软件工程概论
Software Engineering

刘伟

liuwei@xidian.edu.cn

88204608

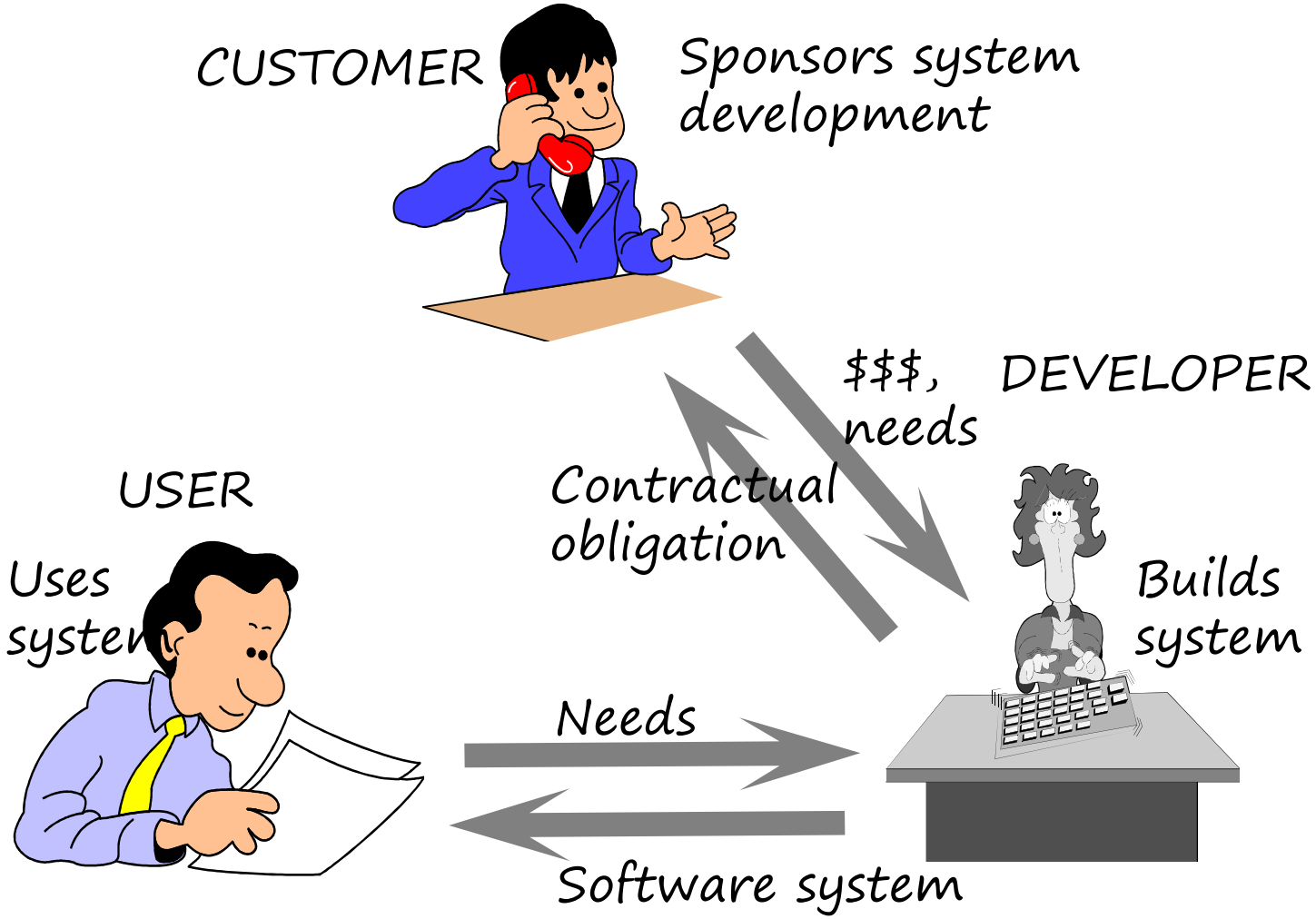
CH2. Process

Content

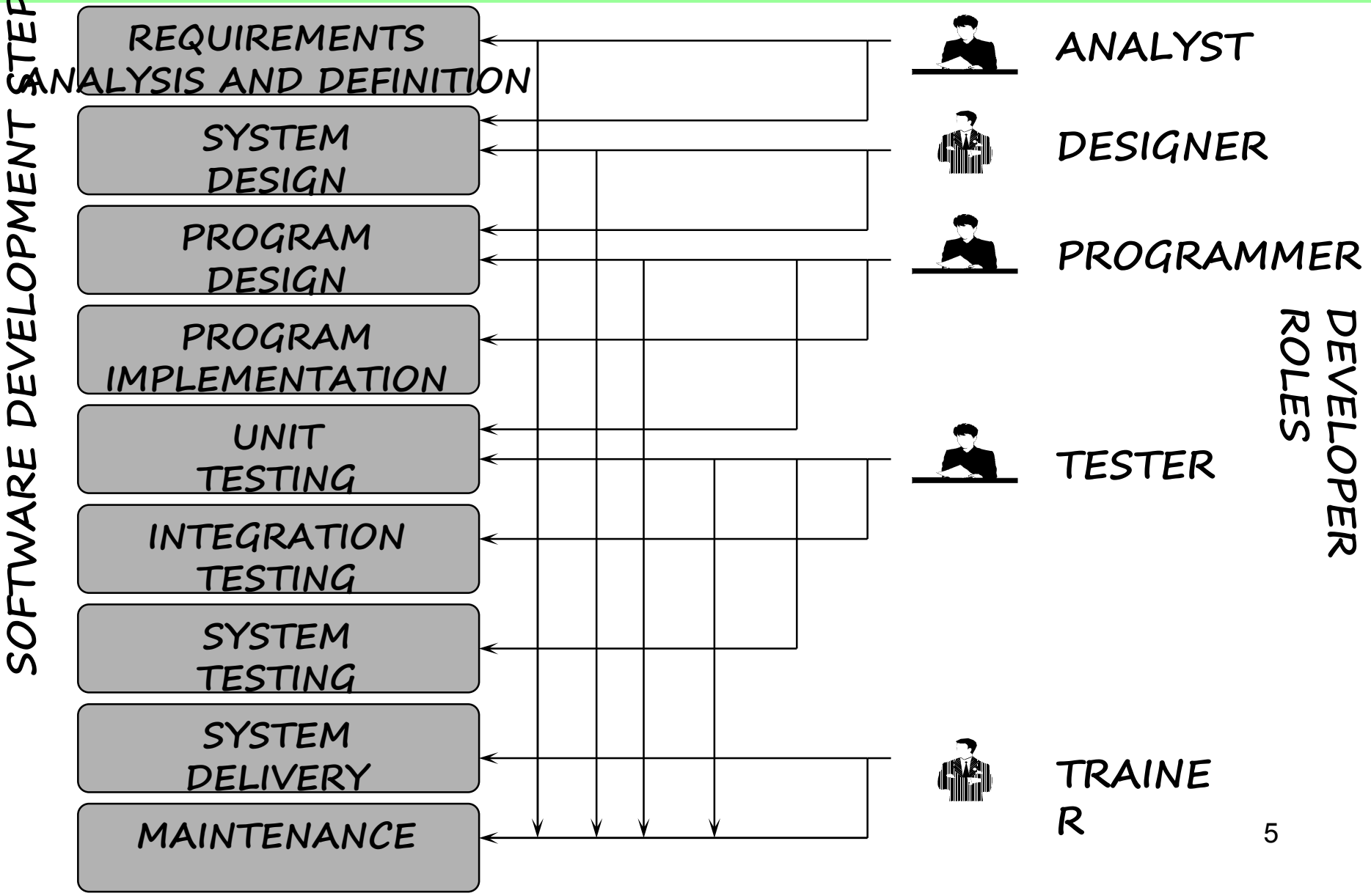
- The meaning of Process
- Software Process Models
  - Waterfall
  - V
  - Prototyping
  - Spiral models
- Tools and Techniques for Process Modeling
  - Static model
  - Dynamic model

Review

Who does software engineering



Review


SOFTWARE DEVELOPMENT STEPS
REQUIREMENTS ANALYSIS AND DEFINITION
SYSTEM DESIGN
PROGRAM DESIGN
PROGRAM IMPLEMENTATION
UNIT TESTING
INTEGRATION TESTING
SYSTEM TESTING
SYSTEM DELIVERY
MAINTENANCE
ANALYST
DESIGNER
PROGRAMMER
TESTER
TRAINER
DEVELOPER ROLES

1.1 The meaning of Process

- You can’t development a system in a single sitting
  - Complexity and Scale
  - Many people involved
- So, project has to be staged in well-defined subprojects or phases
  - Answer questions in turn: what are we doing? How are we going to do it?

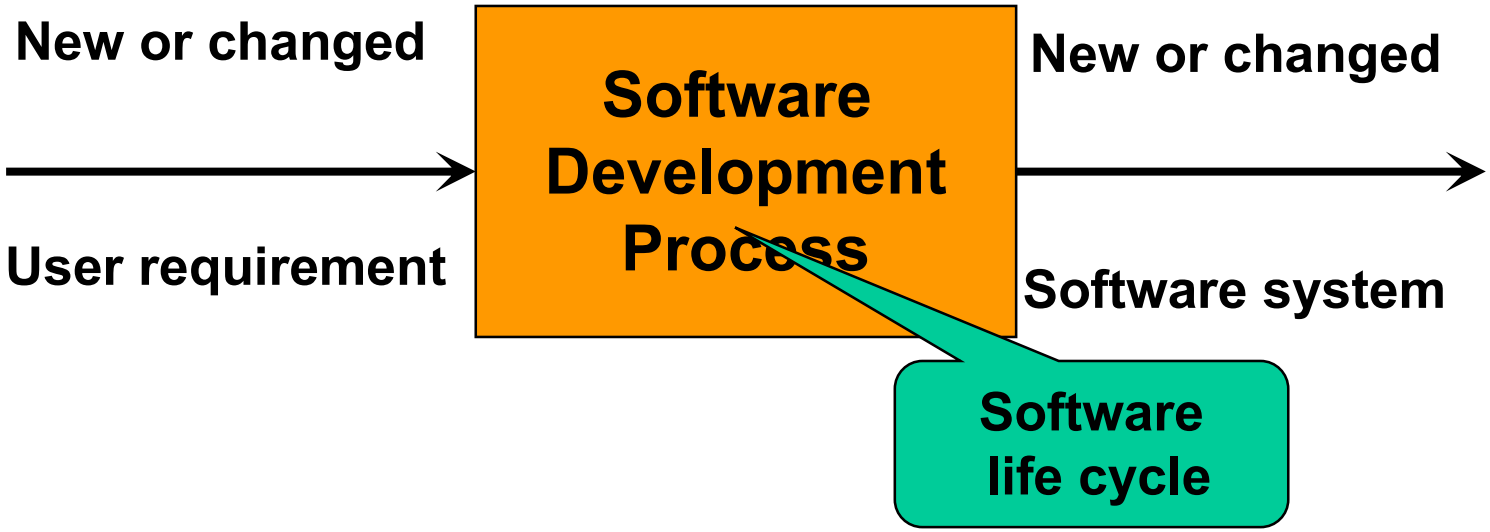
2.1 The meaning of Process

- Process
  - A set of ordered tasks
  - A series of steps involving activities, constraints, and resources that produce an intended output of some kind.
    - Roles and workflow
    - Work products
    - ***Milestones***
    - Guideline
    - ……

2.1 The meaning of Process

Definition of Software Process

- A process defines *who* is doing *what*, *when* and *how*, in order to reach a certain *goal*.

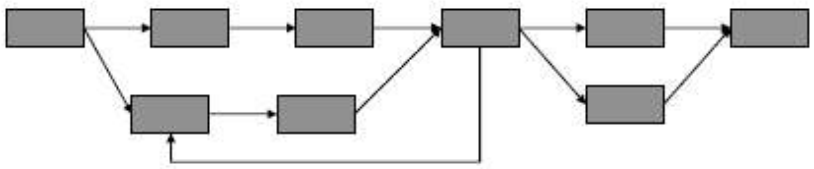


- Life cycle: *when the process involves the building of some product, we sometimes refer to the process as a life cycle.*

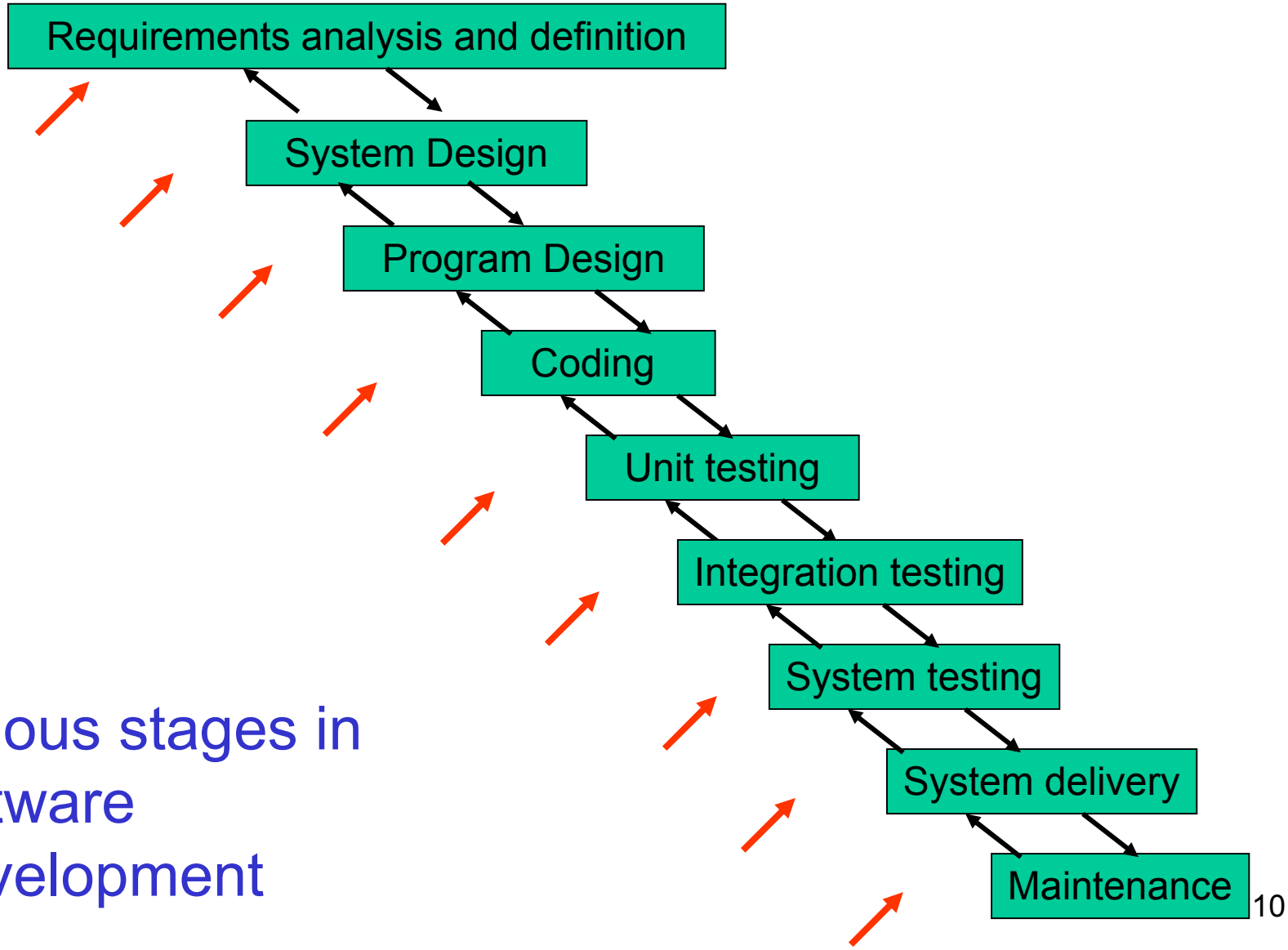
2.1 The meaning of Process

Characteristics (特征) of process

- A process usually involves a set of tools and techniques.
  - The process prescribes all of the major process activities.
  - The process uses resources, subject to a set of constraints.
  - The process may be composed of sub-processes that are linked in some way.
  - Each process activity has entry and exit criteria.
  - The activities are organized in a sequence, so that it is clear when one activity is performed relative to the other activity.
  - Every process has a set of guiding principles that explain the goals of each activity.
  - Constraints or controls may apply to an activity, resource, or product.



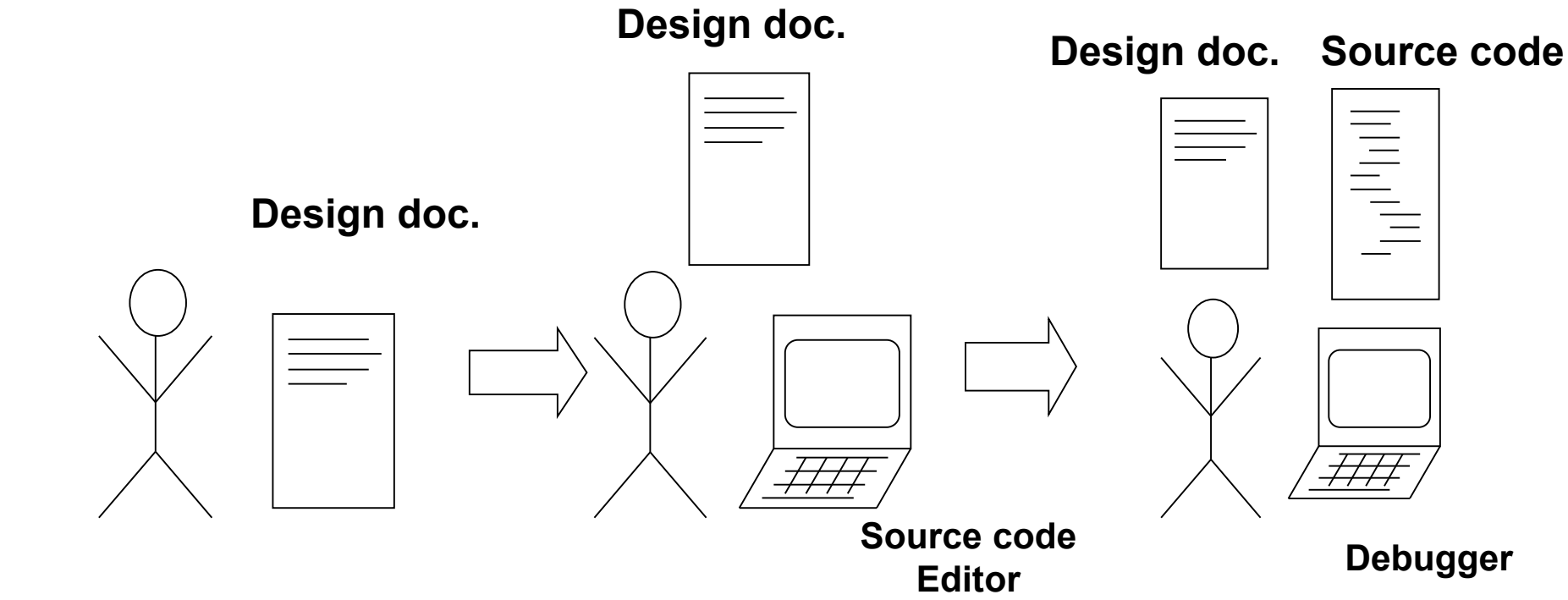
2.1 The meaning of Process



various stages in software Development

2.2 Software Process Models

- A process can be described in a variety of ways, using text, pictures or a combination.



Software Process = Sequence of development activities

2.2 Software Process Models

- In the software engineering literature, descriptions of process models are ***prescriptions*** (or the way software development should progress) or ***descriptions*** (the way software development is done in actuality).
- In theory, the two should be the same, but in practice, they are ***not***.

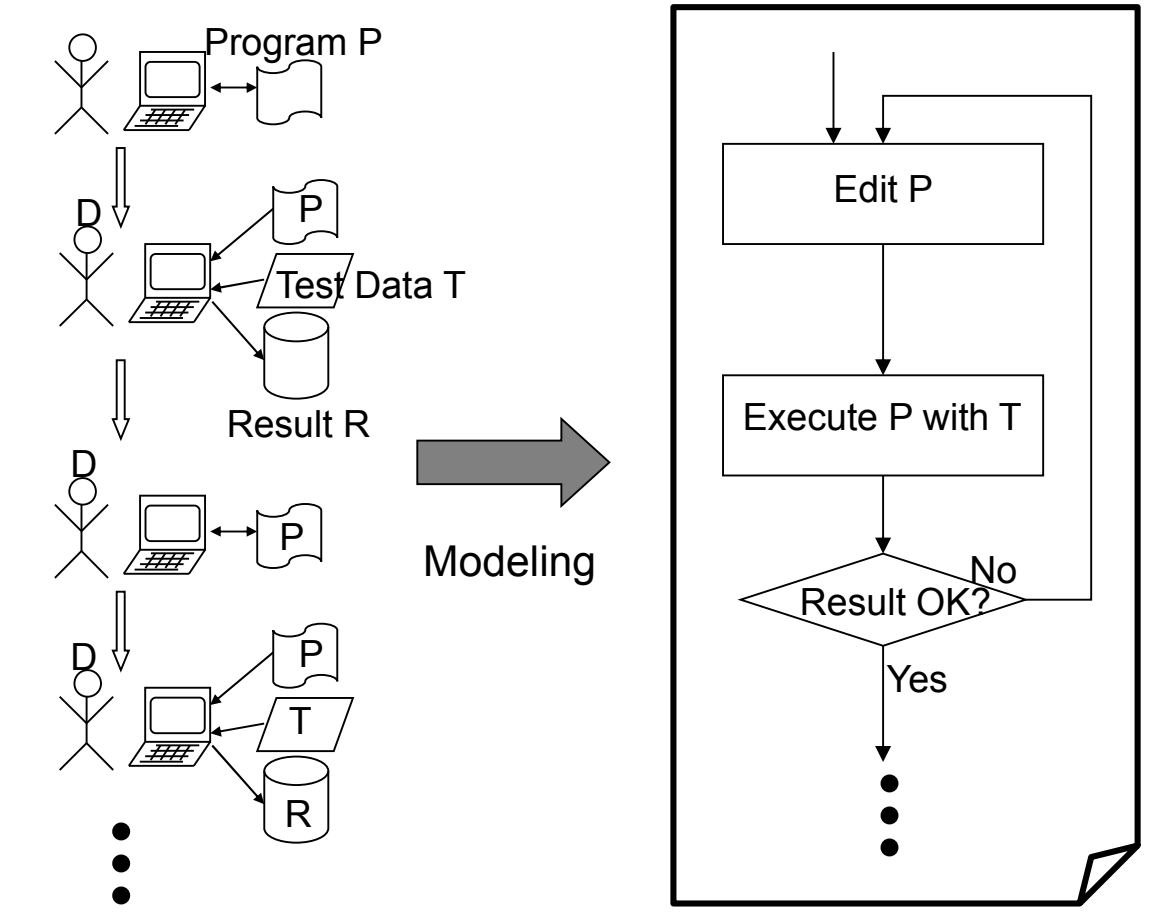
2.2 Software Process Models

Reasons of process modeling

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

2.2 Software Process Models

Process Modeling

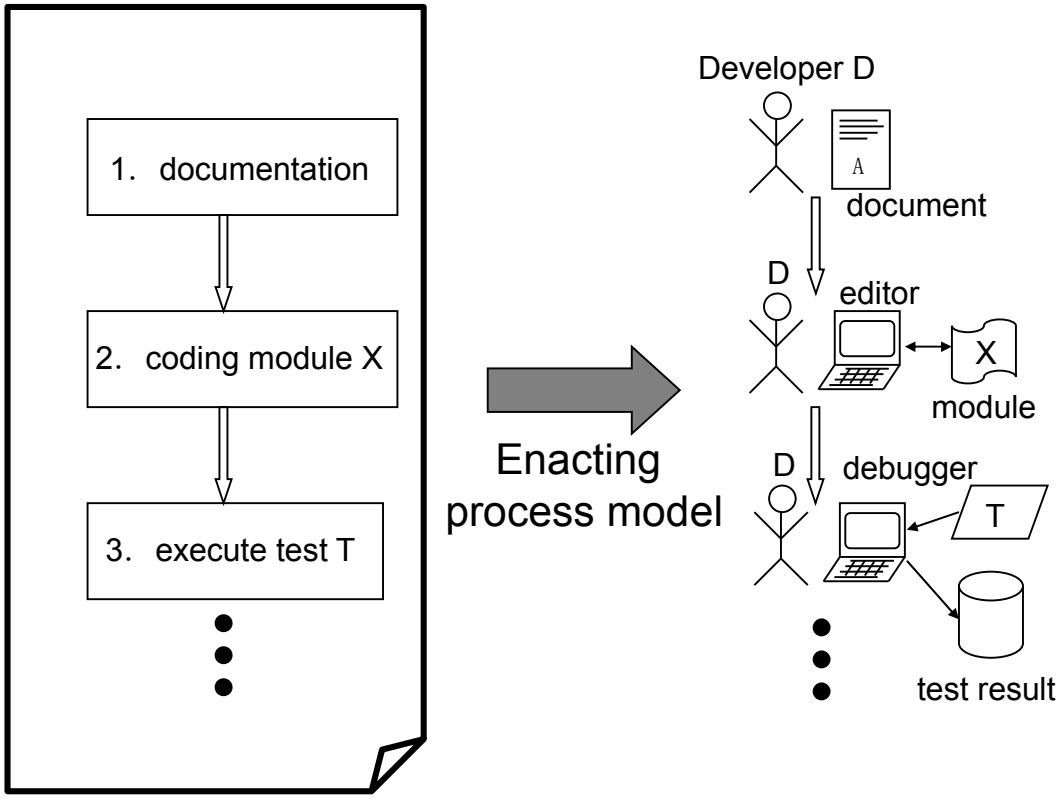


Actual Process

Process Model in Flowchart

2.2 Software Process Models

Process Enaction



Process model already written

Actual development process

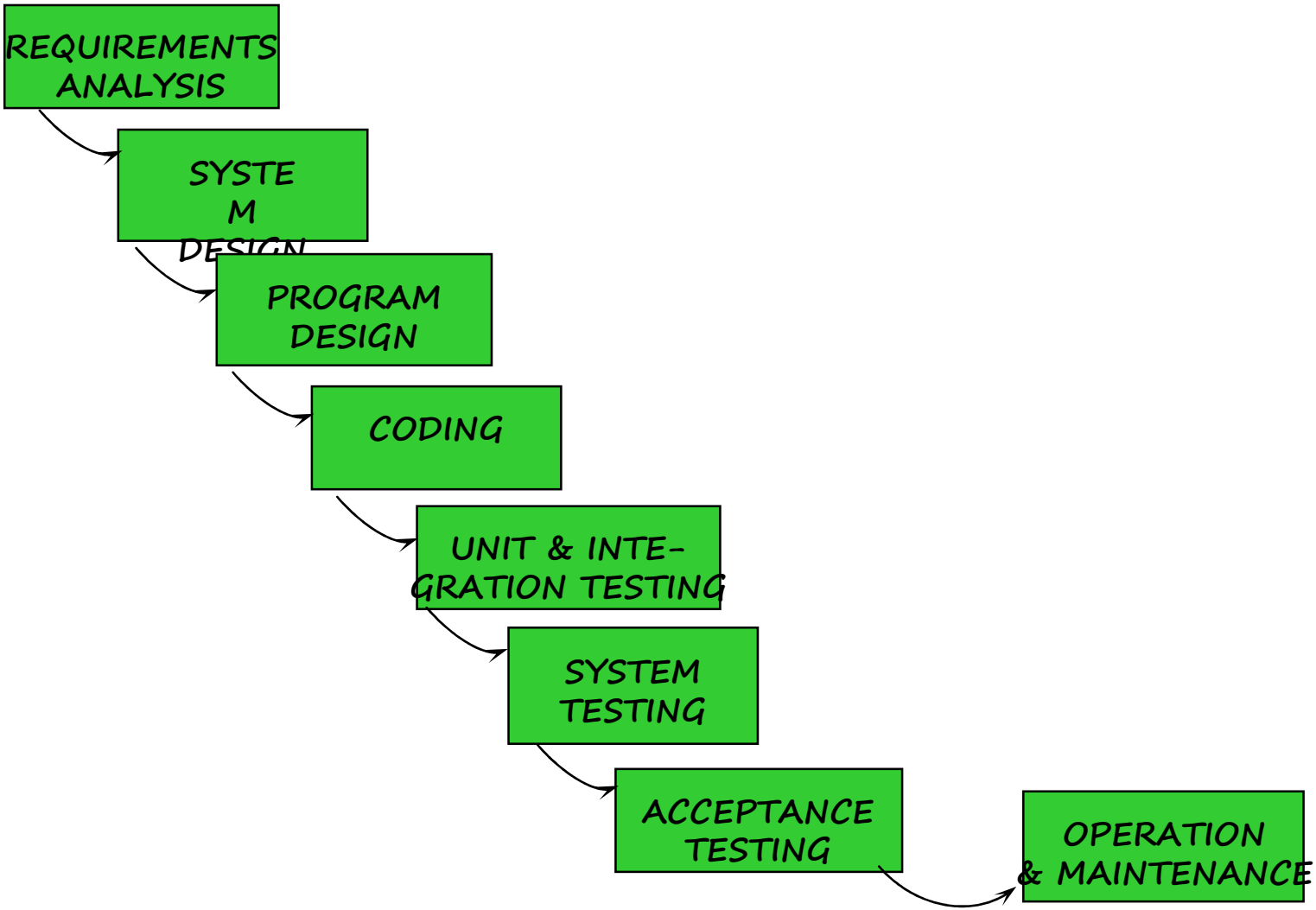
2.2 Software Process Models

Examples of process models

- Waterfall model
- Prototyping
- V-model
- Phased development: increments and iteration
- Spiral (螺旋) model

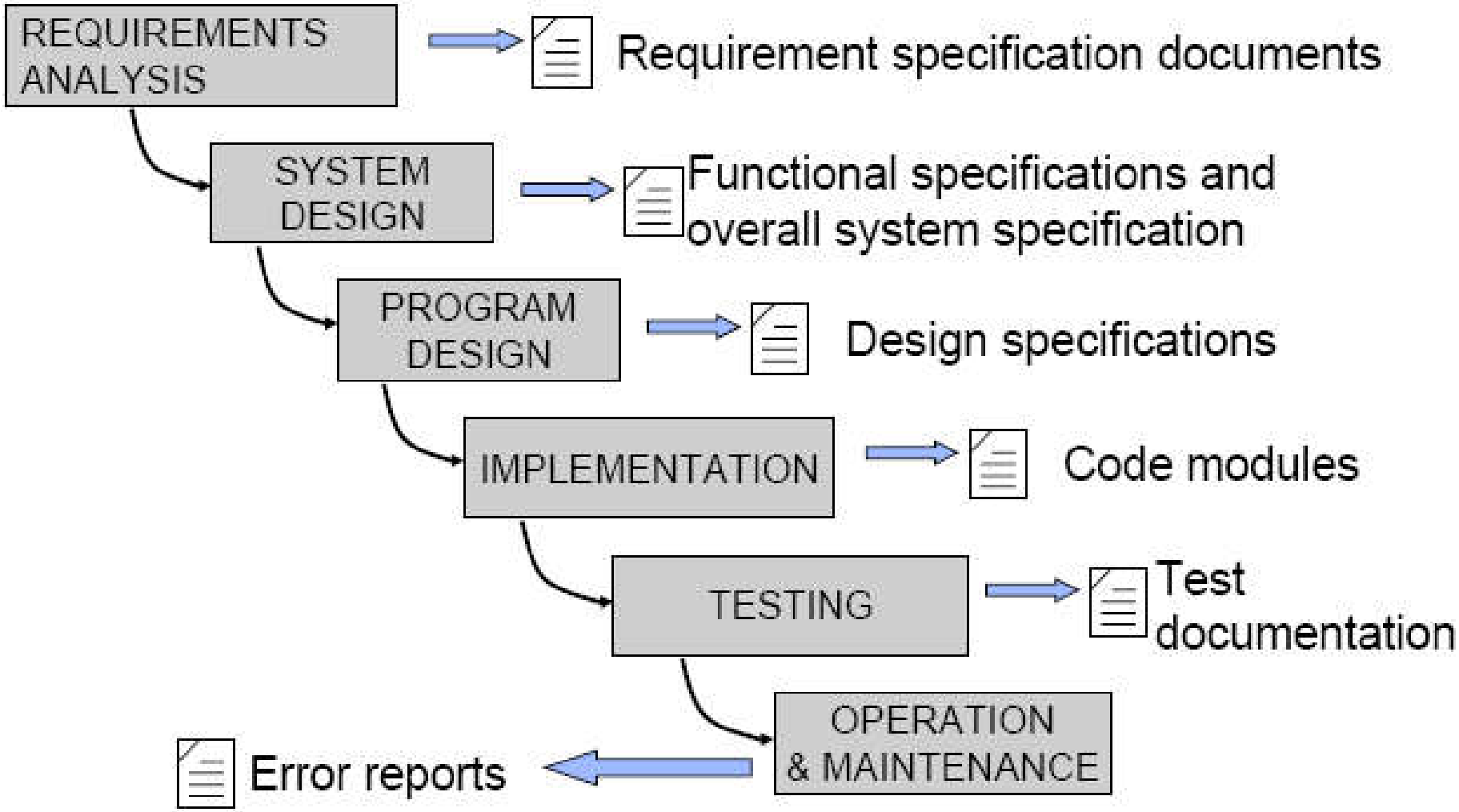
2.2 Software Process Models

Waterfall Model



2.2 Software Process Models

Waterfall Model - characteristics



2.2 Software Process Models

Waterfall model documents

Activity	Output documents
Requirements analysis	Feasibility study, Outline requirements
Requirements definition	Requirements document
System specification	Functional specification, Acceptance test plan Draft user manual
Architectural design	Architectural specification, System test plan
Interface design	Interface specification, Integration test plan
Detailed design	Design specification, Unit test plan
Coding	Program code
Unit testing	Unit test report
Module testing	Module test report
Integration testing	Integration test report, Final user manual
System testing	System test report
Acceptance testing	Final system plus documentation

2.2 Software Process Models

Waterfall Model – features

- Very well distinguished process (easy to understand)
- Every steps in the model completes with a milestone
- A milestone is defined as a completion of set of documents
- When the documents are approved the next step can be taken
- Well defined inputs and outputs of the activities
  - Interfaces between the phases are well defined
- Precisely define different ***roles*** of developers

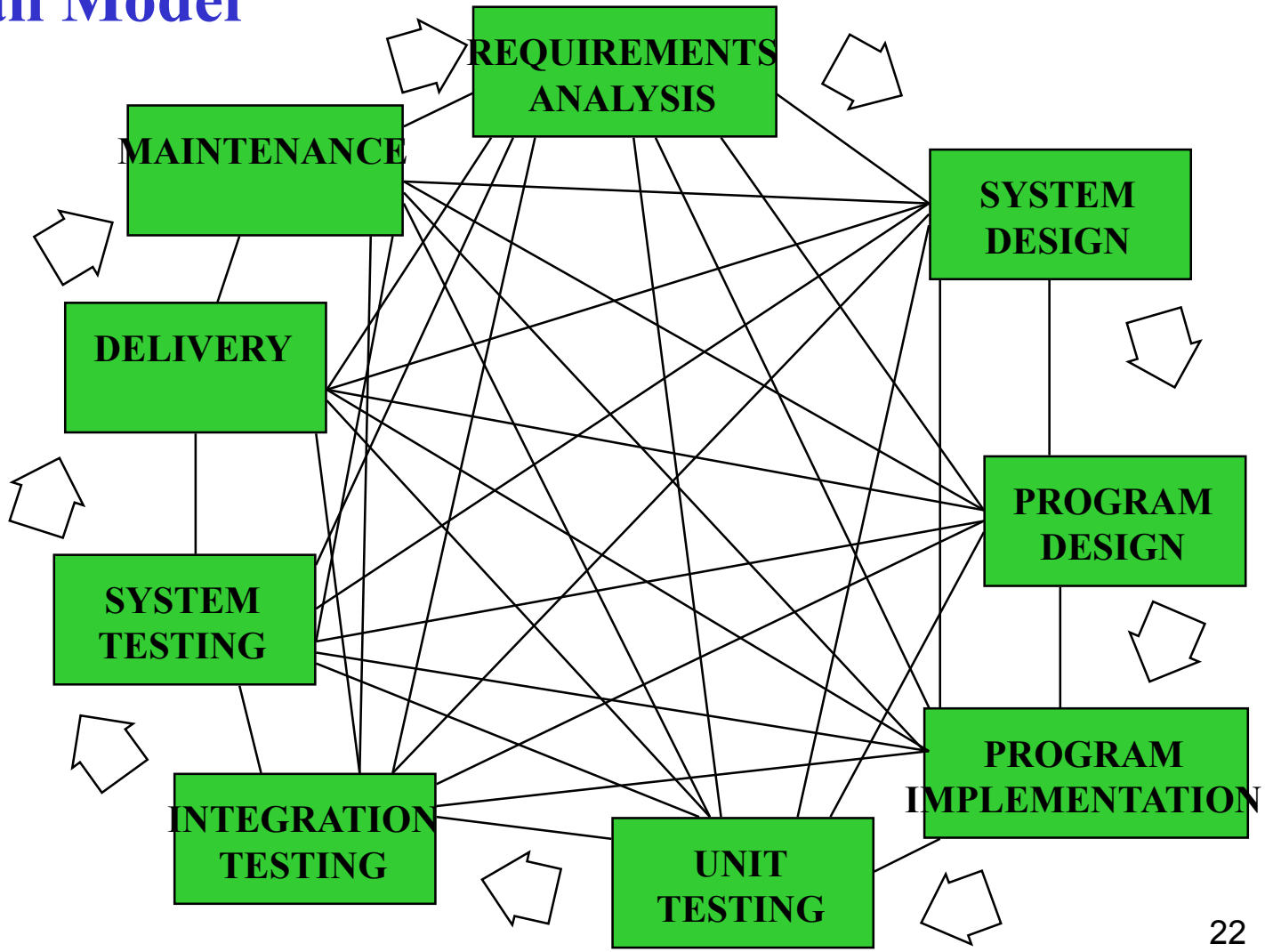
2.2 Software Process Models

Waterfall Model – drawback

- There is no insight into how each activity transform one artifact to another, such as requirements to design.
- Its failure to treat software as a problem-solving process.

2.2 Software Process Models

Waterfall Model



2.2 Software Process Models

Where the waterfall model works fine?

- Established procedures and technologies
- Not too much requirements changes
- Well organized projects with precisely defined roles
- “Repeatable” projects
- Big projects not time and budget critical

- ***What are characteristics of today's development?***
- ***How does Waterfall model fit for them?***

2.2 Software Process Models

Requirements on the development process today

- Requirements
  - Time-to market
  - User Interfaces
  - Possibility for integration with other products
  - Using standard and de-facto standard solutions
  - Using already existing components (COTS = commercial off the shelf)
- Permanent change of technologies
- Development tools, development and target platforms, hardware,
- etc.
- Requirements changes during the entire development process

2.2 Software Process Models

Waterfall model fitness

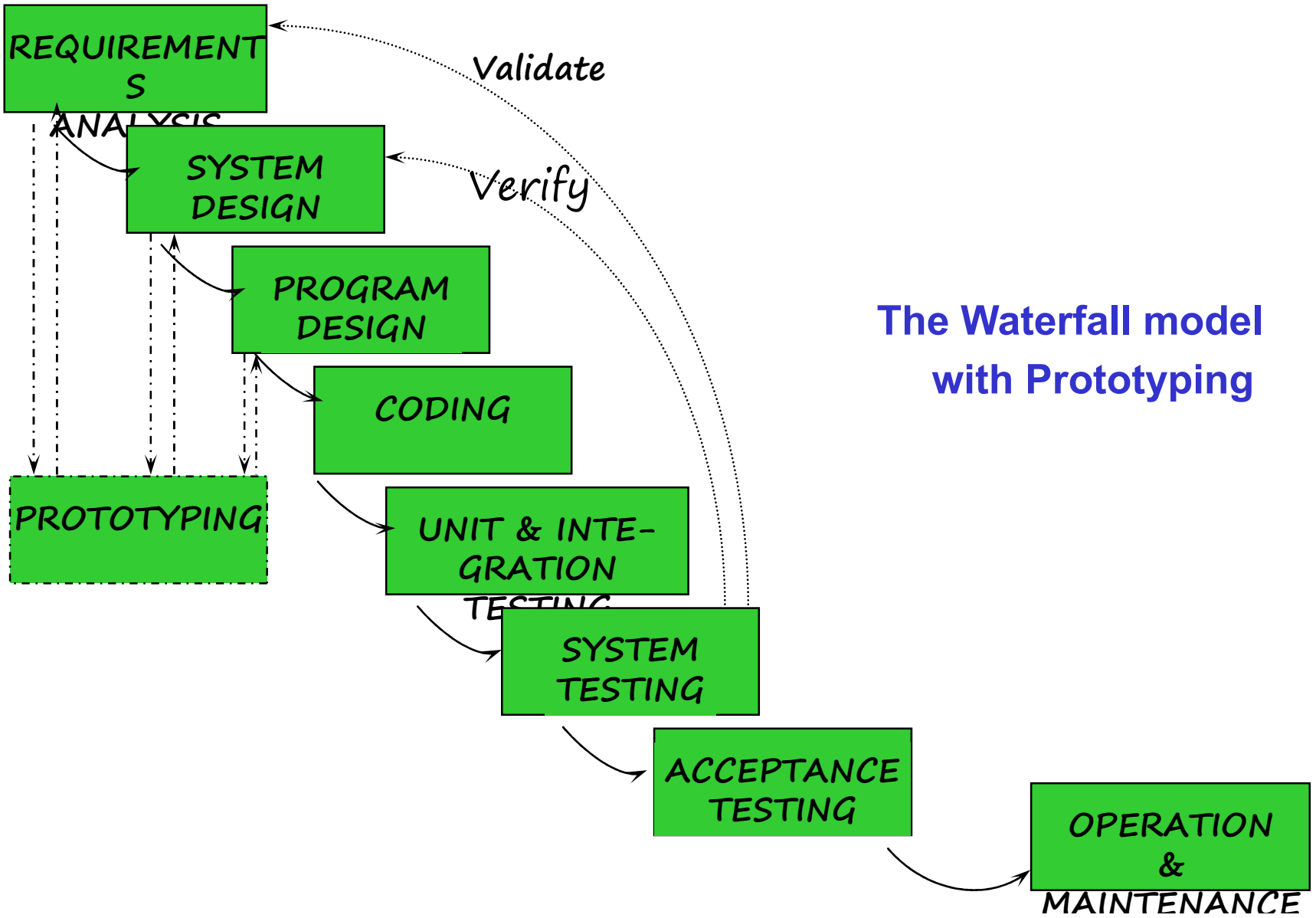
❑ **Requirements** • Time-to market	❑ **Waterfall model** • Not good - too long time to the final result
• User Interfaces	• Prototypes usually used
• Possibility for integration with other products	• Not good - the problems can be found too late
• Using standard and de-facto standard solutions	• Good, predicable results
• Using already existing components (COTS = commercial off the shelf)	• Not good, COTS may not fit to the design model
❑ **Permanent change of technologies.**	• Not good - unpredictable problems arise too late – but it can be planned
❑ **Requirements changes during the entire development process**	• Not good - a lot of overhead for updating the documentation

2.2 Software Process Models

Waterfall model with prototyping

- Prototyping is such a sub-process; a ***prototype*** is a partially developed product that enables customers and developer to examine some aspect of the proposed system and decide if it is suitable or appropriates for the finished product.

2.2 Software Process Models



The Waterfall model with Prototyping

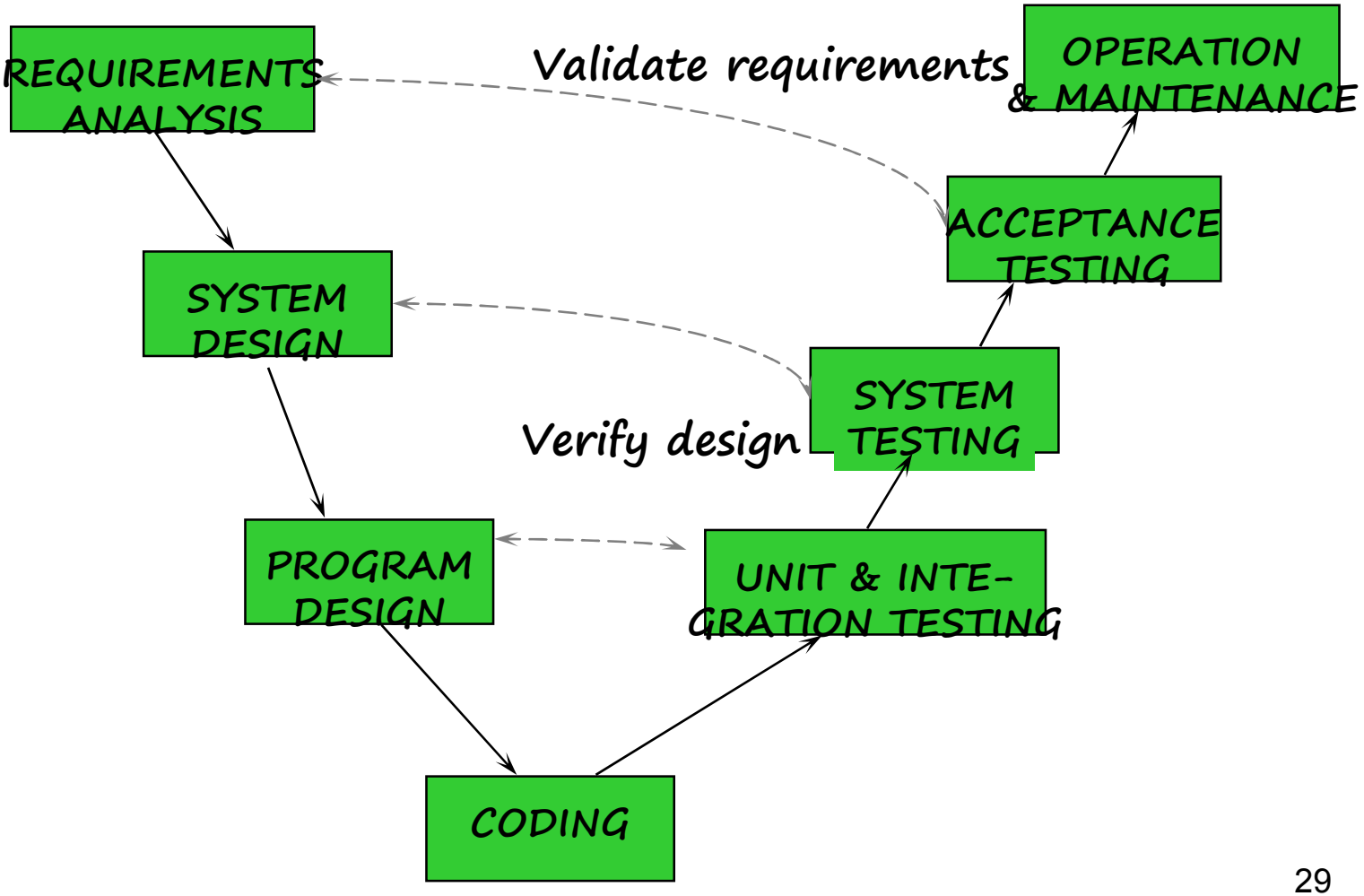
2.2 Software Process Models

V model

- Uses unit testing to verify procedural design
- Uses integration 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

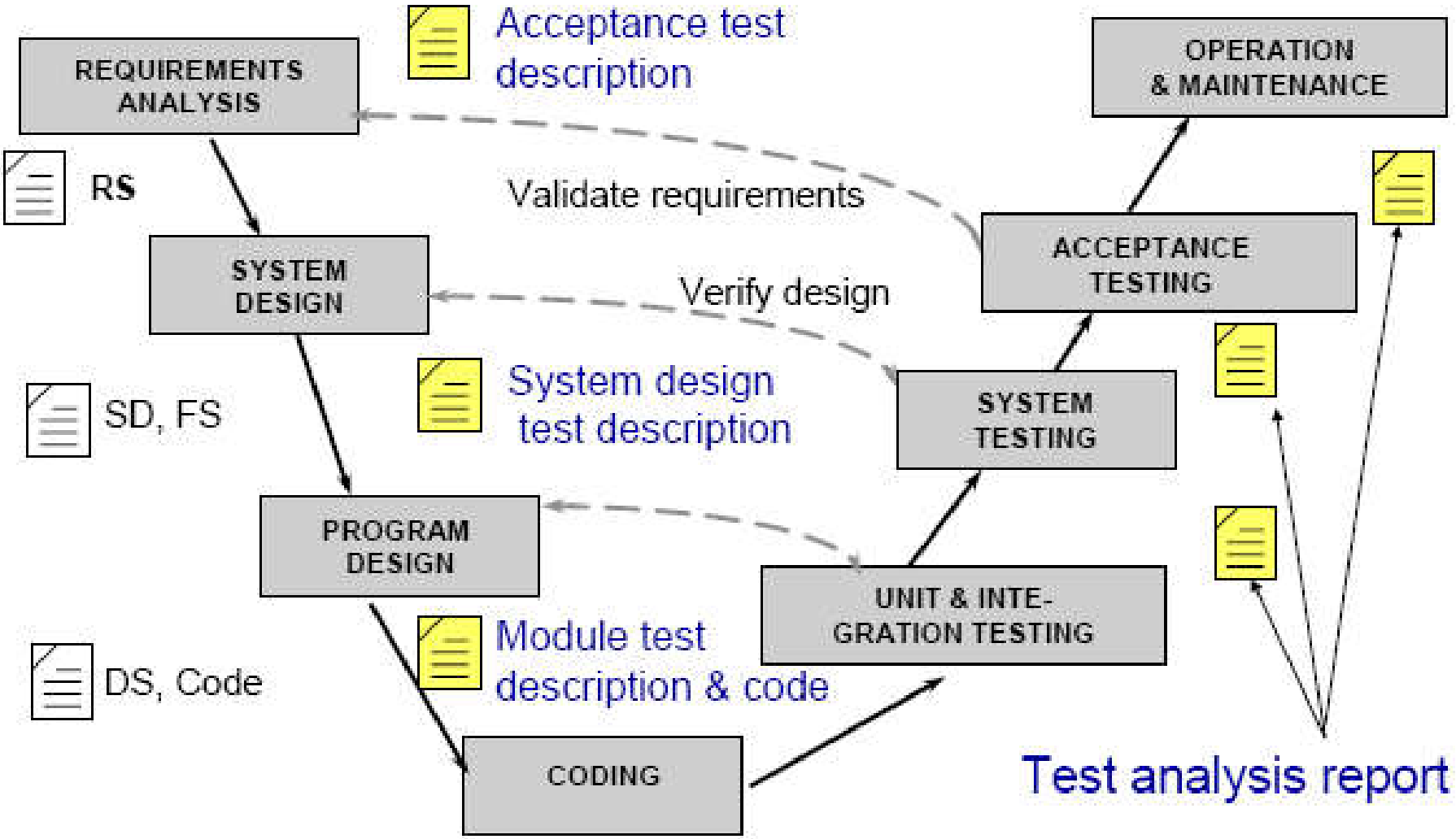
2.2 Software Process Models

V Model



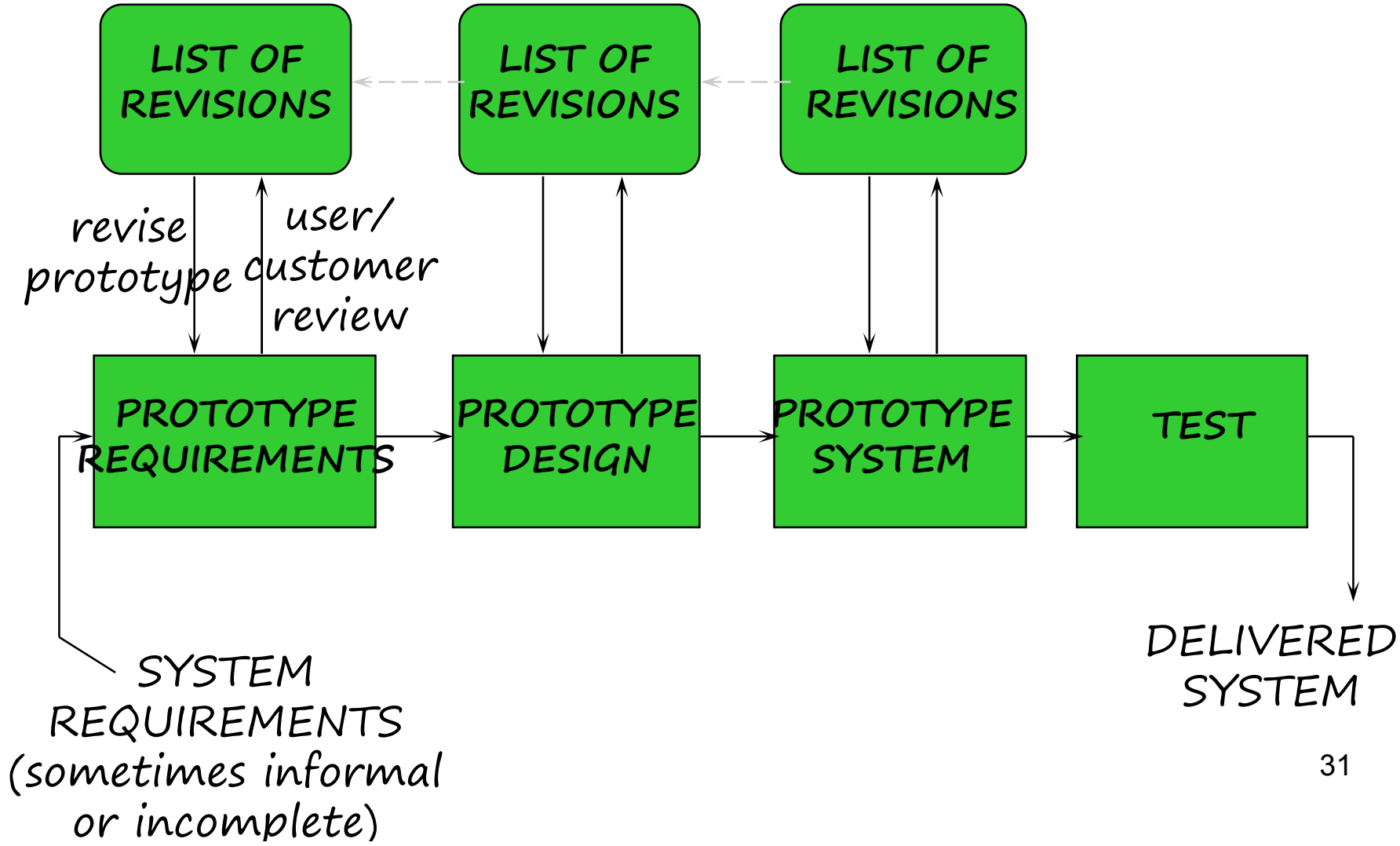
2.2 Software Process Models

V Model - document produced



2.2 Software Process Models

Prototyping model



2.2 Software Process Models

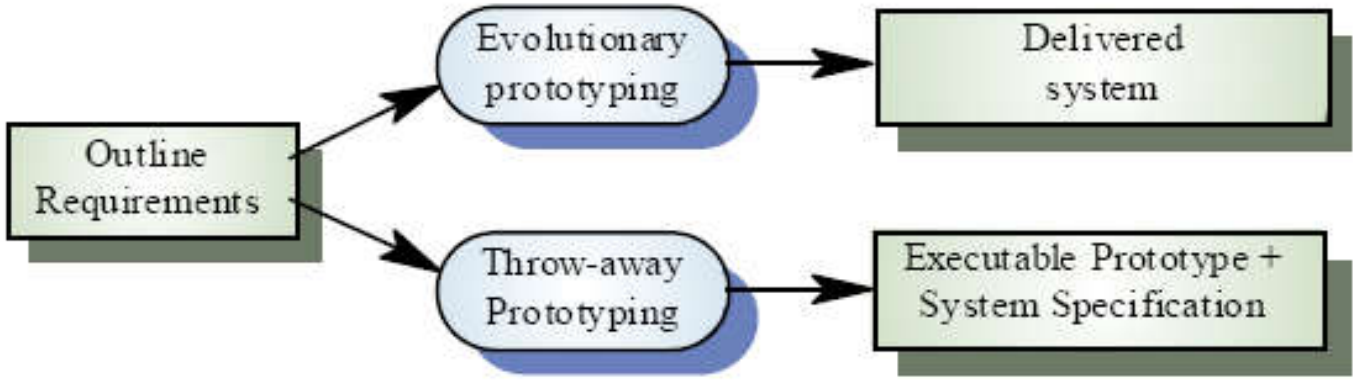
Prototyping model

- The Model allows all or part of a system to be constructed quickly to understand or clarify (澄清) issues (问题).
- requirements or design require repeated investigation to ensure that the developer, user and customer have a common understanding both of what is needed and what is proposed.
- reducing uncertainty in development.

2.2 Software Process Models

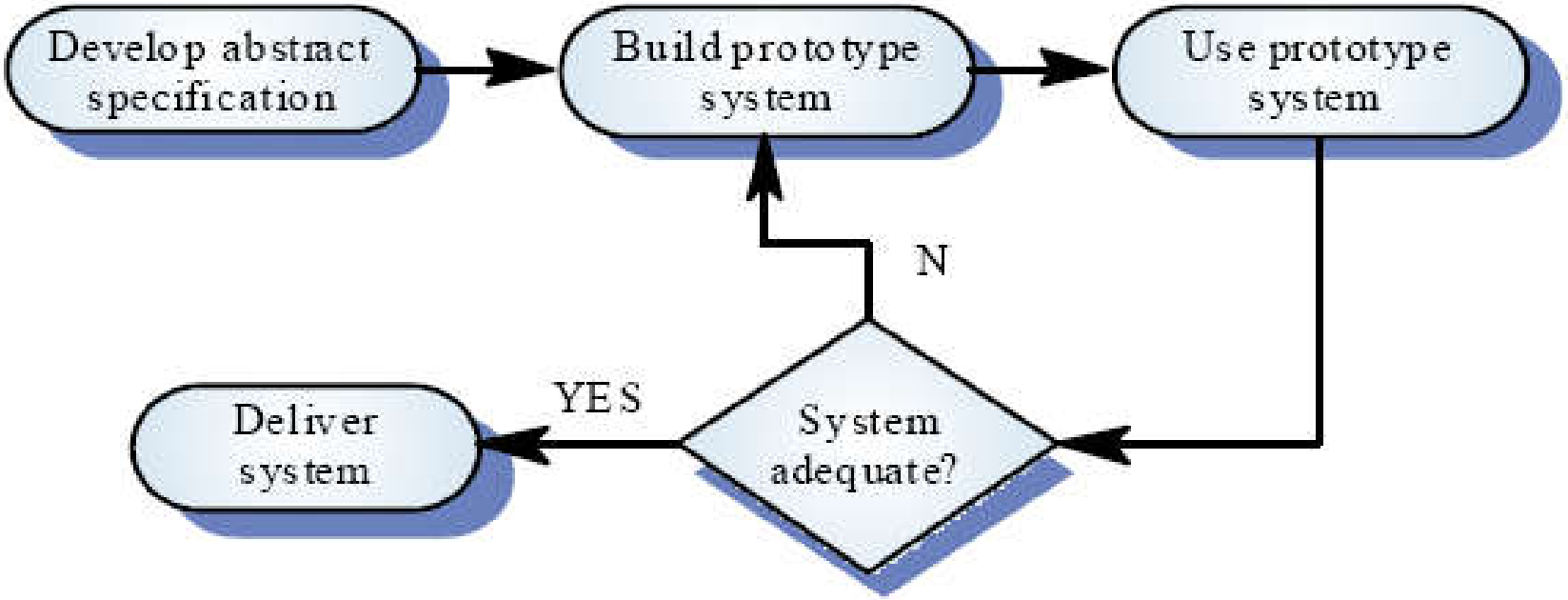
Prototyping

- Evolutionary prototyping
  - Objective is to work with customers and to evolve a final system from an initial outline specification. Should start with well-understood requirements
- Throw-away prototyping
  - Objective is to understand the system requirements. Should start with poorly understood requirements



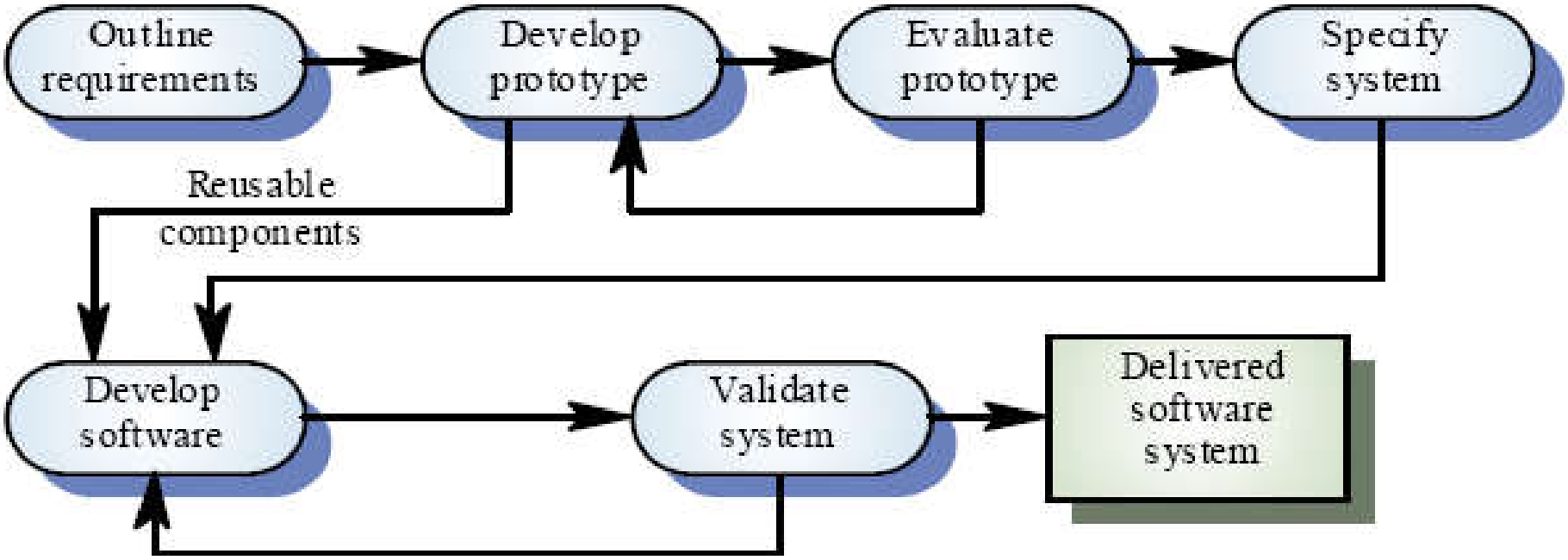
2.2 Software Process Models

Evolutionary prototyping



2.2 Software Process Models

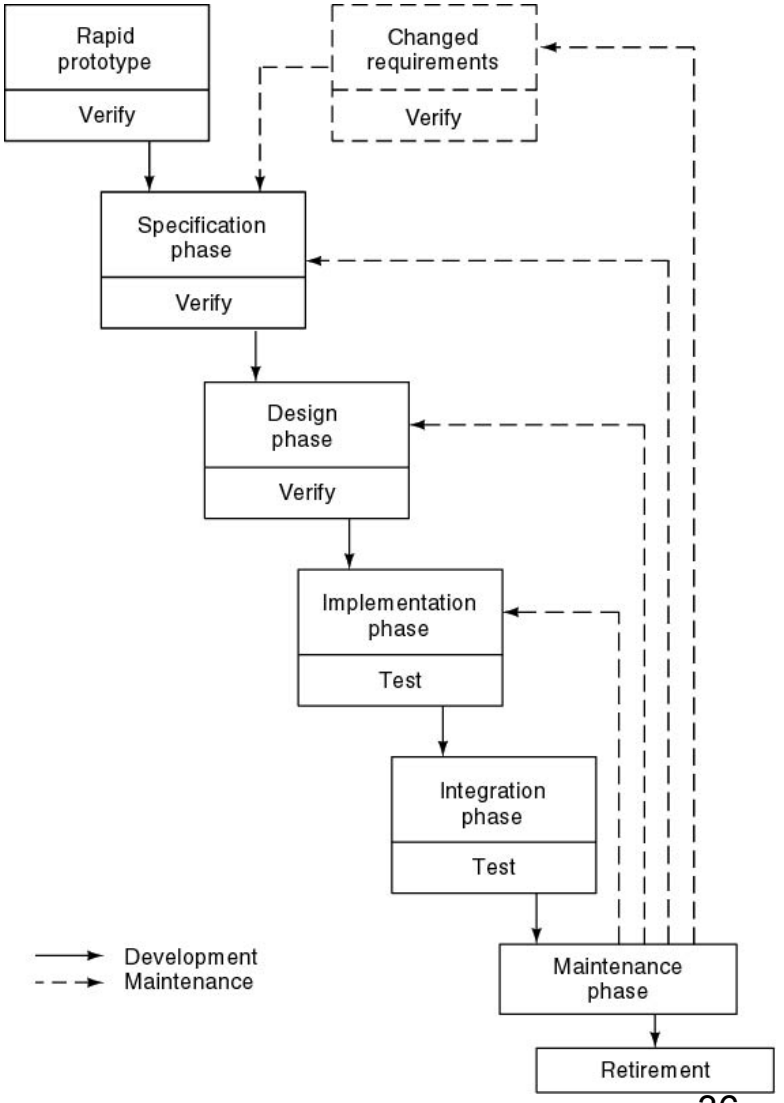
Throw-away prototyping



2.2 Software Process Models

Rapid Prototyping Model

- Rapid prototyping phase followed by waterfall
  - Do not turn the rapid prototype into the product
  - Rapid prototyping may replace the specification phase - never the design phase
- Comparison:
  - Waterfall model - try to get it right the first time
  - Rapid prototyping - frequent change, then discard



2.2 Software Process Models

Advantages and Disadvantages

- Advantages
  - Requirements better specified and validated
  - Early feasibility analysis
  - Strong involvement of the customer in the prototyping phase
- Disadvantage
  - Higher development effort
  - Danger that due to schedule slip, the prototype becomes part of the product

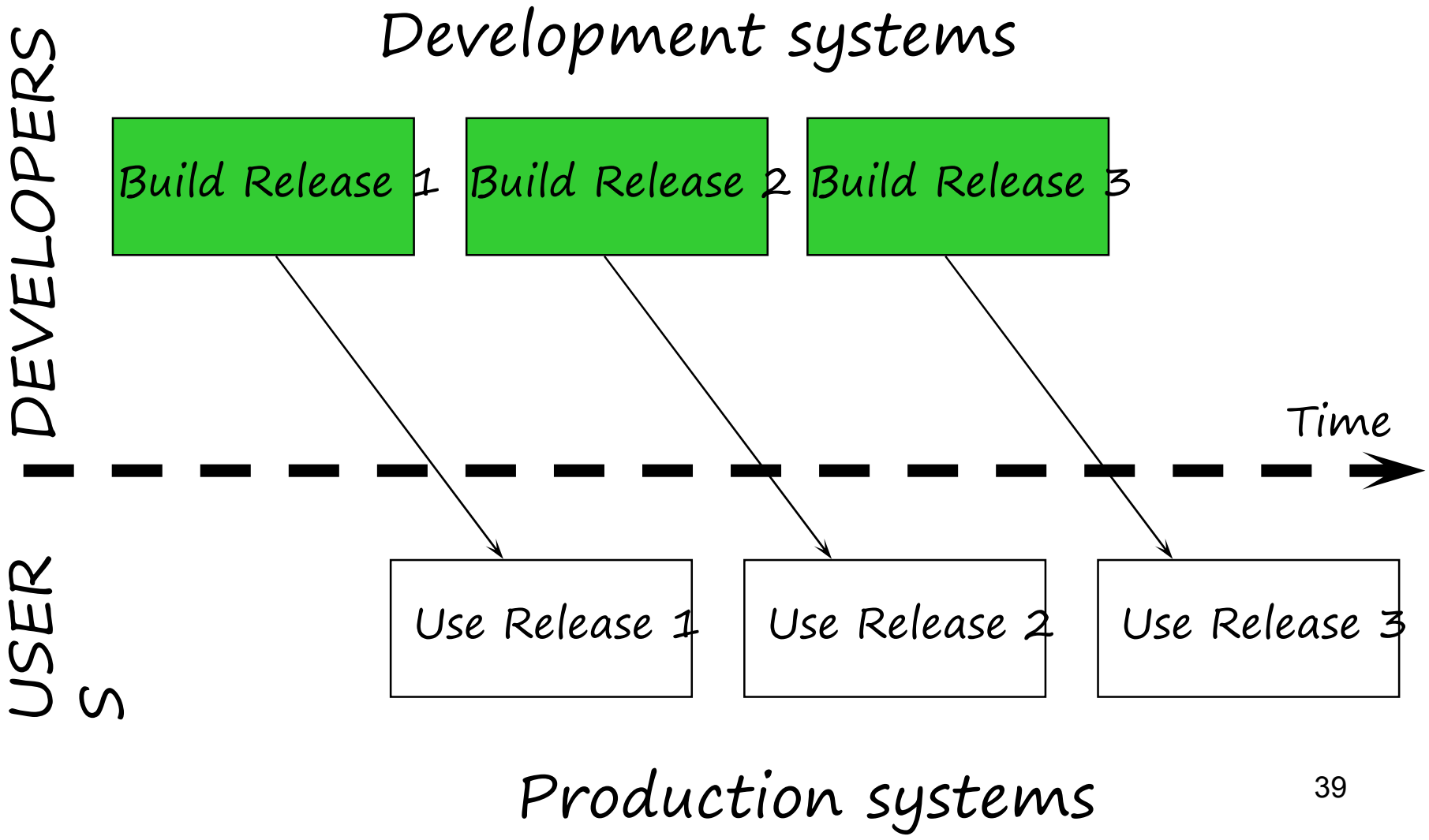
2.2 Software Process Models

Phased Development

- 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

The phased development model



2.2 Software Process Models

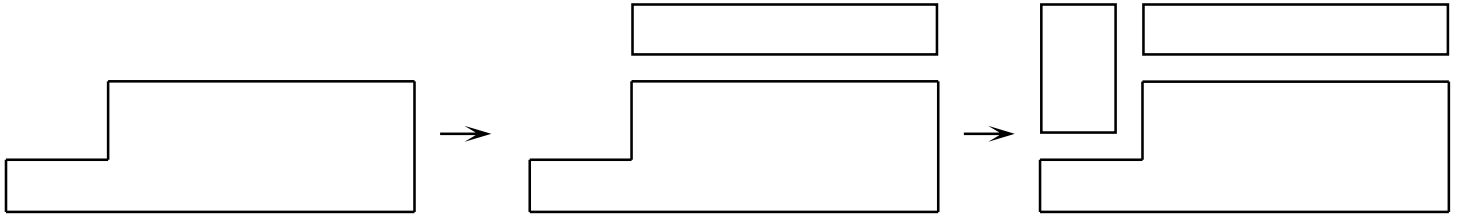
Phased Development - Increments and Iterations

- Incremental(增量) development: starts with small functional subsystem and adds functionality with each new release
  - Divide project into ***builds***
- Iterative(迭代) development: starts with full system, then changes functionality of each subsystem with each new release

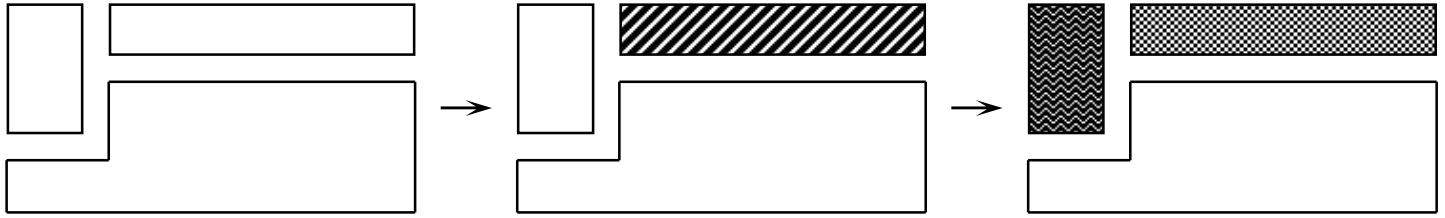
2.2 Software Process Models

The incremental and iterative development

INCREMENTAL DEVELOPMENT

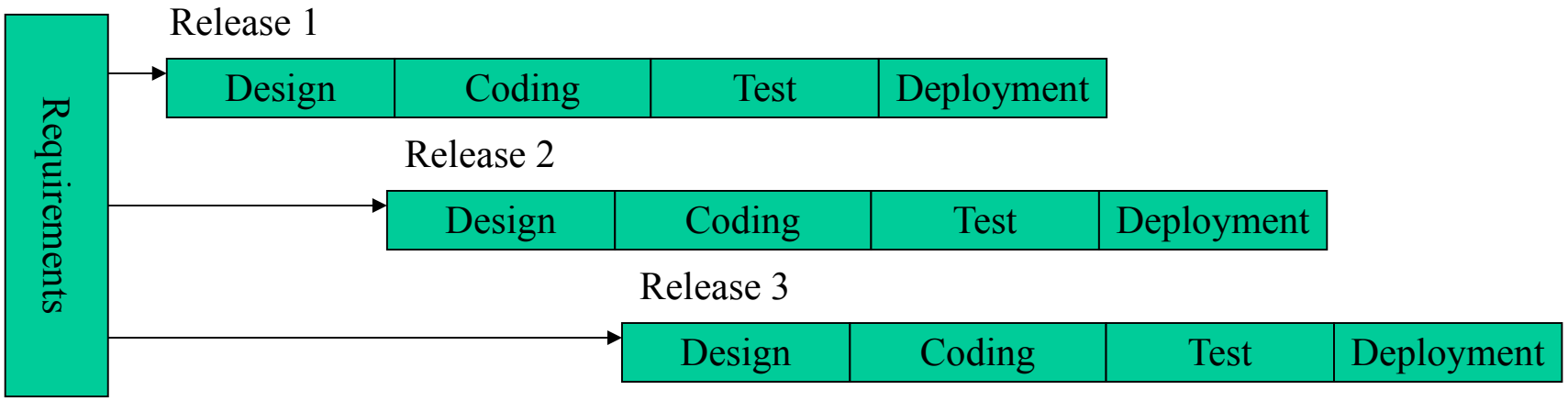


ITERATIVE DEVELOPMENT



2.2 Software Process Models

Incremental

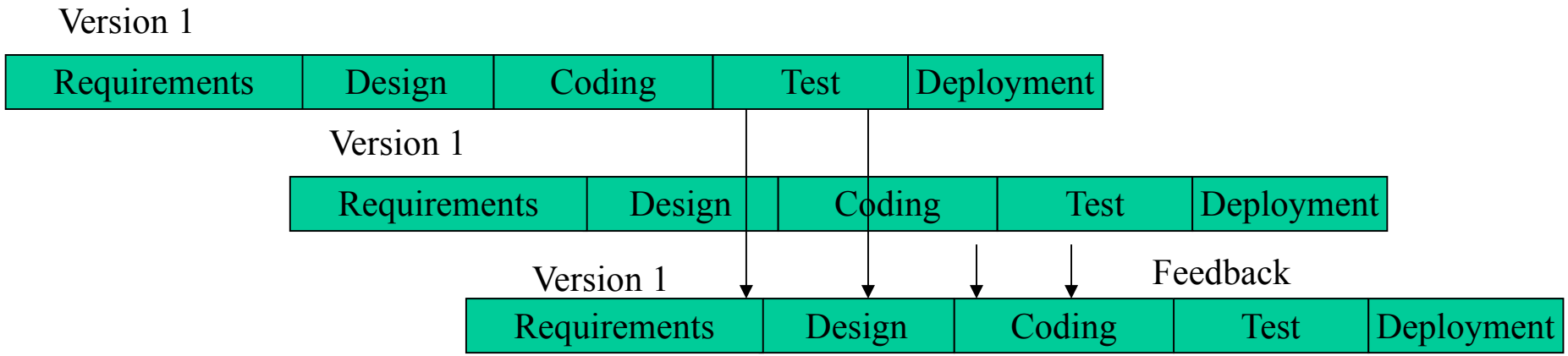


Each release adds more functionality,
i.e., a new increment

(Some call it iterative)

2.2 Software Process Models

Evolutionary



New versions implement *new* and evolving requirements

(Some call it iterative)

2.2 Software Process Models

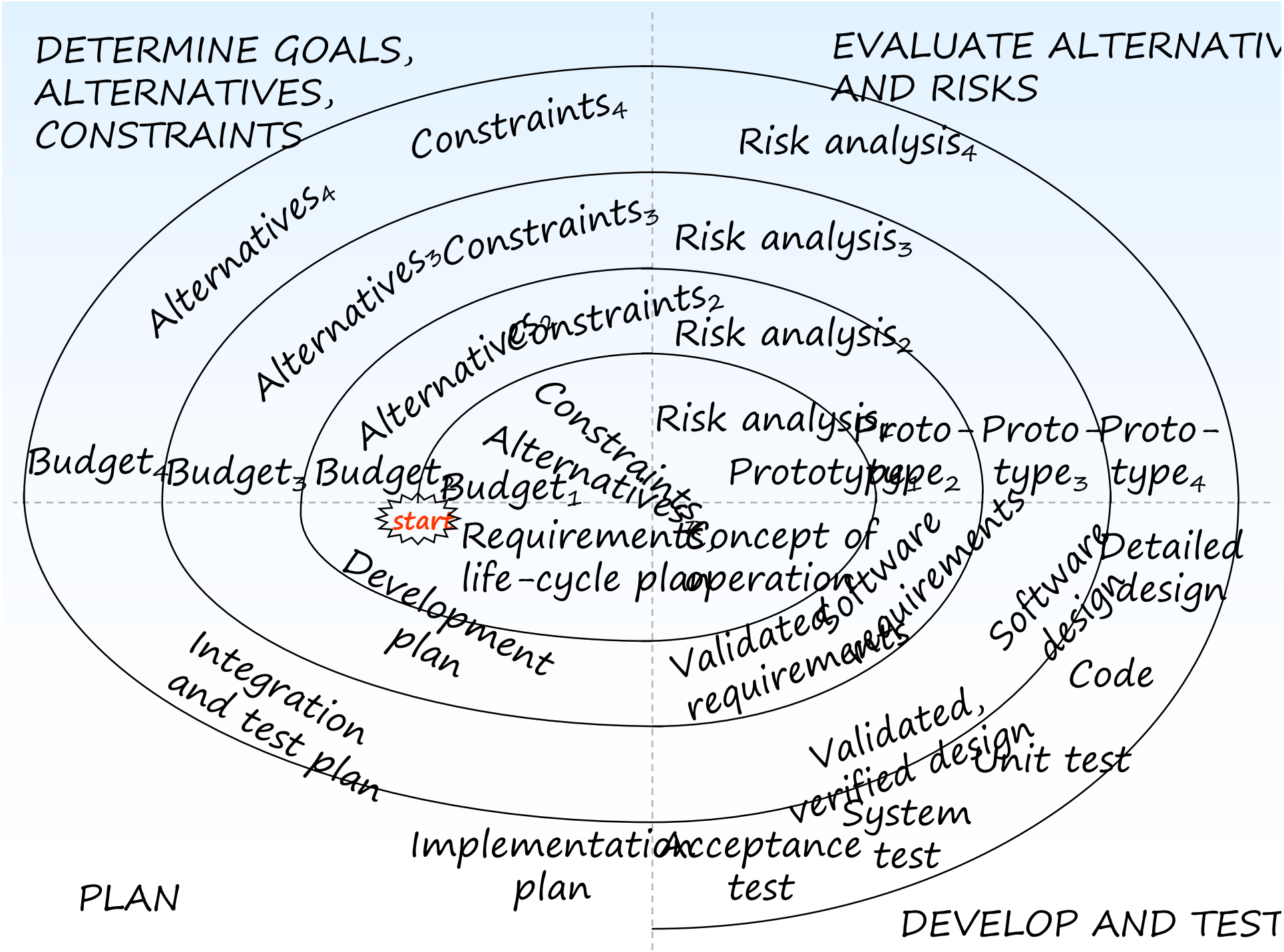
Phased Development - Increments and Iterations

- 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


DETERMINE GOALS, ALTERNATIVES, CONSTRAINTS
EVALUATE ALTERNATIV AND RISKS
Constraints4
Risk analysis4
Alternatives4
Alternatives3
Constraints3
Risk analysis3
Alternatives2
Constraints2
Risk analysis2
Constraints
Alternatives
Risk analysis
Budget4
Budget3
Budget2
Budget1
Proto-type1
Proto-type2
Proto-type3
Proto-type4
Prototype
start
Requirements plan
Concept of operation
Development plan
Validated requirements
Software requirements
Software design
Detailed design
Code
Integration and test plan
Validated, verified design
Unit test
System test
Acceptance test
Implementation plan
PLAN
DEVELOP AND TEST

2.2 Software Process Models

Process model - risk problems

- Waterfall
  - High risk for new systems because of specification and design problems
  - Low risk for well-understood developments using familiar technology
- Prototyping
  - Low risk for new applications because specification and program stay in step
  - High risk because of lack of process visibility
- Evolutionary and Spiral
  - Middle ground between waterfall and prototyping

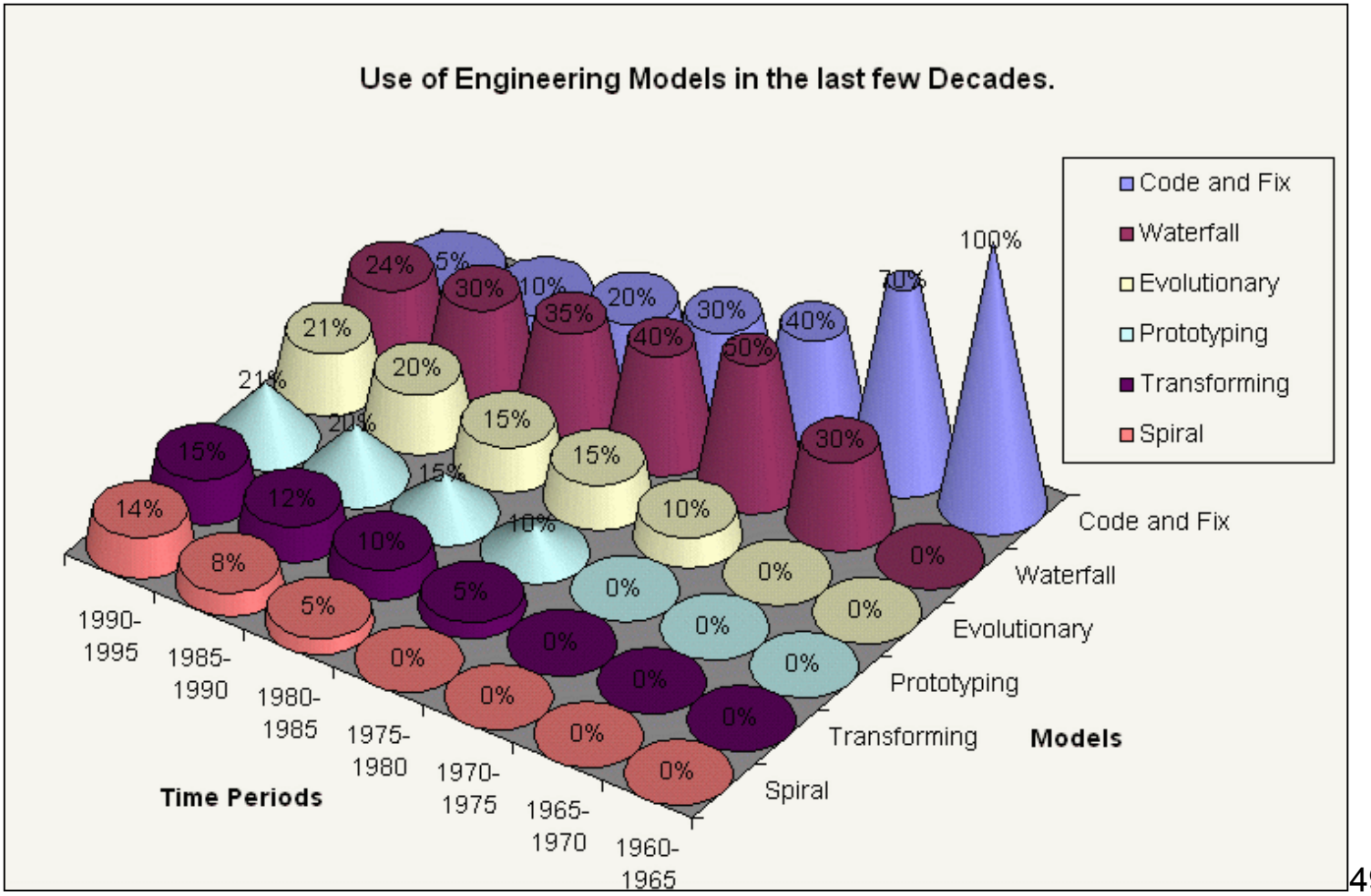
2.2 Software Process Models

Hybrid process models

- Large systems are usually made up of several sub-systems
- The same process model need not be used for all subsystems
- Prototyping for high-risk specifications
- Waterfall model for well-understood developments
- Taylor the process to a problem

2.2 Software Process Models

Use of the Models in Practice



2.2 Software Process Models

- Many of the software-development processes proposed and used from the 1970s through the 1990s tried to impose some form of rigor on the way in which software is conceived, documented, developed, and tested.
- In the late 1990s, some developer who had resisted this rigor formulated their own principles, trying to highlight the roles that flexibility could play in producing software quickly and capably.

2.2 Software Process Models

Agile Methods

- Agile Manifesto
  - They value individuals and interactions over processes and tools.
  - They prefer to invest time in producing working software rather than in producing comprehensive documentation.
  - They focus on customer collaboration rather than contrast negotiation, thereby involving the customer in key aspects of the development process.
  - They concentrate on responding to change rather than on creating a plan and then following it, because they believe that it is impossible to anticipate all requirements at the beginning of development.

2.2 Software Process Models

Agile Methods

- The overall goal of agile development
  - to satisfy the customer by "early and continuous delivery of valuable software".
- It is thought that by building flexibility into the development process, agile methods can enable customers to add or change requirements late in the development cycle.
  - Many customers have business needs that change over time, reflecting not only newly discovered needs but also the need to respond to changes in the marketplace.

2.2 Software Process Models

Agile Methods

- There are many examples of agile processes. Each is based on a set of principles that implement the tenets pf the agile manifesto.
  - Extreme programming (XP) is a set of techniques for leveraging the creativity of developers and minimizing the amount of administrative overhead.
  - Crystal is a collection of approaches based on the notion that every project needs a different set of policies, conventions, and methodologies. This method believes that people have a major influence on software quality.

2.2 Software Process Models

Agile Methods

- Many examples of agile processes
  - Scrum:
    - It uses iterative development, where each 30-day iterative is called a “Sprint”, to implement the product’s backlog of prioritized requirements.
    - Multiple self-organizing and autonomous teams implement product increments in parallel.
    - Coordination is done at a brief daily status meeting called a “Scrum” (as in rugby).

2.2 Software Process Models

Agile Methods

- Many examples of agile processes
  - Adaptive software development (ASD)
    - There is a mission that acts as a guideline, setting out the destination but not prescribing how to get there.
    - Features are viewed as the crux of customer value, so the project is organized around building components to provide the features.
    - Iteration is important, so redoing is as critical is doing; change is embraced, so that a change is viewed not as a correction but as an adjustment to the realities of software development.
    - Fixed delivery times force developers to scope down the requirements essential for each version produced.
    - At the same time, risk is embraced, so that the developers track the hardest problems first.

2.2 Software Process Models

Agile Methods

- XP is used to describe the more general concept of agile methods. XP is a particular form of agile process, with guiding principles that reflect the more general tenets of the agile manifesto.
- Proponents of XP emphasize four characteristics:
  - Communication
  - Simplicity
  - Courage
  - Feedback

2.2 Software Process Models

Agile Methods

- The twelve facets of XP
  - The planning game
  - Small releases
  - Metaphor
  - Simple design
  - Writing tests first
  - Refactoring
  - Pair programming
  - Collective ownership
  - Continuous integration
  - ❿ Sustainable pace
  - ❿ On-site customer
  - ❿ Coding standards