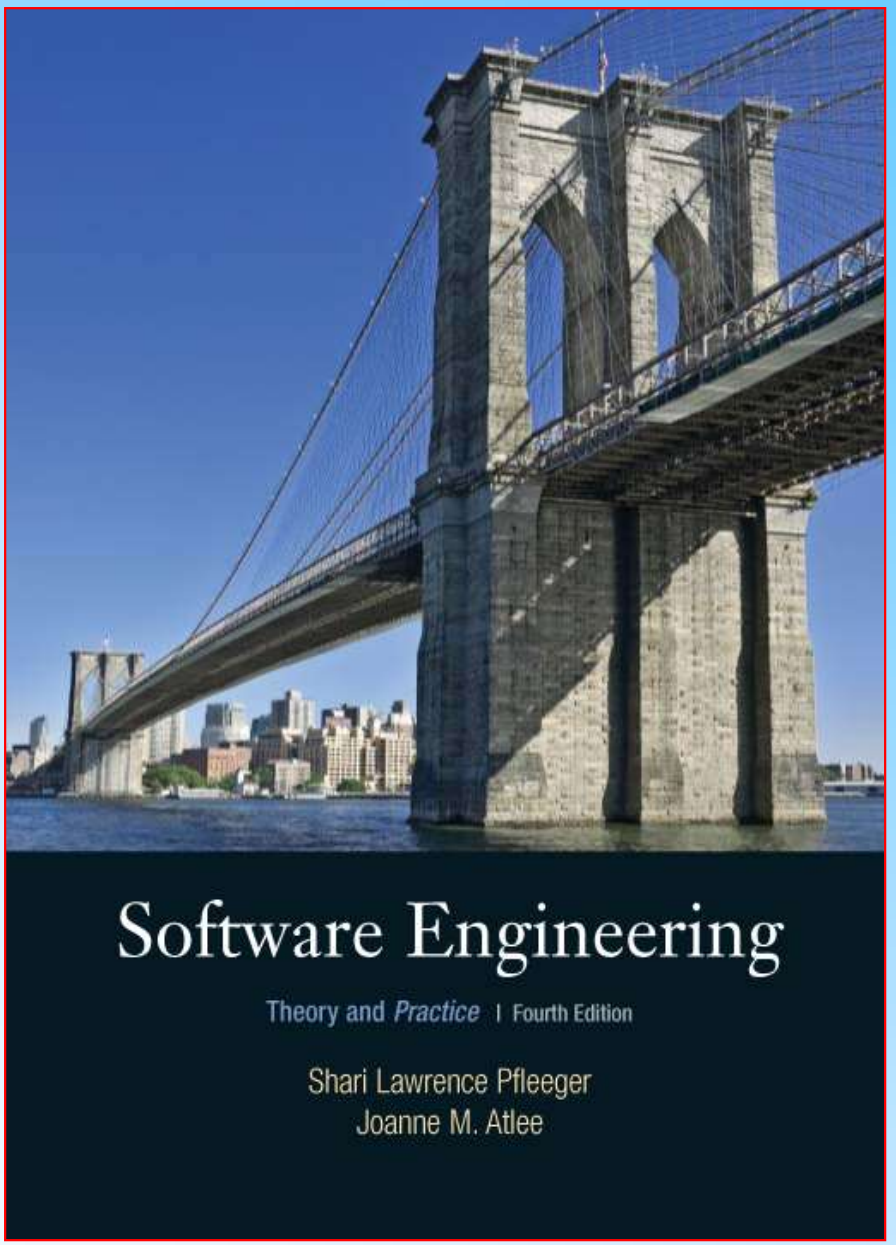
Chapter 2

Modelling the Process and Life Cycle



Contents

2.1 The Meaning of Process
2.2 Software Process Models
2.3 Tools and Techniques for Process Modeling
2.4 Practical Process Modeling
2.5 Information System Example
2.6 Real Time Example
2.7 What this Chapter Means for You

2.2 Software Process Models
Reasons for Modeling 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

2.2 Software Process Models
Software Life Cycle

- 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

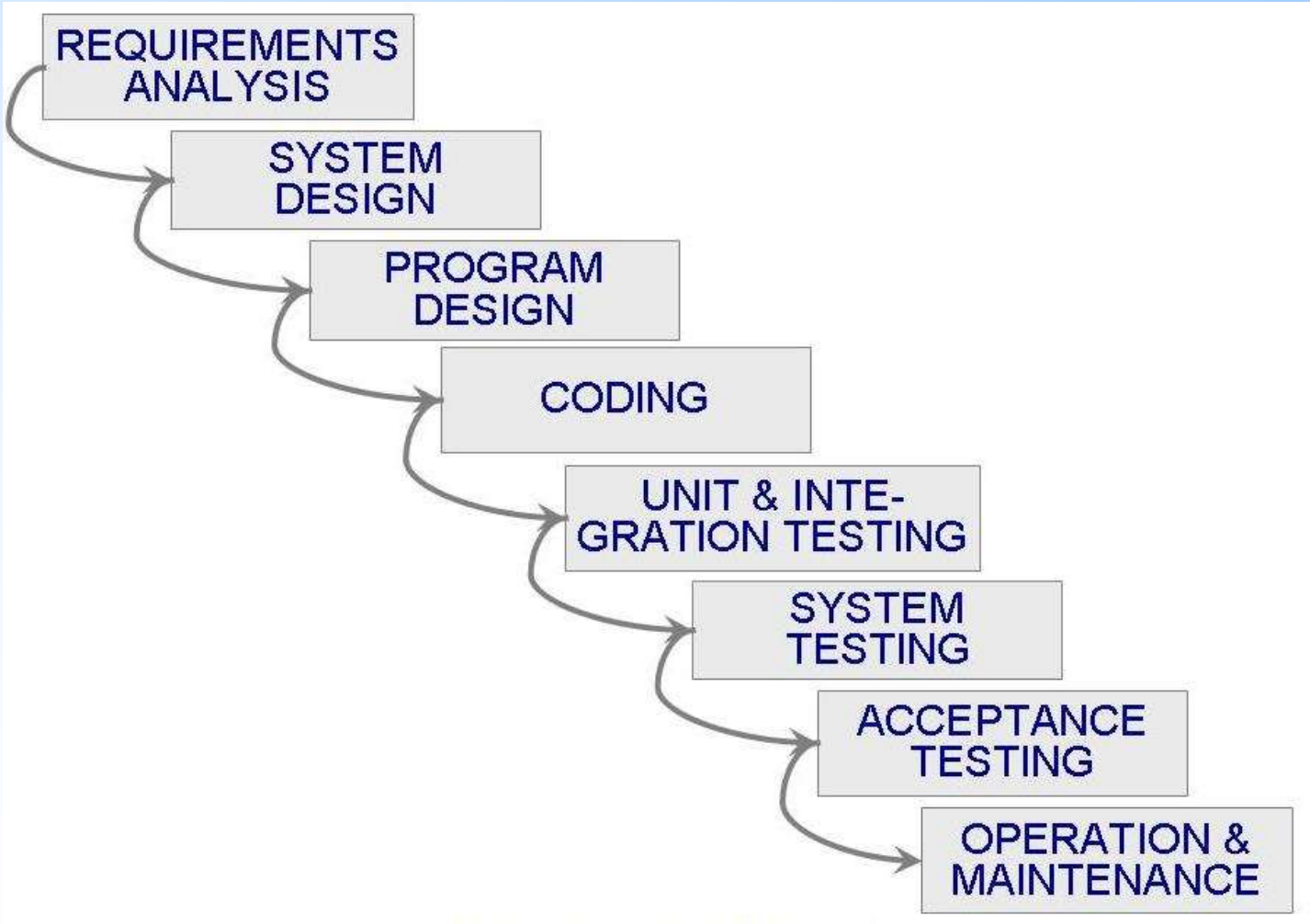
2.2 Software Process Models
Software Development Process Models

- Waterfall model
- V model
- Prototyping model
- Operational specification
- Transformational model
- Phased development: increments and iteration
- Spiral model
- Agile methods

2.2 Software Process Models
Waterfall Model

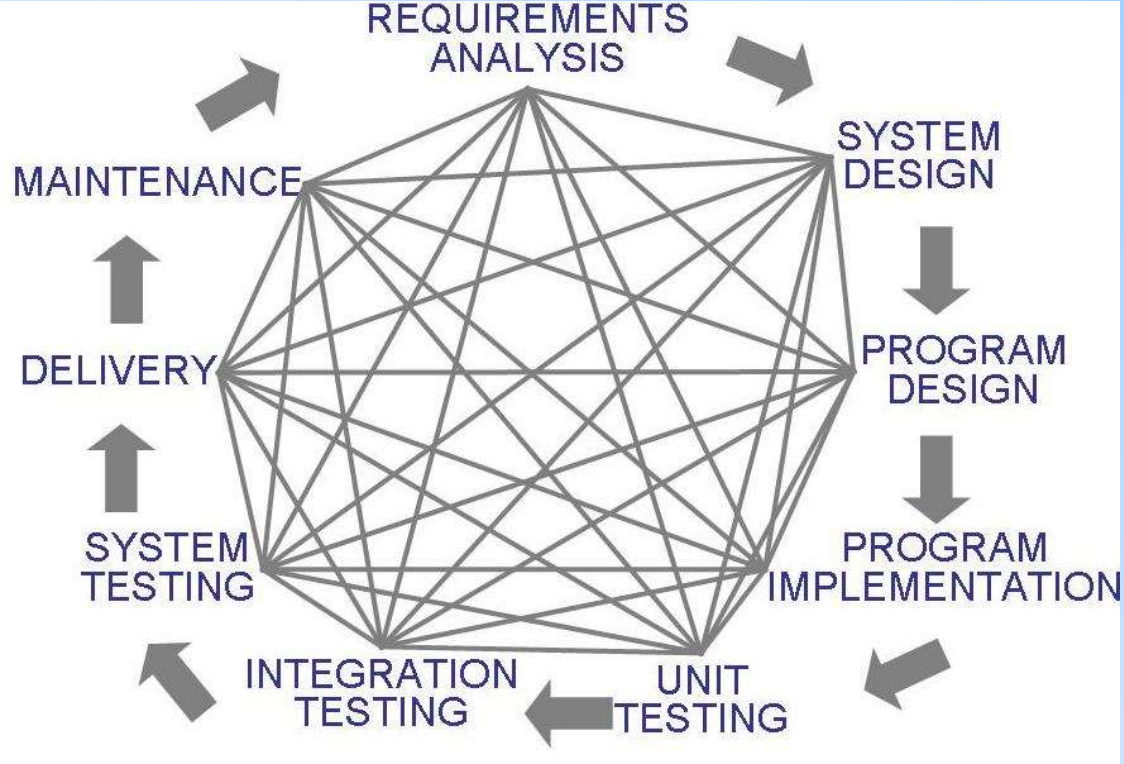
- One of the first process development models proposed
- Works for well understood problems with minimal or no changes in the requirements
- Simple and easy to explain to customers
- It presents
  - a very high-level view of the development process
  - sequence of process activities
- Each major phase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

2.2 Software Process Models
Waterfall Model (continued)



2.2 Software Process Models
Waterfall Model (continued)

- There is no iteration in waterfall model
- Most software developments apply a great amount of iterations



2.2 Software Process Models
Sidebar 2.1 Drawbacks of The Waterfall Model

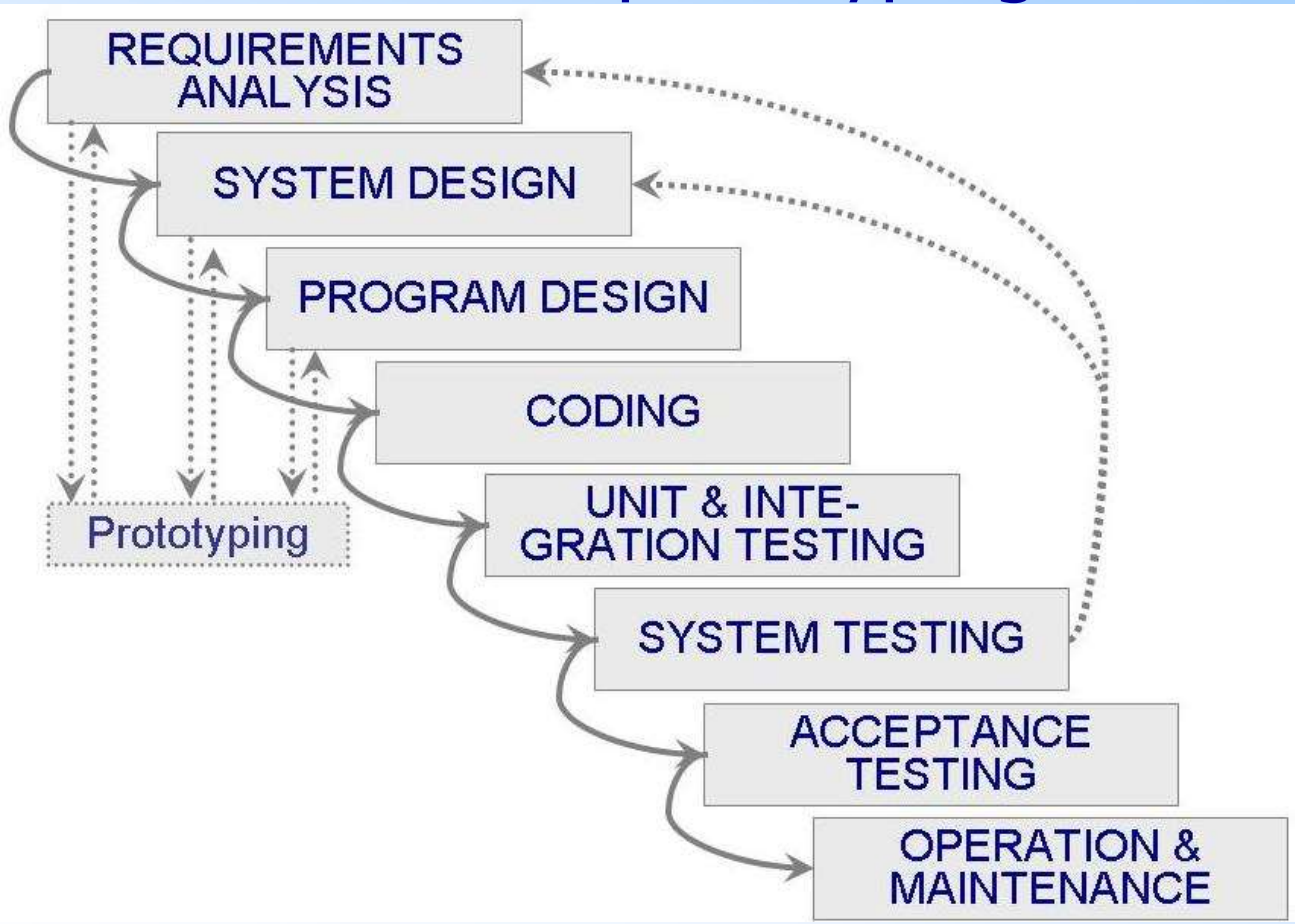
- Provides no guidance how to handle changes to products and activities during development
- Views software development as manufacturing process rather than as creative process
- There is no iterative activities (i.e. developing and evaluating prototypes) that lead to creating a final product
- Long wait before a final product

2.2 Software Process Models
Waterfall Model with Prototype

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

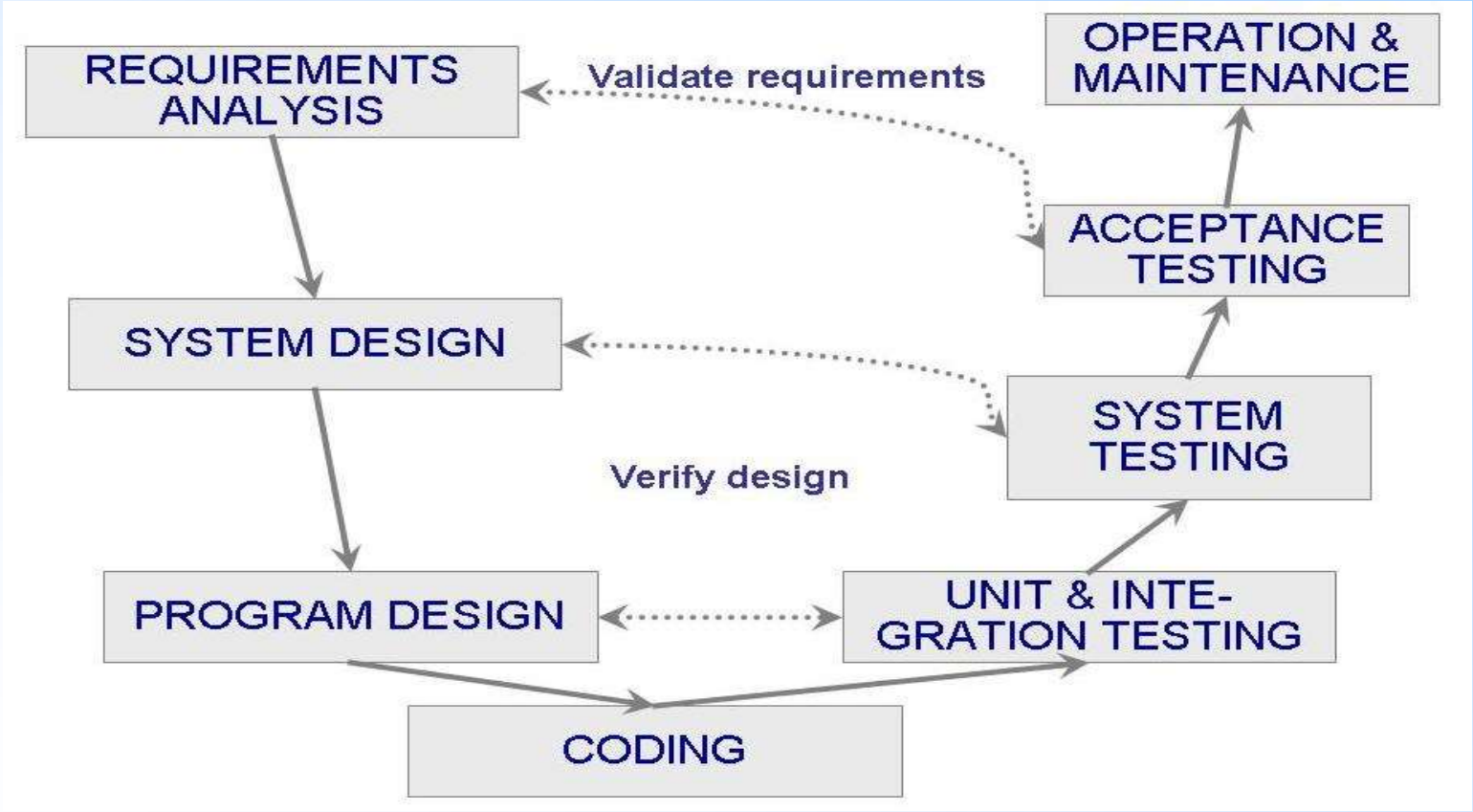
2.2 Software Process Models
Waterfall Model with Prototype (continued)

- Waterfall model with prototyping



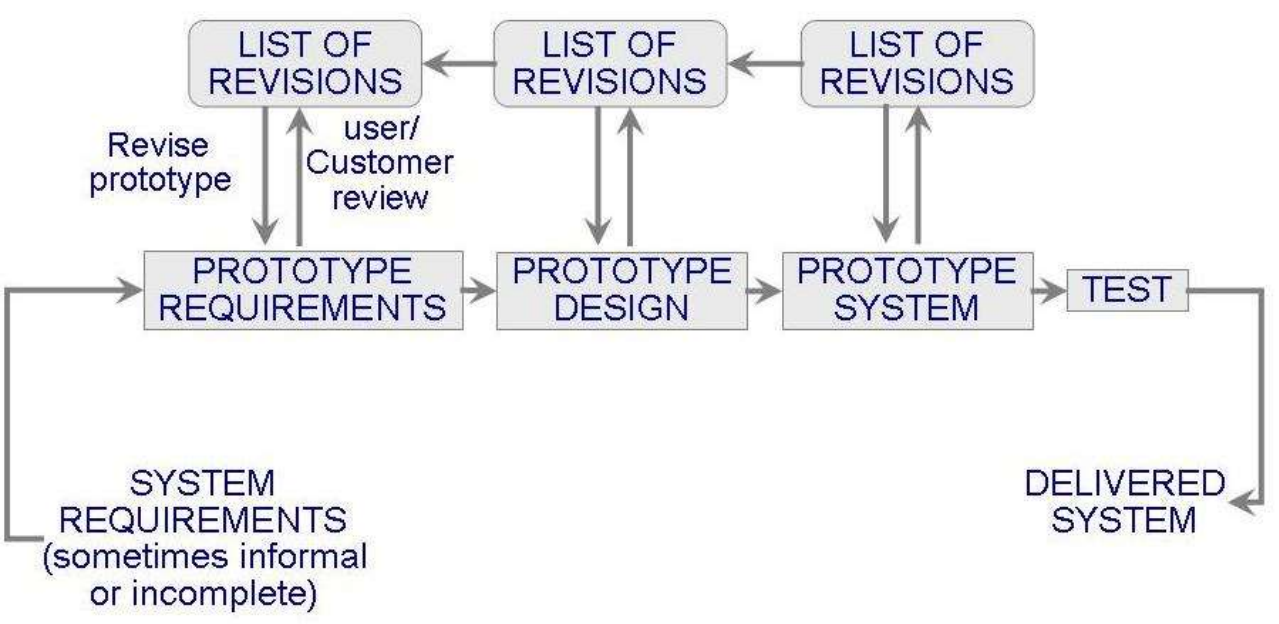
2.2 Software Process Models
V Model

- A variation of the waterfall model



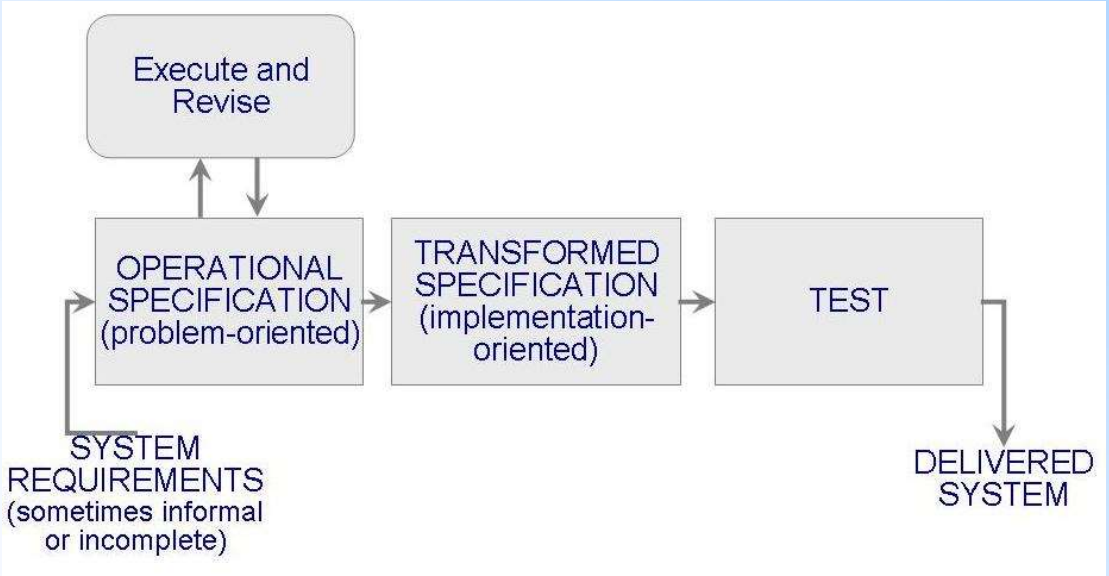
2.2 Software Process Models
Prototyping Model

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



2.2 Software Process Models
Operational Specification Model

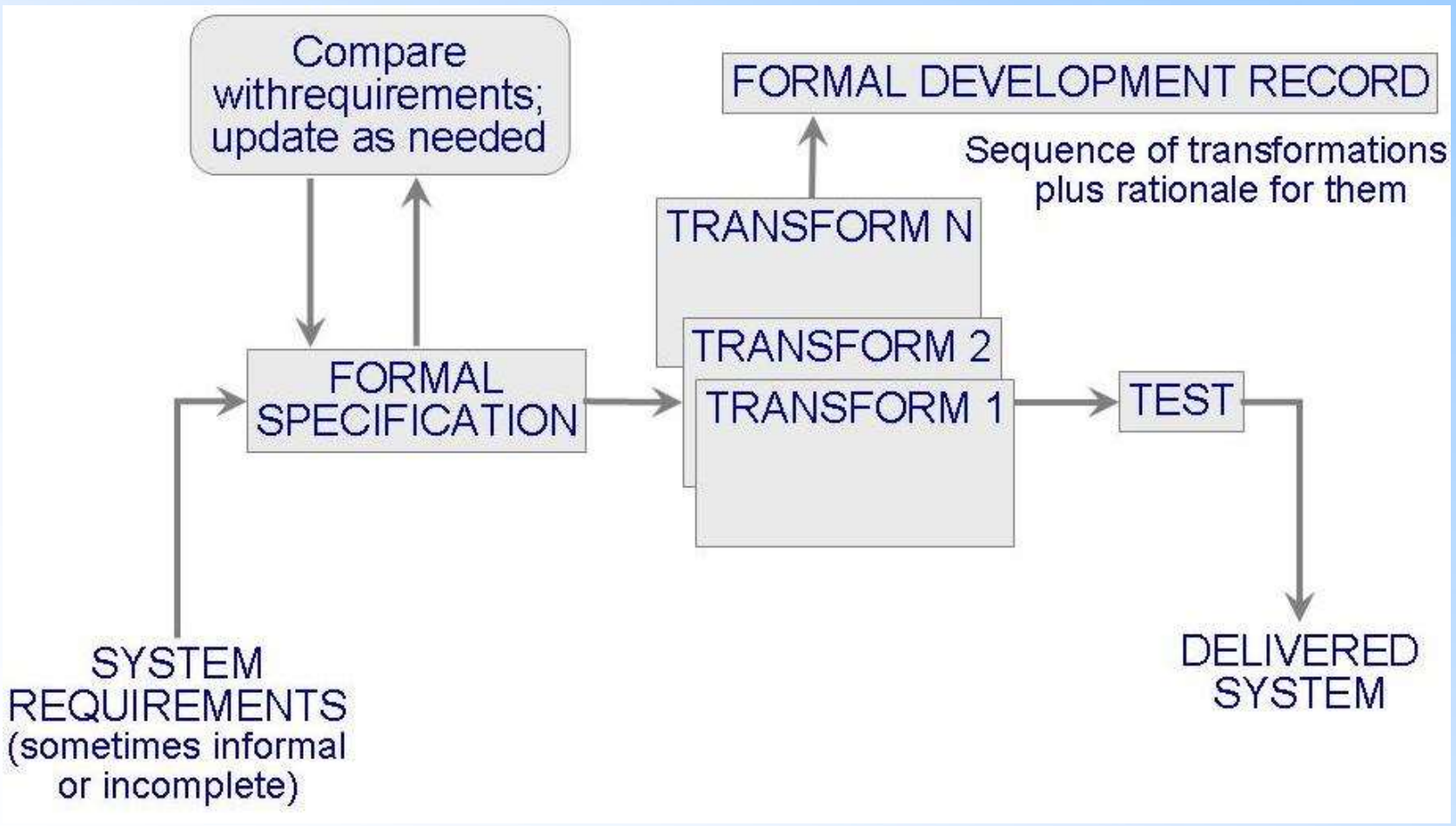
- Requirements are executed (examined) and their implication evaluated early in the development process
- Functionality and the design are allowed to be merged



2.2 Software Process Models
Transformational Model

- Fewer major development steps
- Applies a series of transformations to change a specification into a deliverable system (such as)
  - Change data representation
  - Select algorithms
  - Optimize
  - Compile
- Relies on formalism
- Requires formal specification to allow transformations

2.2 Software Process Models
Transformational Model (continued)

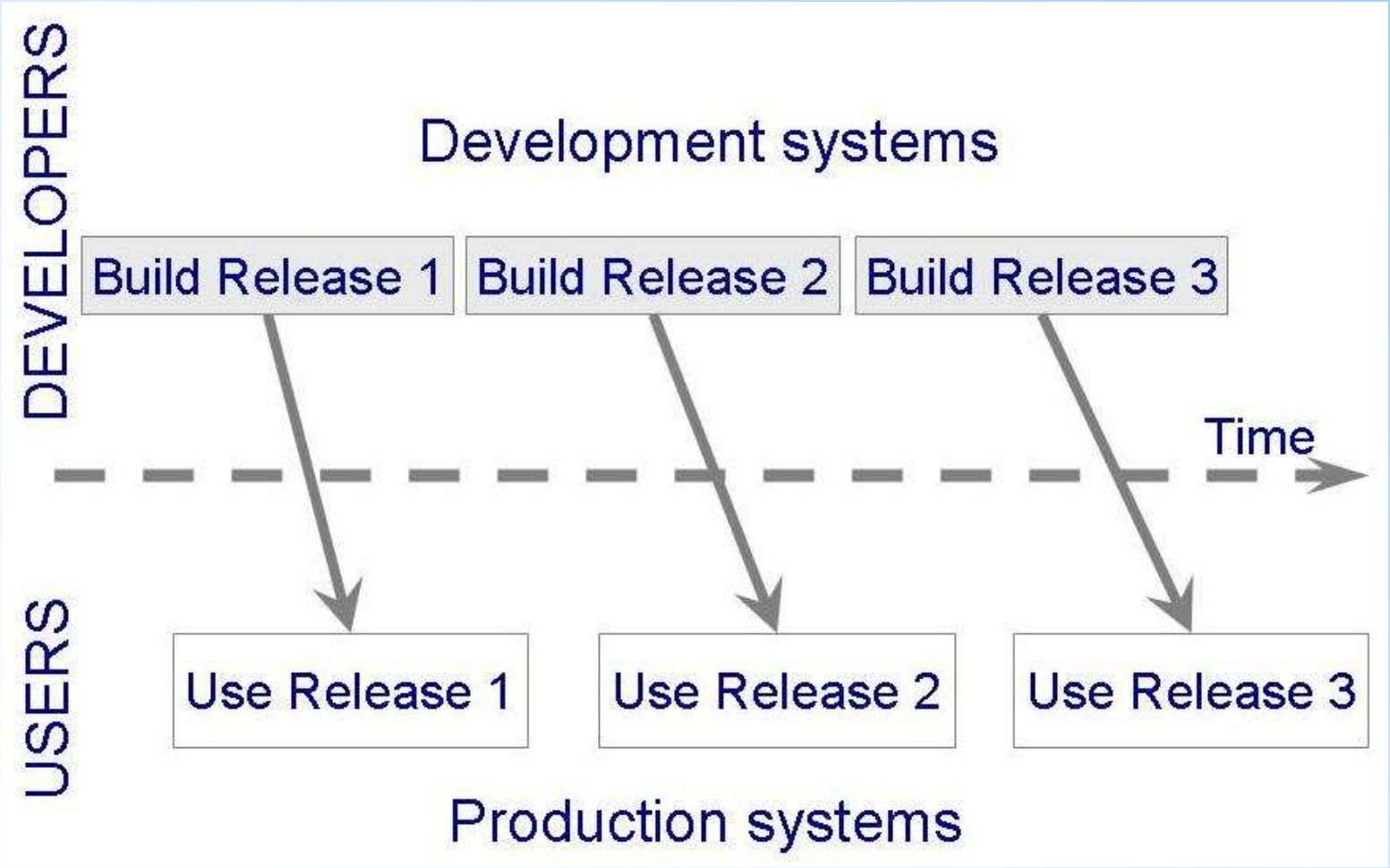


2.2 Software Process Models
Phased Development: Increments and Iterations

- 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

2.2 Software Process Models

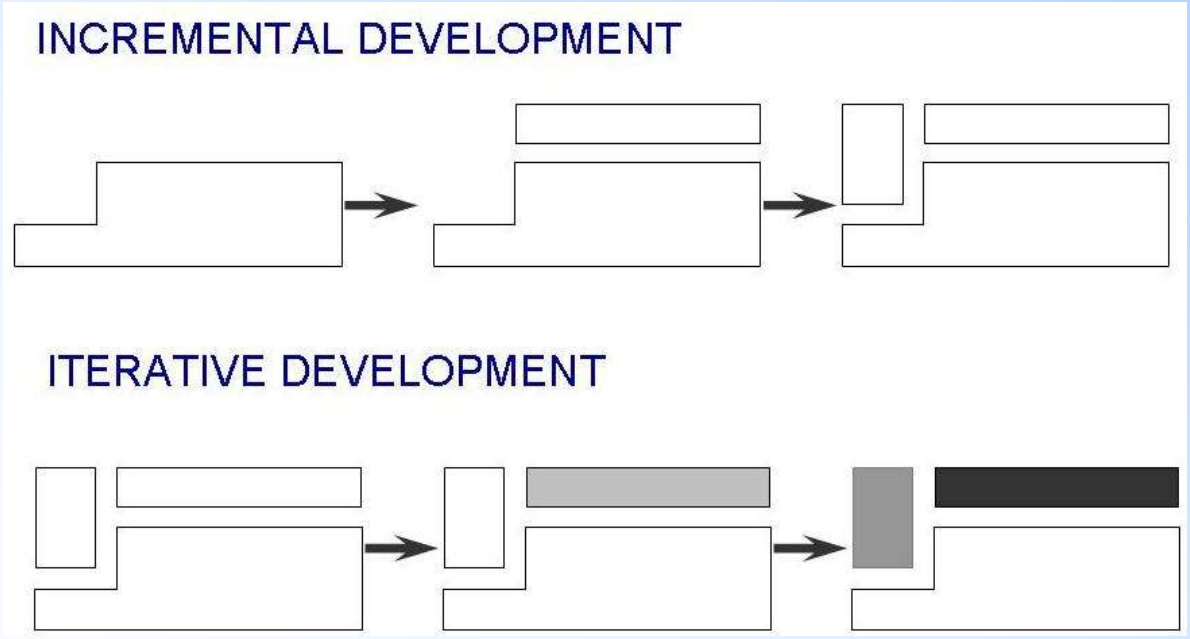
Phased Development: Increments and Iterations (continued)



2.2 Software Process Models

Phased Development: Increments and Iterations (continued)

- **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



2.2 Software Process Models

Phased Development: Increments and Iterations (continued)

- 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 globaly and quickly
  - The development team can focus on different areas of expertise with different releases

2.2 Software Process Models
Spiral Model

- Suggested by Boehm (1988)
- Combines development activities with risk management to minimize and control risks
- 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

2.2 Software Process Models
Spiral Model (continued)

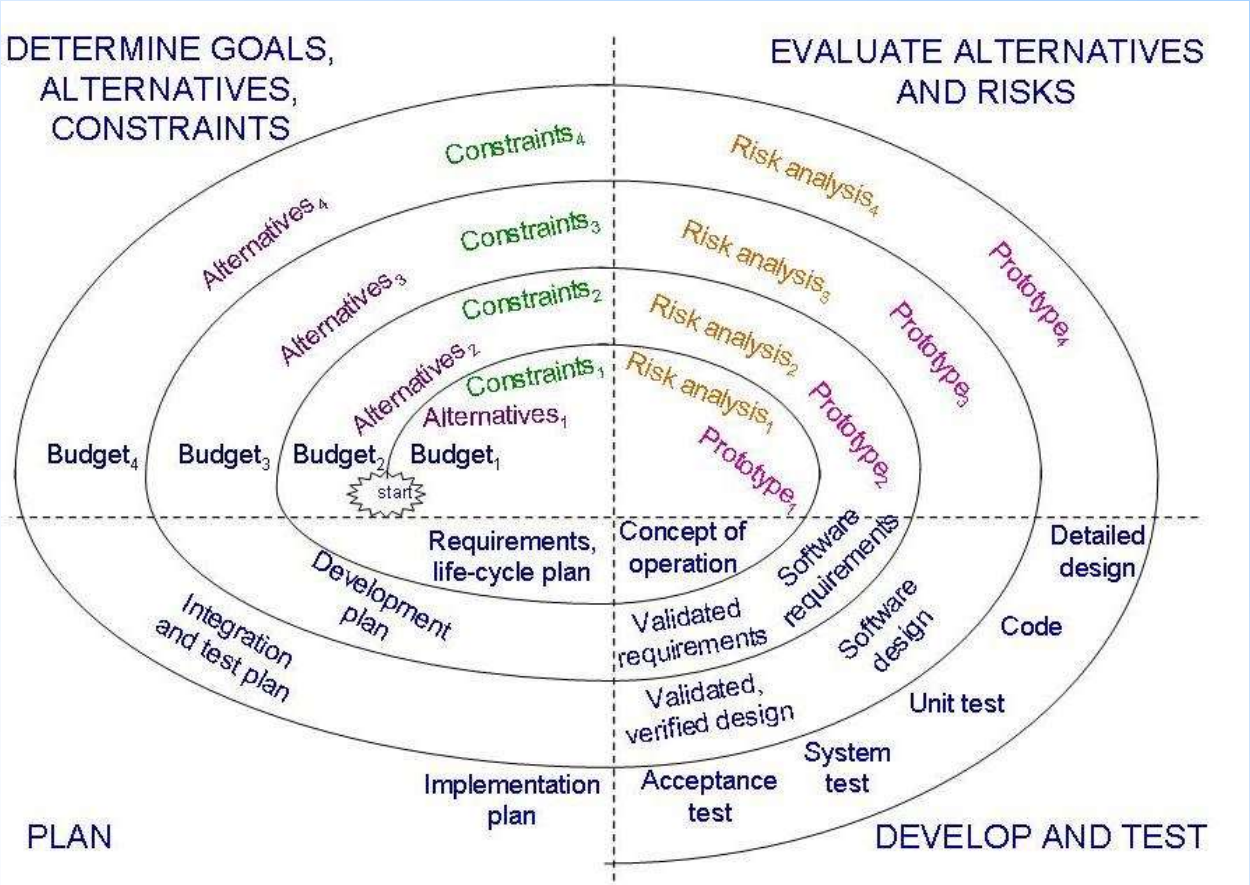


Figure 2.10 the spiral model.

2.2 Software Process Models
Agile Methods

- Emphasis on flexibility in producing software quickly and capably
- Agile manifesto
  - Value individuals and interactions over process and tools
  - Prefer to invest time in producing working software rather than in producing comprehensive documentation
  - Focus on customer collaboration rather than contract negotiation
  - Concentrate on responding to change rather than on creating a plan and then following it, since requirements change frequently.

2.2 Software Process Models
Agile Methods: Examples of Agile Process

- Extreme programming (XP)
- Crystal: a collection of approaches based on the notion that every project needs a unique set of policies and conventions
- Scrum: 30-day iterations; multiple self-organizing teams; daily “scrum” coordination

2.2 Software Process Models
Agile Methods: Extreme Programming

- Emphasis on four characteristics of agility
  - *Communication*: continual interchange between customers and developers
  - *Simplicity*: select the simplest design or implementation
  - *Courage*: commitment to delivering functionality early and often
  - *Feedback*: loops built into the various activitites during the development process

2.2 Software Process Models
Agile Methods: Twelve Facets of XP

- The planning game *(customer defines value)*
- Small release *(incremental or iterative cycles)*
- Metaphor *(development team agrees on common vision, common names)*
- Simple design
- Writing tests first *(functional and unit)*
- Refactoring *(revisiting the requirements)*
- Pair programming *(one keyboard two people)*
- Collective ownership *(any developer can change)*
- Continuous integration *(small increments)*
- Sustainable pace *(40 hours/week)*
- On-site customer
- Coding standard

2.2 Software Process Models
Sidebar 2.2 When Extreme is Too Extreme?

- Extreme programming's practices are interdependent
  - Uncomfortable with pair programming, many developers prefer to do some design documents before they write code.
- Requirements expressed as a set of test cases must be passed by the software
  - System passes the tests but is not what the customer is paying for (moving away from goals)
- Refactoring issue
  - Difficult to rework a system without degrading its architecture