Sommerville Chapter 2 and 3

The Process

Slides courtesy Prof. Mats Heimdahl

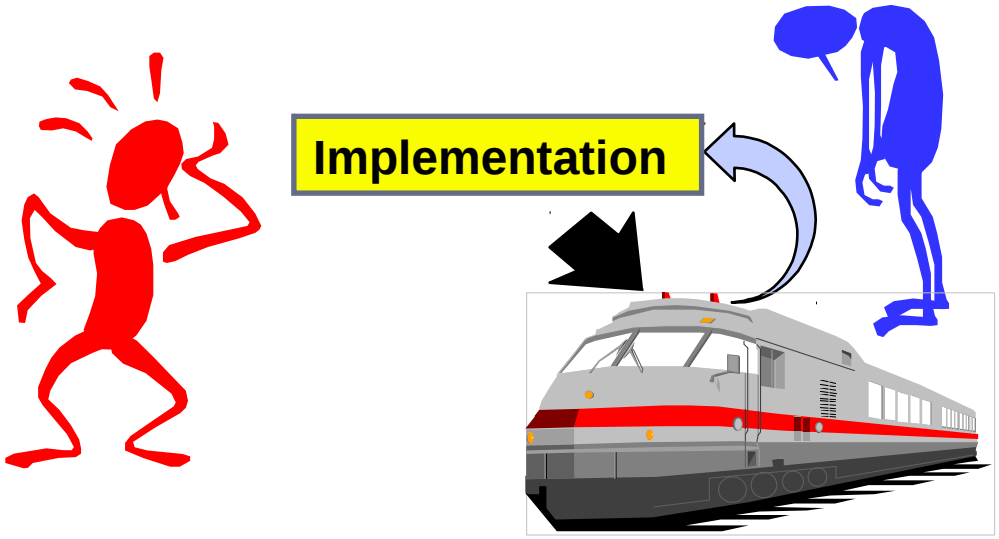
Today's Goals

- Introduce and/or Review Software Development Processes
  - Definitions, Processes, and Process Models
  - Examples of Software Process Models

Why is Software Development so %$##% Hard? (L)

- Complexity
  - Software systems are the most complex artifacts ever created
- Changeability
  - Software is “easy” to change
- Invisibility
  - We cannot see the progress of the development
- Conformity
  - The software will have to be molded to fit whatever external constraints may imposed

Code and Fix



We Need a Software Process

- Structured set of activities required to develop a software system
  - Specification
  - Design
  - Validation
  - Evolution
- Activities vary depending on the organization and the type of system being developed
- Must be explicitly modeled if it is to be managed

Code and Fix Model (L)

- Applicability
  - Used for small, simple projects
- Potential Problems
  - Quality
  - Maintainability

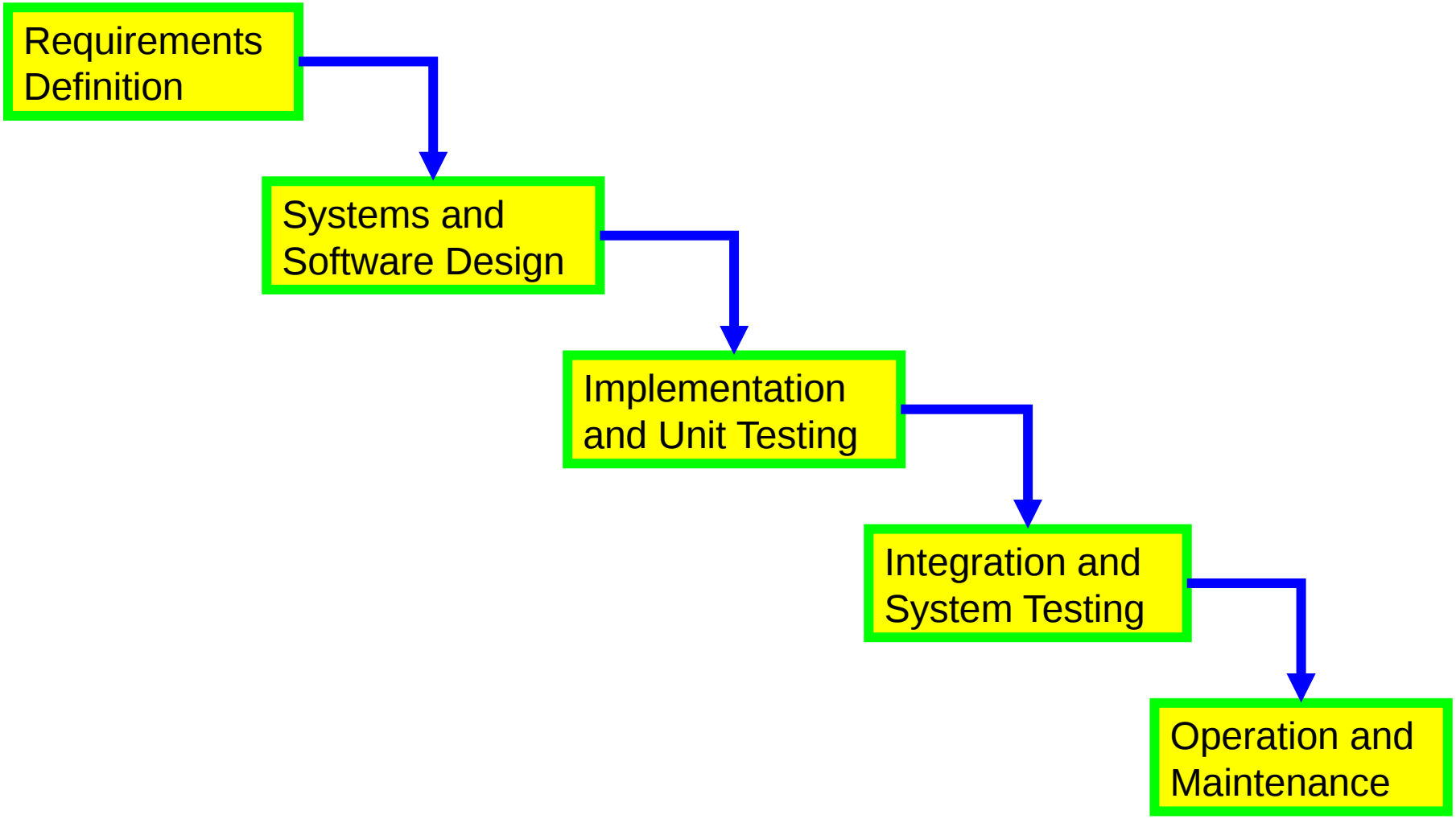
Generic Software Process Models

- The Waterfall Model
  - Separate and distinct phases of specification and development
- Evolutionary Development
  - Specification and development are interleaved
- Spiral Model
  - Let risk analysis drive your process
- Incremental Development
  - Deliver your system in small planned increments
- Agile and eXtreme Programming

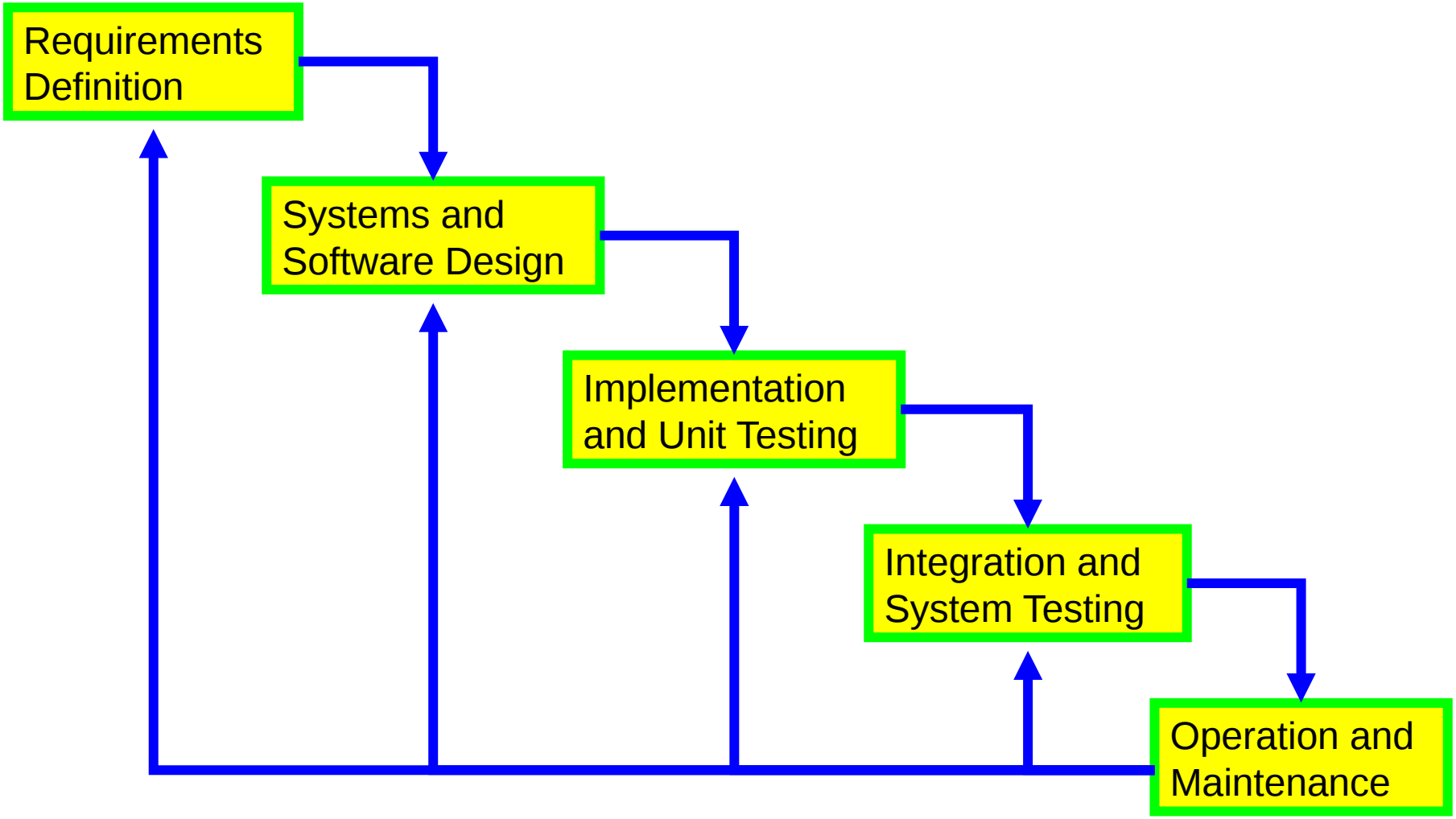
Process Characteristics

- Understandability
  - Is the process defined and understandability
- Visibility
  - Is the process progress externally visible
- Supportability
  - Can the process be supported by CASE tools
- Acceptability
  - Is the process acceptable to those involved in it
- Reliability
  - Are process errors discovered before they result in product errors
- Robustness
  - Can the process continue in spite of unexpected problems
- Maintainability
  - Can the process evolve to meet changing organizational needs
- Rapidity
  - How fast can the system be produced

Waterfall Model



Waterfall Model



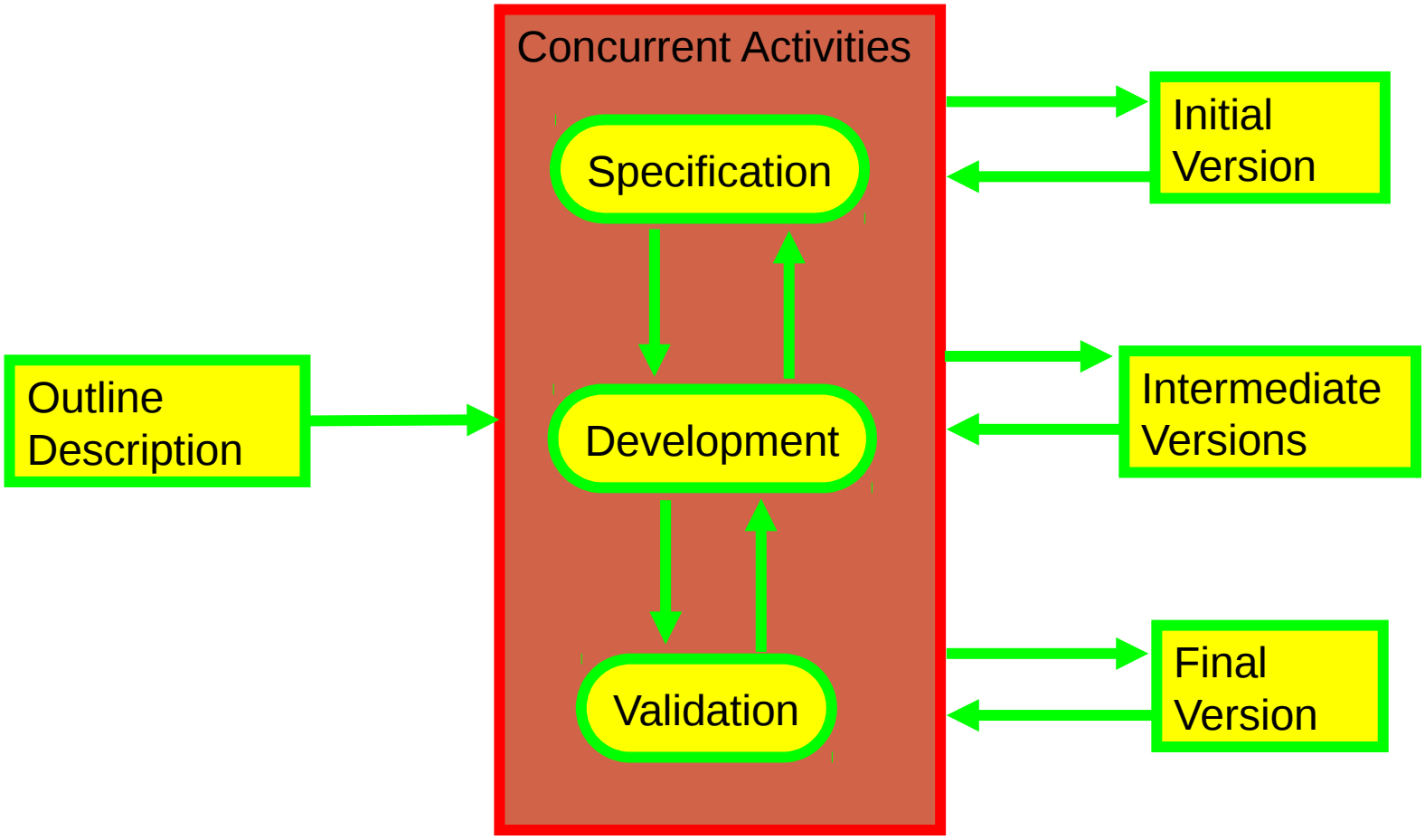
Waterfall Model Documents

Activity	Output document
Requirements analysis	Feasibility study, outline the requirements
Requirements definition	Requirements document
System specification	Functional specification, Acceptance test plan, Draft User's 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's manual
System testing	System test report
Acceptance testing	Final system and documentation

Waterfall Model (L)

- Problems
  - Inflexible model not accommodating change
  - You seldom know the requirements that early
  - Does not accommodate evaluation of project risk
- Applicability
  - Projects where the requirements are very well known
  - Low risk projects

Evolutionary Development



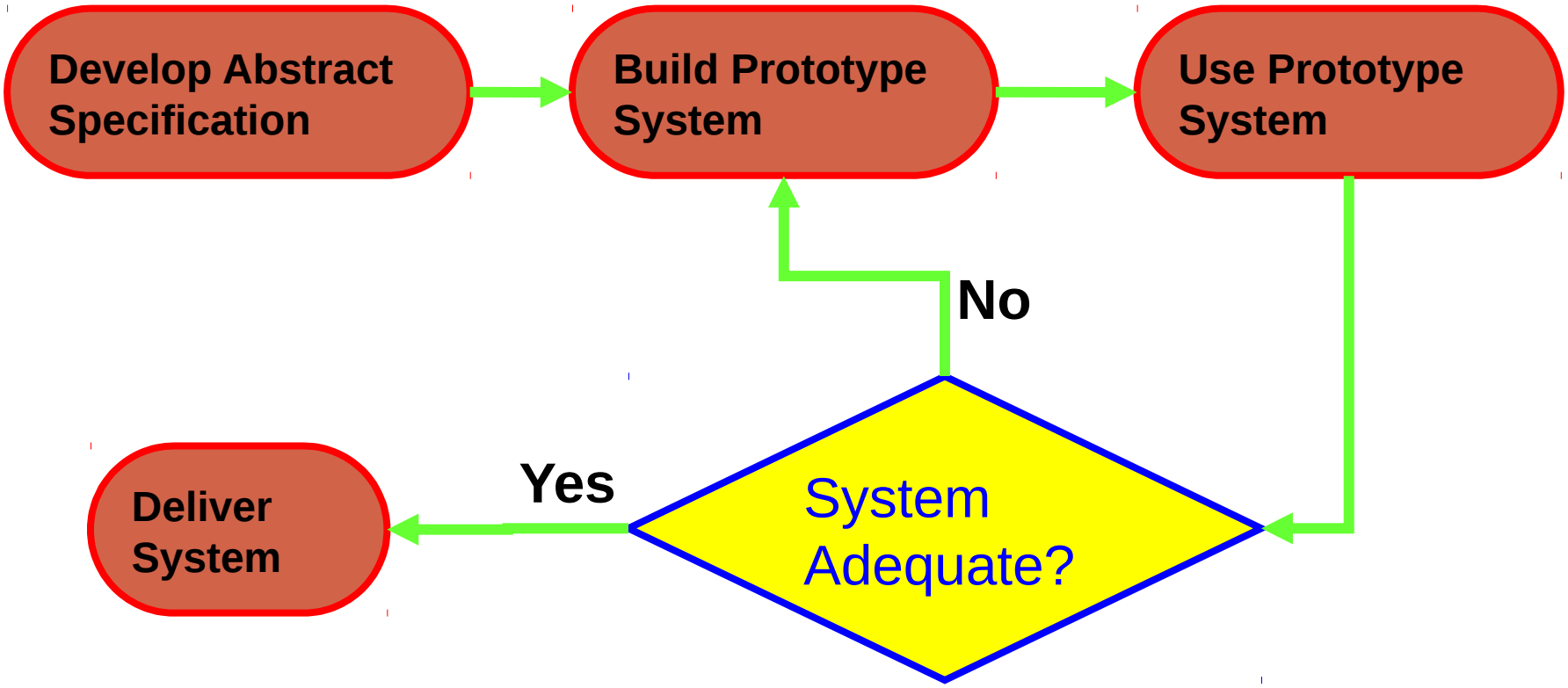
Prototyping Benefits

- Misunderstandings between software users and developers are exposed
- Missing services may be detected
- Confusing services may be identified
- A working system is available early in the process
- The prototype may serve as a basis for deriving a system specification

Evolutionary Development

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

Evolutionary Prototyping



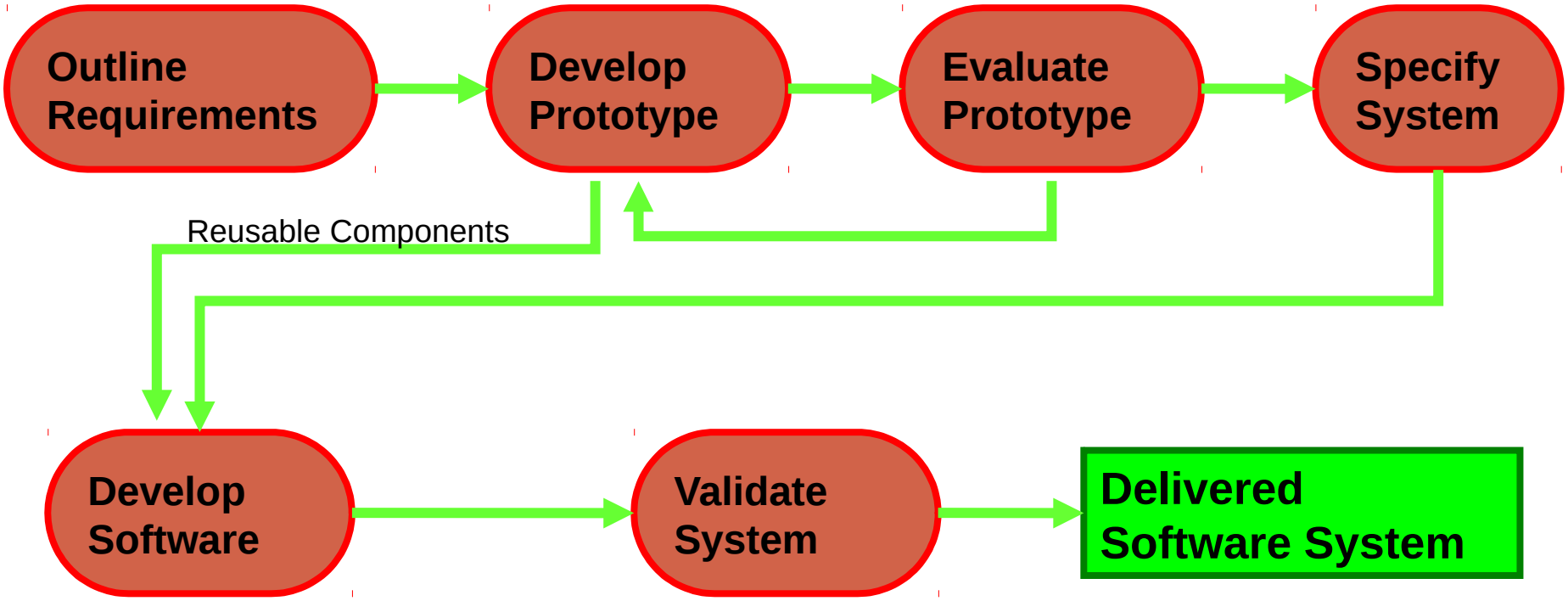
Evolutionary Prototyping

- Must be used for systems where the specification cannot be developed in advance
  - AI systems and user interface systems
- Based on techniques which allow rapid system iterations
- Verification is impossible as there is no specification
  - Validation means demonstrating the adequacy of the system

Evolutionary Prototyping Problems

- Existing management processes assume an “organized” model of development
- Continual change tends to corrupt system structure so long-term maintenance is expensive
- Specialist skills are required which may not be available in all development teams
- Organizations must accept that the lifetime of systems developed this way will inevitably be short

Throw-away Prototyping



Throw-away Prototyping

- Used to reduce requirements risk
- The prototype is developed from an initial specification, delivered for experiment then discarded
- The throw-away prototype should NOT be considered as a final system
  - Some system characteristics may have been left out
  - There is no specification for long-term maintenance
  - The system will be poorly structured and difficult to maintain

Evolutionary Development (L)

- Problems
  - Lack of process visibility
  - Systems are often poorly structured
  - Special skills (e.g., in languages for rapid prototyping) may be required
- Applicability
  - For small or medium-size interactive systems
  - For parts of large systems (e.g. the user interface)
  - For short-lifetime systems

We Have “Learned” (at least seen)

- The waterfall model
  - Separate and distinct phases of specification and development
- Evolutionary development
  - Specification and development are interleaved
  - Evolutionary and throw away prototyping

Next Time

- More process
  - Spiral model
  - Incremental development
  - eXtreme programming (Agile)