

Classical Software Life Cycle Models

SWEN 301

Trimester 1, 2016

Lecturer: Dr Hui Ma

Engineering and Computer Science



Lecture slides make use of material provided on
the textbook's companion website

Motivation

- Problem:
 - Create high quality software with limited resources, by a deadline, in the context of constant change
- Quality development processes help
- Need to model development process to address
 - Where to find a particular kind of fault
 - How to find faults earlier
 - How to build in fault tolerance
 - What are alternative activities

The Meaning of Process

- A **process**: a series of steps involving activities, constraints, and resources that produce an intended output of some kind
- A process involves a set of tools and techniques
- **Tool**: an instrument or automated system for accomplishing something in a better way
- We sometimes refer to the process as a **life cycle**

The Importance of Processes:

- Impose consistency and structure on a set of activities
- Guide us to understand, control, examine, and improve the activities
- Enable us to capture our experiences and pass them along

Software Process Models

Reasons for Modelling a Process:

- To form a common understanding
- To find inconsistencies, redundancies, omissions
- To find and evaluate appropriate activities for reaching process goals
- To tailor a general process for a particular situation in which it will be used

Software Process Models

- When a process involves building a software, the process may be referred to as **software life cycle**
 - Requirements analysis and definition
 - System (architecture) design
 - Program (detailed/procedural) design
 - Writing programs (coding/implementation)
 - Testing: unit, integration, system
 - System delivery (deployment)
-
- Maintenance

Software Process Models

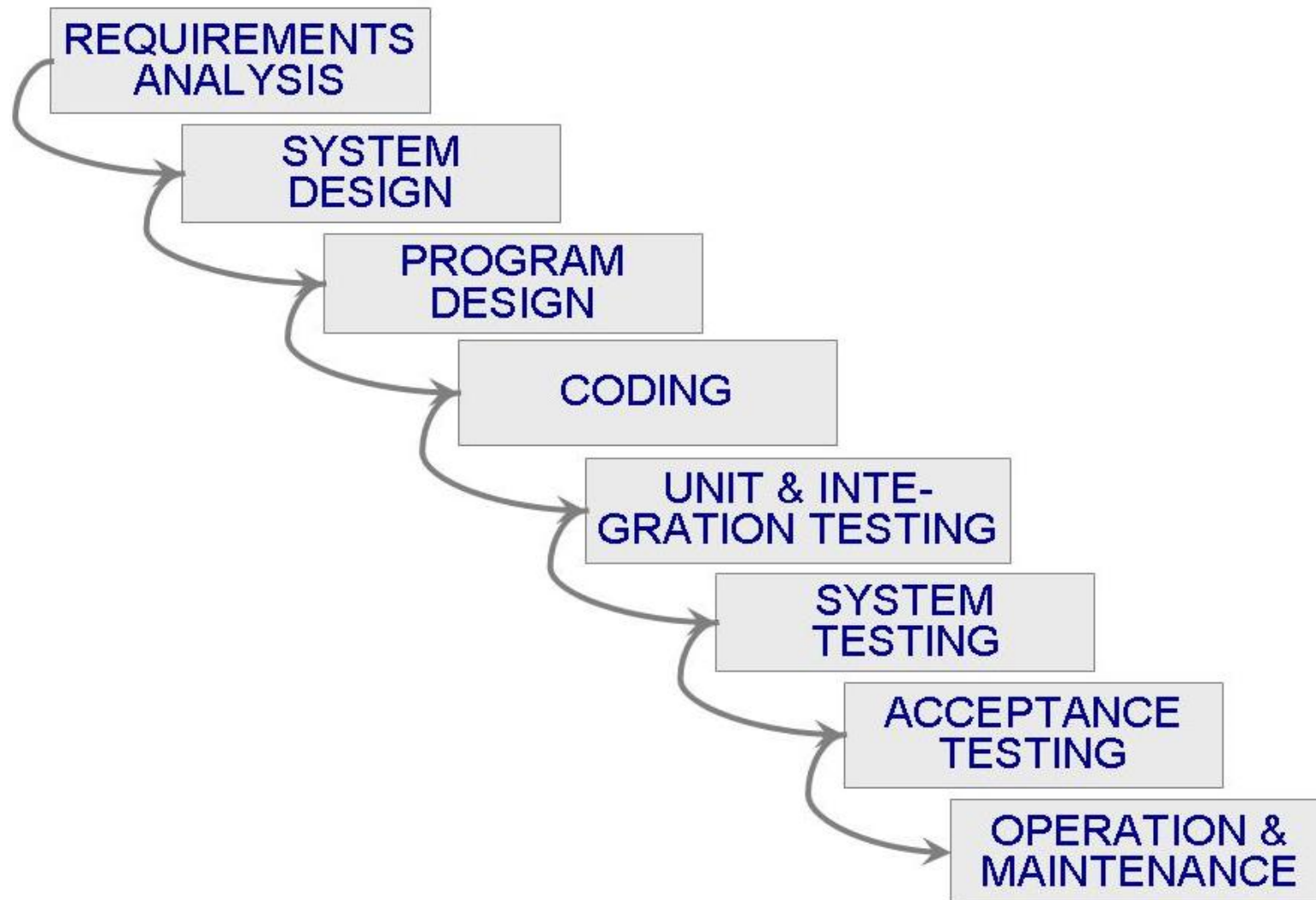
Software Development Process Models

- Waterfall model
- V model
- Prototyping model
- Spiral model
- Phased development: increments and iterations
- Rational Unified Model
- Other models: Agile methods, Operational specification, Transformational model



Waterfall Model

- First proposed by Winston Royce
 - “Managing the Development of Large Software Systems: Concepts and Techniques”, 1970
- Commonly assumed to be a linear, highly-structured lifecycle
 - Royce actually advocated iteration and incremental development
 - Later argued that the Waterfall lifecycle only works for straight-forward projects

Waterfall Model (cont.)



Waterfall Model (cont.)

- One of the first process development models proposed
- Works for well understood problems with minimal or no changes in the requirements
- Simple and easy to explain to customers 
- It presents
 - a very high-level view of the development process
 - sequence of process activities
- Each major phase is marked by milestones and deliverables (artifacts) 

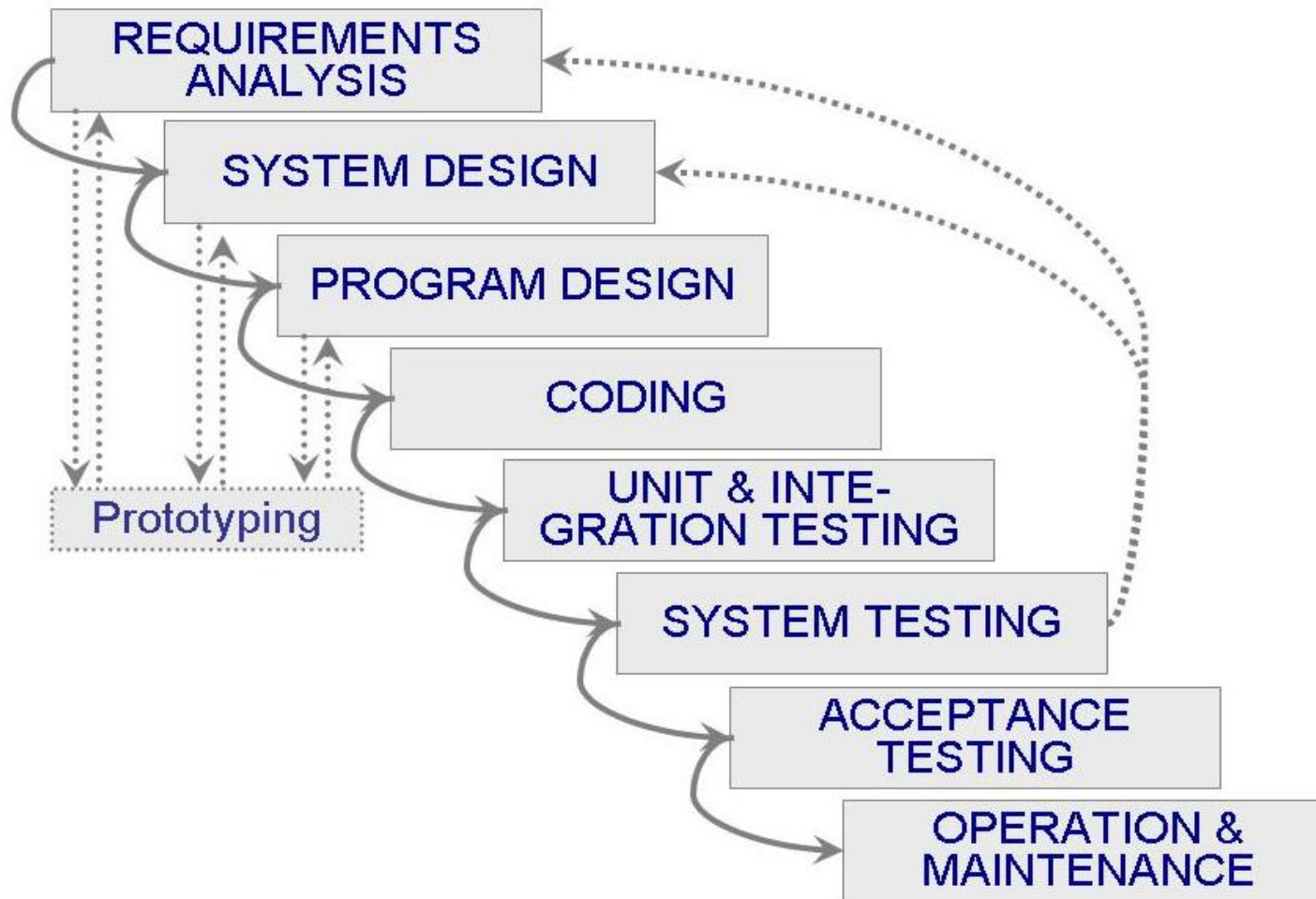


Drawbacks of The Waterfall Model

- There is no iteration in waterfall model
 - Most software developments apply a great many iterations
- Provides no guidance how to handle changes to products and activities during development (assumes requirements can be frozen)
- Views software development as manufacturing process rather than as creative process
- Long wait before a final product

Waterfall Model with Prototype

- Waterfall model with prototyping



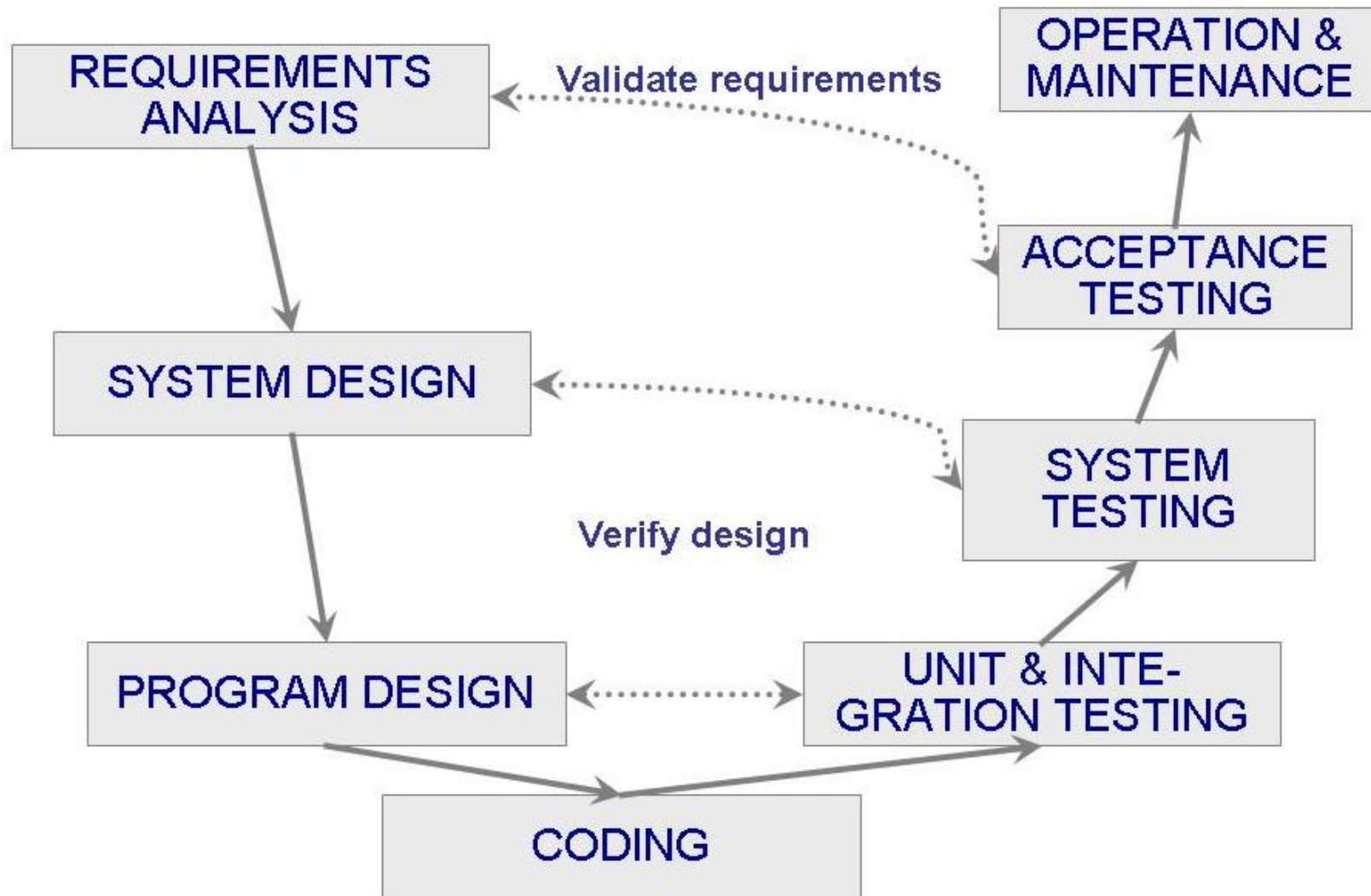
Waterfall Model with Prototype (cont.)

- A prototype¹ is a partially developed product
- Prototyping helps
 - developers assess alternative design strategies (design prototype)
 - users understand what the system will be like (user interface prototype)
- Prototyping is useful for verification and validation

V Model

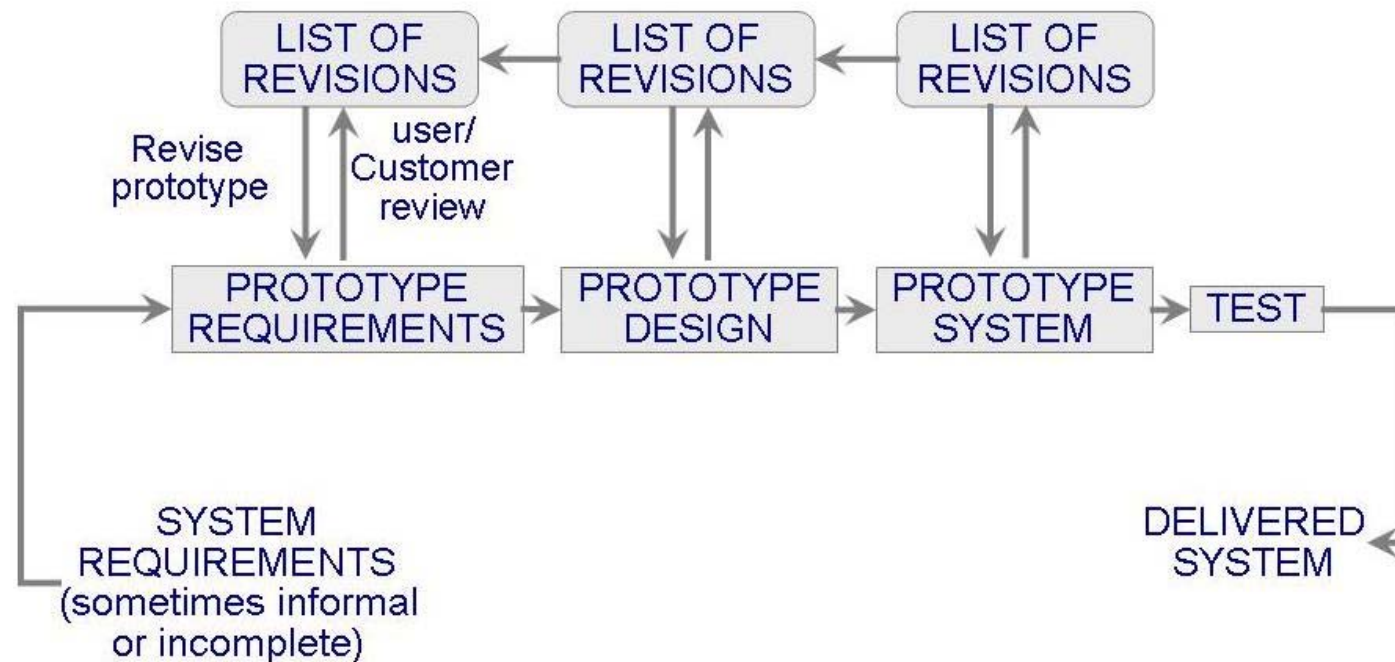
- A variation of the waterfall model (German Ministry of Defence, 1992)
- Uses unit and integration testing to verify procedural design
- Uses system testing to verify architectural (system) design
- Uses acceptance testing to validate the requirements
- If problems are found during verification and validation, the left side of the V can be re-executed before testing on the right side is re-enacted

V Model (cont.)



Prototyping Model

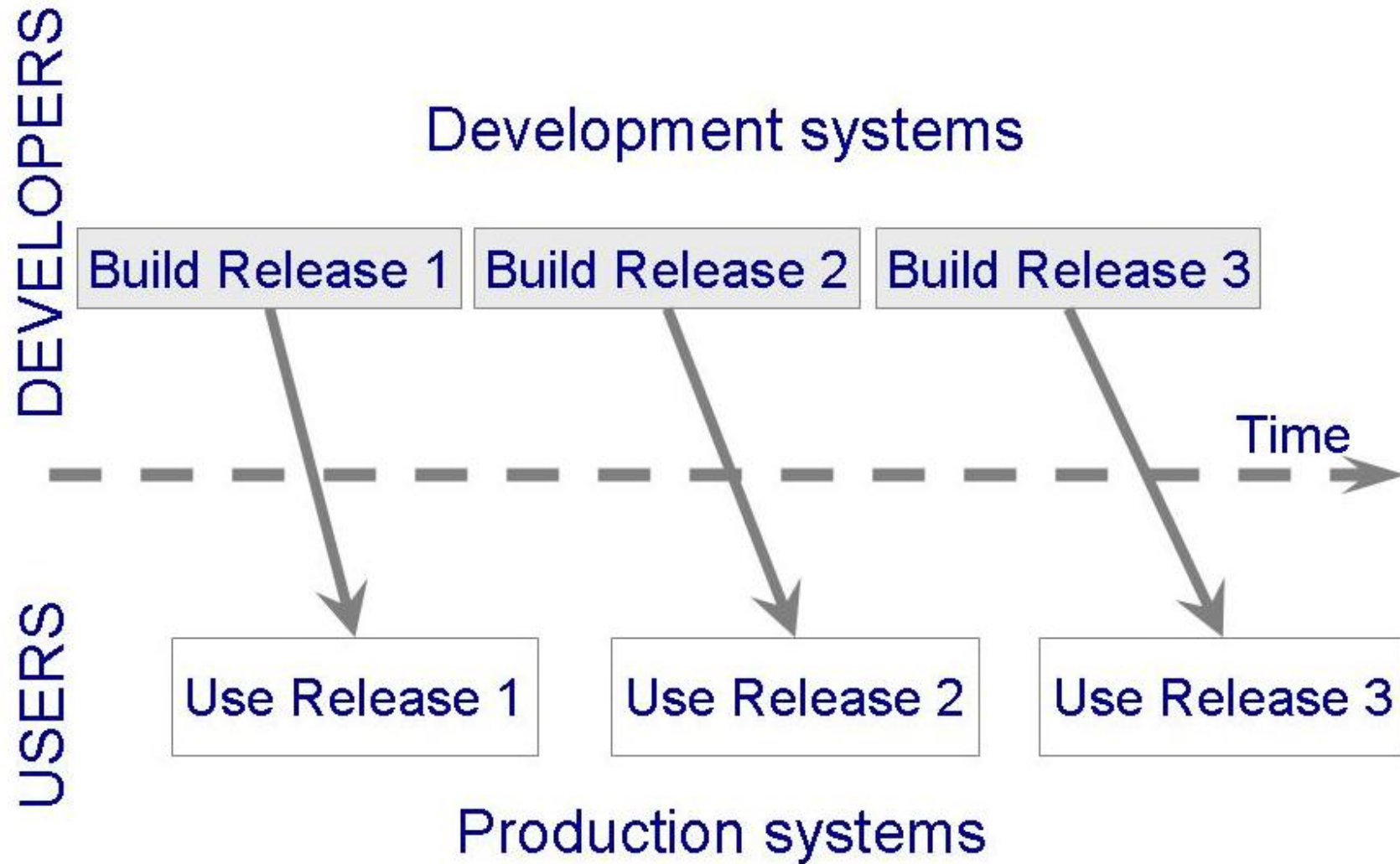
- Allows repeated investigation of the requirements or design
- Reduces risk and uncertainty in the development



Phased Development: Increments and Iterations

- Shorter cycle time
- System delivered in pieces
 - enables customers to have some functionality while the rest is being developed
- Allows two systems functioning in parallel
 - the ***production system*** (release n): currently being used
 - the ***development system*** (release $n+1$): the next version

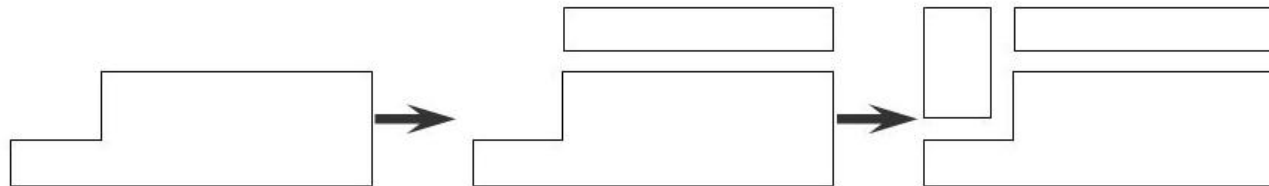
Phased Development: Increments and Iterations



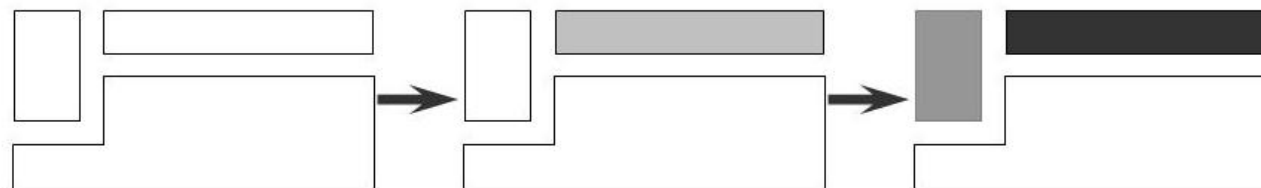
Phased Development: Increments and Iterations

- **Incremental development:** starts with small functional subsystem and adds functionality with each new release
- **Iterative development:** starts with full system, then changes functionality of each subsystem with each new release

INCREMENTAL DEVELOPMENT



ITERATIVE DEVELOPMENT



Phased Development: Increments and Iterations

- Phased development is desirable for several reasons
 - Training can begin early, even though some functions are missing
 - Markets can be created early for functionality that has never before been offered
 - Frequent releases allow developers to fix unanticipated problems globally and quickly
 - The development team can focus on different areas of expertise with different releases

Spiral Model: Adding Risk to the Waterfall

- Waterfall does not explicitly address the issue of risk.
- Suggested by B. W. Boehm in "A spiral model for software development and enhancement", 1988
- Combines development activities with risk management to minimize and control risks
 - Each Waterfall activity becomes a cycle.
 - Each cycle has four phases.
 - Supports prototyping and reuse.

Spiral Model

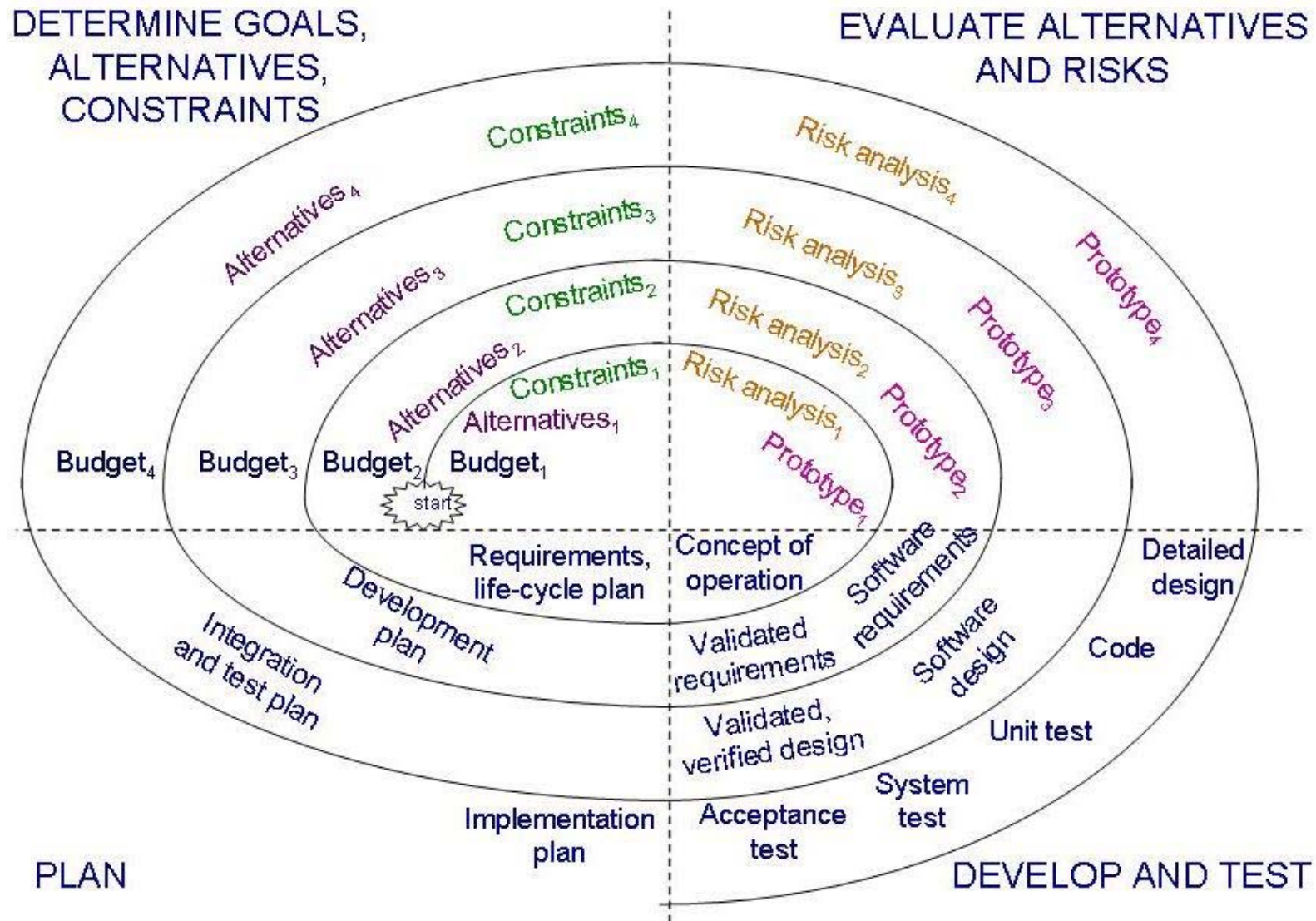


Figure 2.10 the spiral model.

Spiral Model

- The model is presented as a spiral in which each iteration is represented by a circuit around four major activities
 - Plan
 - Determine goals, alternatives, and constraints
 - Evaluate alternatives and risks
 - Develop and test

Spiral Model

- The spiral model has the following set of activities:
 - Determine objectives and constraints
 - Evaluate alternatives
 - Identify risks
 - Resolve risks by assigning priorities to risks
 - Develop a series of prototypes for the identified risks starting with the highest risk
 - Use a waterfall model for each prototype development
 - If a risk has successfully been resolved, evaluate the results of the round and plan the next round
 - If a certain risk cannot be resolved, terminate the project immediately

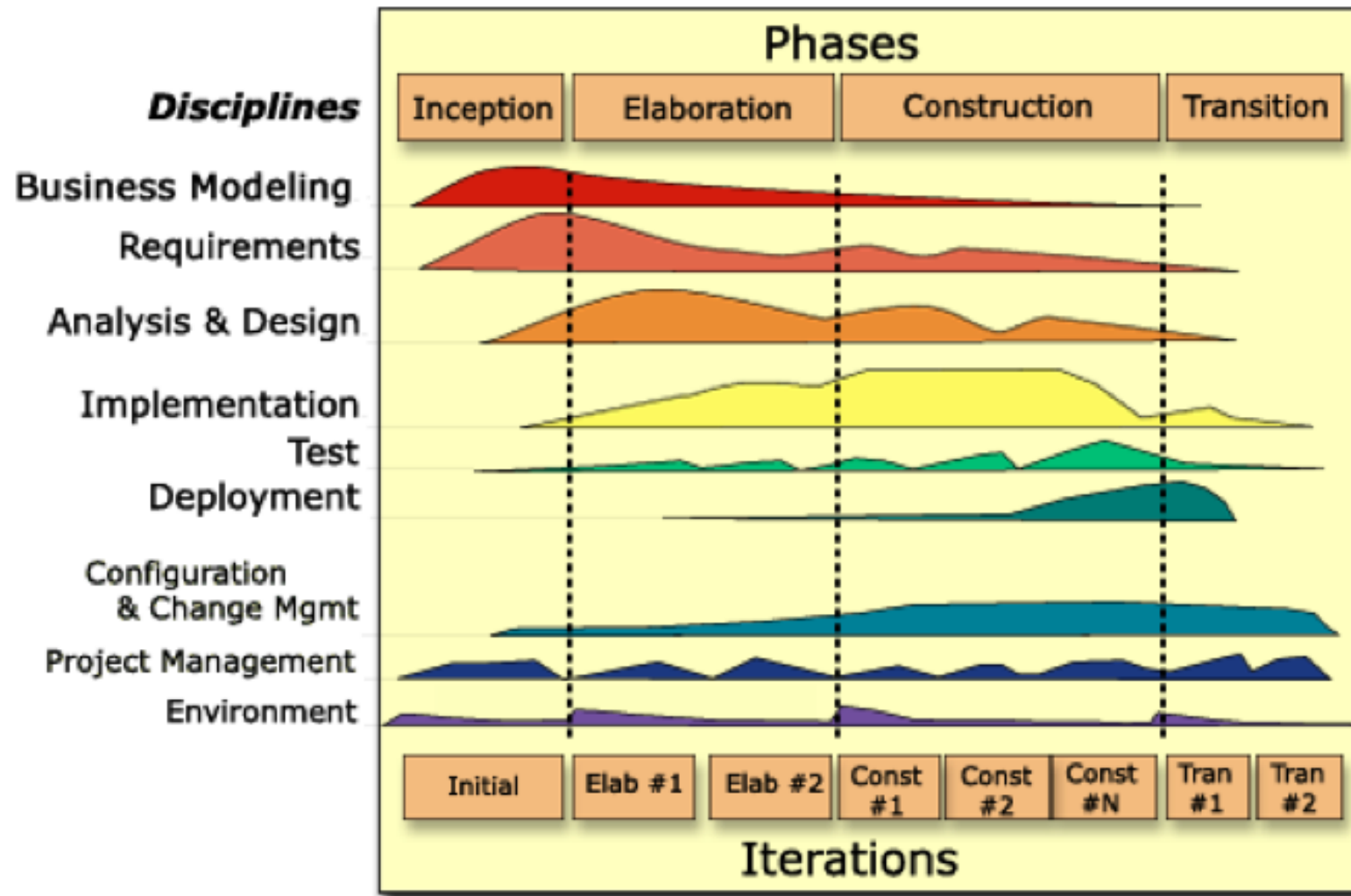
Limitations of Waterfall and Spiral Models

- Neither of these models deal well with frequent changes
 - The Waterfall model assumes that once you are done with a phase, all issues covered in that phase are closed and cannot be reopened
 - The Spiral model can deal with change between phases, but does not allow change within a phase
- What do you do if change is happening more frequently?
 - “The only constant is the change”

Rational Unified Process

- Rational Unified Process (RUP): Iterative, incremental development methodology
- RUP is structured along two dimensions:
 - Time: phases and iterations
 - Process components: artifacts + activities
- Tied in with UML and Use Cases
 - UML: specify, visualize and document artifacts
 - Use cases: link different artifacts using UML

RUP in Diagram



Workflows

- 6 engineering workflows:
 - Business Modelling
 - Requirements
 - Analysis & Design
 - Implementation
 - Test
 - Deployment
- 3 supporting workflows
 - Project management
 - Configuration and change management
 - Environment

Inception

- Milestone: Lifecycle Objective
- Primary objective: scope the system
- Develop business cases
 - Success factors
 - Risk analysis
- Develop primary use cases
 - Assist in scope-agreement between stakeholders

Elaboration

- Milestone: Lifecycle Architecture
- Primary objective: mitigate key risk items
- Majority of use cases completed
- Create executable architecture
- Address significant use cases
- Build prototype to evaluate known risks or discover new risks
- Revise business and development risks

Construction

- Milestone: Initial Operational Capability
- Primary objective: build the software system
- Majority of coding done
- May divide along use case lines to enhance modularity and incremental integration
- Creates first external software release for testing and deployment
- Multiple iterations may refer to different sets of components within the architecture

Transition

- Milestone: Product Release
 - Development ends if successful
- Primary objective: deploy to end users and maintainers
 - Validates users' expectations
- Train end users and maintainers
- Verify against quality level from Inception phase
 - Restart cycle if verification fails

Major Ideas

- Software life cycle is a process that involve building a software
- Modeling the software life cycle:
 - Sequential models
 - The waterfall model
 - V-model
 - Prototyping
 - Iterative models
 - Boehm's spiral model
 - Rational Unified Process
 - Phased Development: Increments and Iterations

Readings

- The following readings might stimulate you to learn more about software life cycle models:
 - P. Kruchten, *The Rational Unified Process – An Introduction*, 3rd Edition, Addison Wesley, 2003
 - I. Jacobson, G. Booch, J. Rumbaugh, *The Unified Software Development Process*, Addison Wesley, 1999
 - S.H. Pfleeger, J.M. Atlec, *Software Engineering: Theory and Practice*, 4th Edition, Pearson Education, 2010