to be delivered to users.

Disadvantages of Incremental Development

- The process is not visible
  - Regular deliverables are needed to measure progress
  - It is not cost effective to produce documentation for every version of the system if the incremental iteration is short
- System structure tends to degrade as new increment are added
  - Frequent changes lead to extra burden on source code and project management
  - This can also be harmful to the software system architecture and as more functional components being added to the system, the existing architecture might need to be modified accordingly, which could lead to cascade effect. Thus, ***refactoring*** is often needed.

Increment Development models MAY NOT be suitable for development of large, complex and long-lifetime software system projects.

Terminology: Refactoring

"A set of program restructuring operations (refactoring) that support the design, evolution and reuse of object-oriented application frameworks."

-- Opdyke, W.F., 1992. Refactoring object-oriented frameworks.

"A change made to the internal structure of software to make it easier to understand and cheaper to modify without changing its observable behavior."

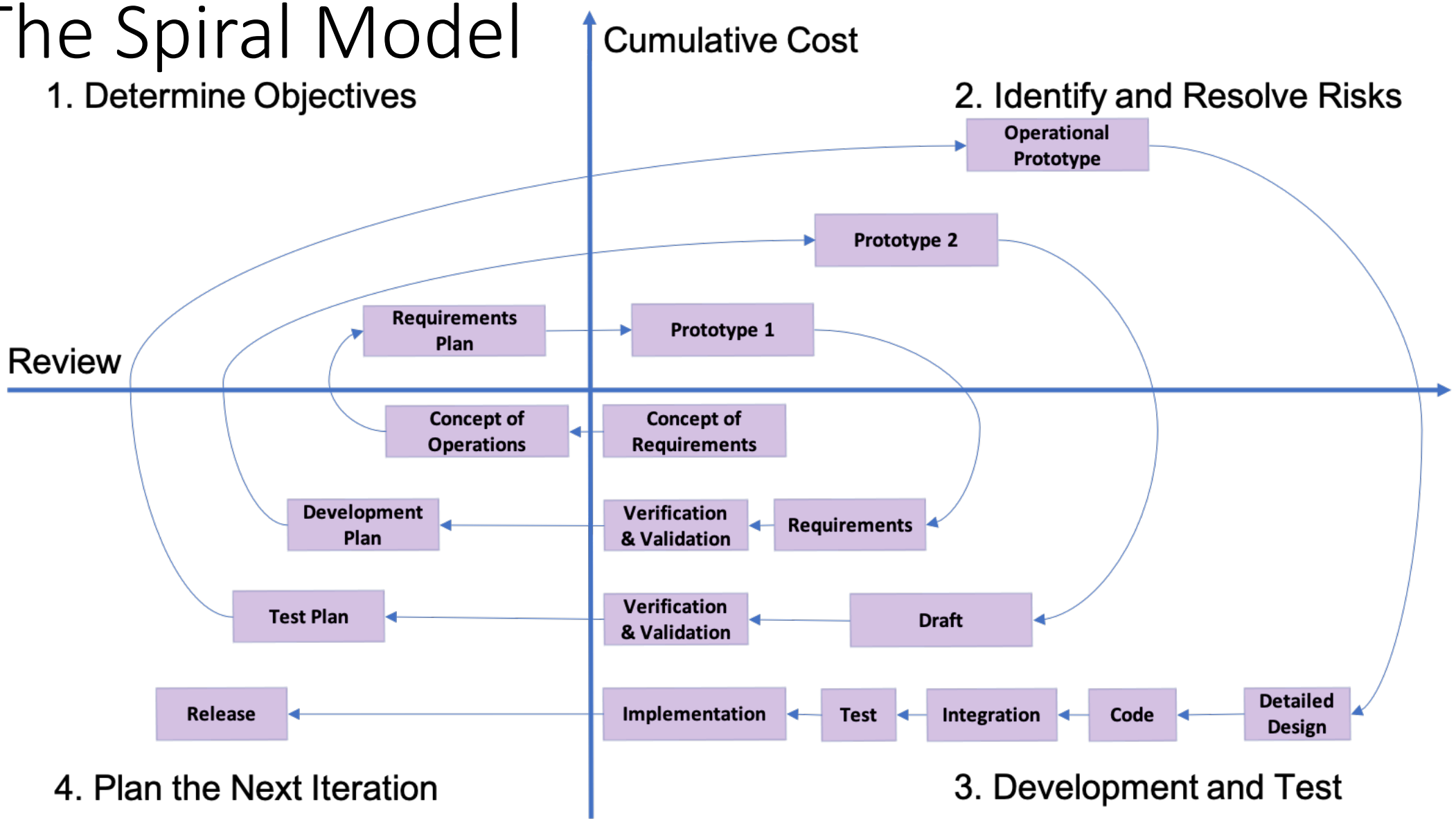
-- Fowler, M., 2018. Refactoring: improving the design of existing code. Addison-Wesley Professional.

Consider:

- *"when you have to add a feature to a program, but the code is not structured in a convenient way, first refactor to make it easy to add the feature, then add the feature."*
- *"refactoring changes the programs in small steps, so if you make a mistake, it is easy to find where the bug is."*
- *"before you start refactoring, make sure you have a solid suite of tests."*

-- Fowler, M., 2018. Refactoring: improving the design of existing code. Addison-Wesley Professional.

The Spiral Model



Reading (The Spiral Model): *Boehm, B.W., 1988. A spiral model of software development and enhancement. Computer, 21(5), pp.61-72.*

Terminologies: *Prototype, Proof-of-Concept and Skeleton System*

From a Software Architecture Perspective

"A prototype is a temporary implementation of some functional subset of the system, often presented to users for feedback and validation, which is then discarded when the validation exercise is complete."

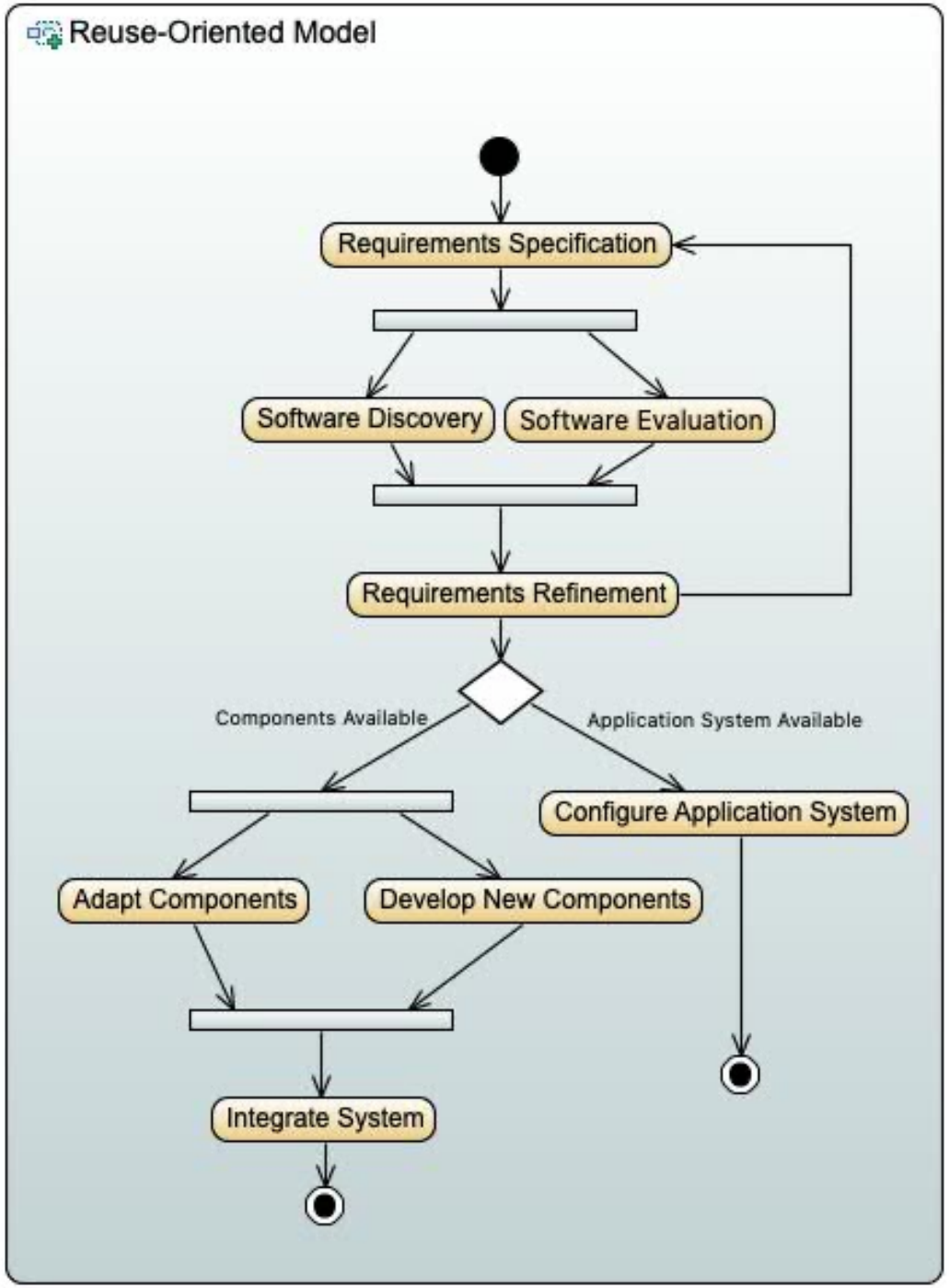
"A proof-of-concept is some code designed to prove that a risky element of the proposed architecture is feasible and to highlight any problems and pitfalls. A proof-of-concept is also a temporary implementation, which is discussed when it has served its purpose and the risk under investigation is understood."

"A skeleton implements the system's main architectural structures but contains only a minimal subset of the system's functionality. A skeleton system is retained rather than discarded and becomes the basis for the construction phase."
A skeleton system is sometimes called an "evolutionary prototype".

Reference: Rozanski, N. and Woods, E., 2012. *Software systems architecture: working with stakeholders using viewpoints and perspectives*. Addison-Wesley.

Integration and Configuration

- The Reuse-Oriented development process focuses on the reuse of existing software.
  - E.g., existing classes, libraries, patterns, designs, standalone application systems, web-services, etc.
- It relies on a base of reusable software component and an integration framework for the composition of the components.
- It has gained extreme popularity in development of software system projects since 2000.



The Stages in the Process

- Requirements Specification
  - The initial requirements for the system are collected.
  - It should include brief descriptions of essential requirements and system features.
- Software Discovery and Evaluation
  - Outline the software requirements, and search for components and systems that provide the functionality required.
  - Candidate components and systems are evaluated to see if they are suitable for use in the system.
- Requirements Refinement
  - The requirements are refined or modified using information about the reusable components and applications
- Application System Configuration
  - If an off-the-shelf application system is available, it can be configured for use to create the new system, then it should be considered first.
- Component Adaptation and Integration
  - If there are no off-the-shelf application systems available, available components and/or development of new components should be considered, then integrated to create the system.

Advantages and Disadvantages

- Reducing the amount of software to be developed
- Reducing the cost and risks
  - less development activities leads to reduced cost
  - existing components/application systems are usually proven to be correct, stable and secure
  - Existing components/application systems offers good Application Programming Interfaces (APIs) for integration
- Faster delivery of the software

- The requirements compromises may drift the original idea of the system
- Limited control over the system evolution
  - Newer version of the reusable components may not be compatible with the current version used, for example, refactored APIs, new functionalities, deprecated function calls, etc.

Applicability of Software Models

”There is no universal process model that is right for all kinds of software development.”

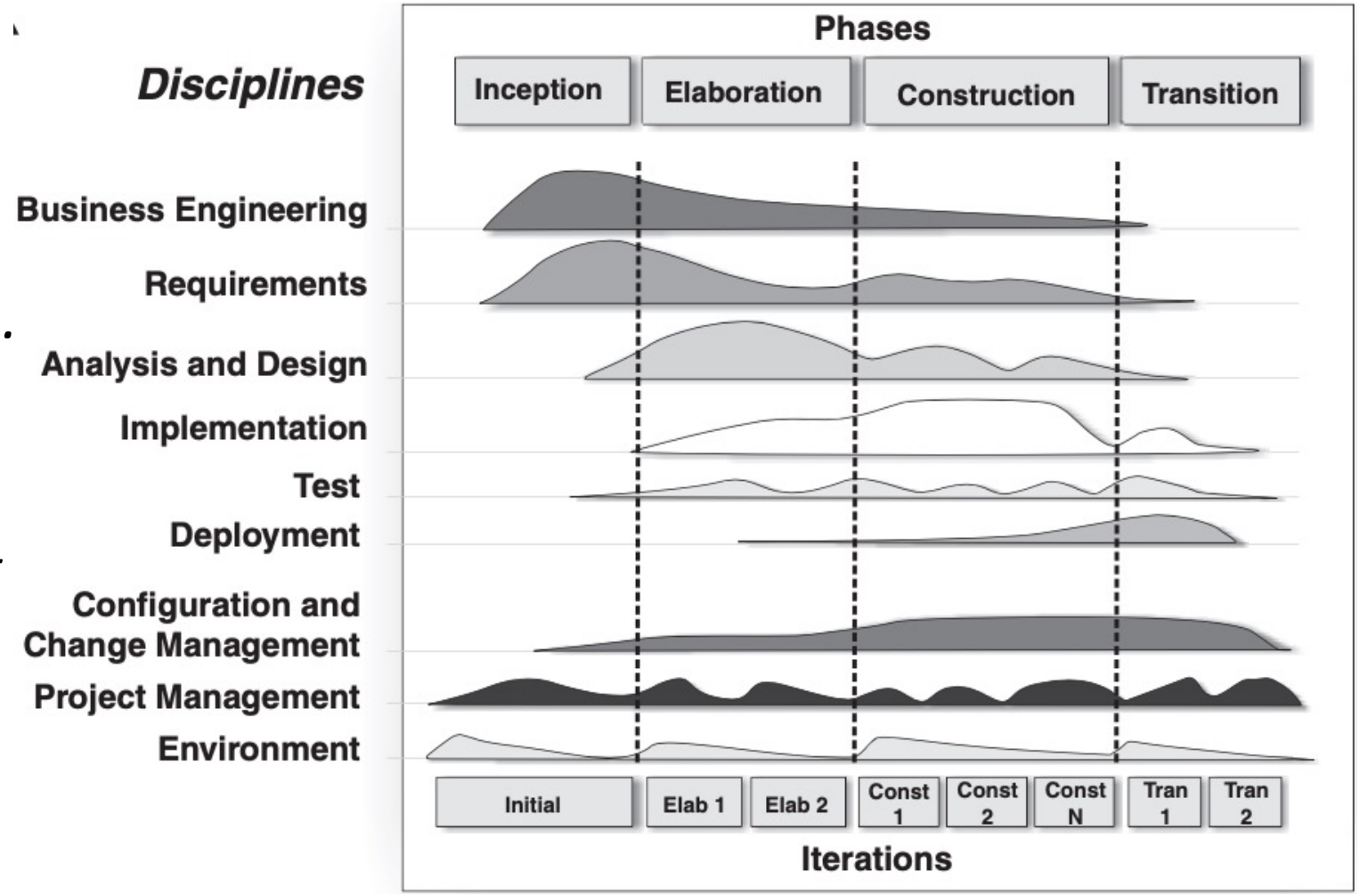
”The right process depends on the customer and regulatory requirements, the environment where the software will be used, and the type of software being developed”

“The majority of practical software processes are based on a general model but often incorporate features of other models.”

-- Sommerville, I., 2016. Software engineering., 10th Edition. Pearson Education.

An Attempt at A Universal Process Model

- *A disciplined approach to assigning tasks and responsibilities within a development organization.*
- *Ensures the production of high-quality software that meets the needs of its users within a predictable schedule and budget.*



The Rational Unified Process (RUP)

***Source (RUP)**: TP026B, R., 2017. Rational Unified Process. October, zv, 18.*

Rational Unified Process -- Inception

"Envision the product scope, vision, and business case."

"Inception is the initial short step to establish a common vision and basic scope for the project. The purpose of the inception phase is NOT to define all the requirements, or generate a believable estimate or project plan"

--Larman, C., 2012. Applying UML and patterns: an introduction to object oriented analysis and design and interactive development. Pearson Education.

- Analyses part of the use cases
- Analysis of critical non-functional requirements
  - E.g., security, reliability, extensibility, etc.
- Creation of business cases
- Preparation of the development environment

Rational Unified Process – Elaboration

"The code and design are production-quality portions of the final system."

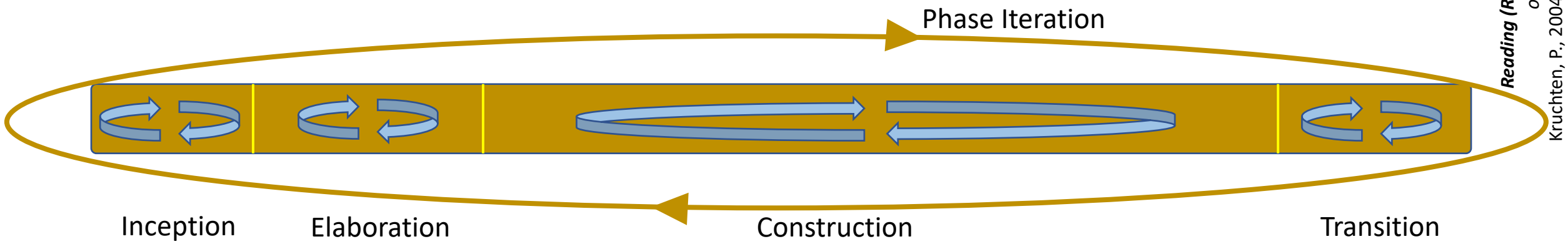
"Build the core architecture, resolve the high-risk elements, define most requirements and estimate the overall schedule and resources."

--Larman, C., 2012. Applying UML and patterns: an introduction to object oriented analysis and design and interactive development. Pearson Education.

- The core, risky software architecture is programmed and tested
- The majority of requirements are discovered and stabilized
- The major risks are mitigated
- It involves a series iterations
  - Recommended to be between two and six weeks

Rational Unified Process – Construction and Transition

- Construction
  - Concerned with system design, implementation and testing
  - Parts of the system are developed and integrated
  - On completion of this phase, a partially working system and associated documentation should be ready for delivery to users
- Transition
  - Deploy the system in a real production environment
  - On completion of this phase, the software system should be:
    1. well documented
    2. working correctly
    3. running in its operational environment



Reading (RUP): *Larman, C., 2012. Applying UML and patterns: an introduction to object oriented analysis and design and interactive development. Pearson Education.*
Kruchten, P., 2004. *The rational unified process: an introduction.* Addison-Wesley Professional.