SOFTWARE ENGINEERING

CO3001

CHAPTER 2 – SOFTWARE PROCESSES

Anh Nguyen-Duc
Tho Quan Thanh

Adapted from https://iansommerville.com/software-engineering-book/slides/

TOPICS COVERED

- ✓ Software process models
- ✓ Process activities
- ✓ Coping with change
- ✓ Process improvement

- ✓ Menti + Breakout room for discussion

- ✓ What is your understanding of a “Software Process”?
- ✓ Have you used any “Software Process Model” in your practice?
  - Which models?
  - Examples?
  - Uses? Strengths/Weaknesses?
  - Observations?

SOFTWARE ENGINEERING – *FOR ORIENTATION*

- ✓ Software Engineering is a branch of systems engineering concerned with the development of large and complex software intensive systems. It focuses on:
  - the real-world goals for, services provided by, and constraints on such systems,
  - the precise specification of systems structure and behaviour, and the implementations of these specifications,
  - the activities required in order to develop an assurance that the specifications and real world-world goals have been met,
  - the evolution of these systems over time, and across systems families,
  - It is also concerned with the processes, methods and tools for the development of software intensive systems in an economic and timely manner.

Reference: A. Finkelstein

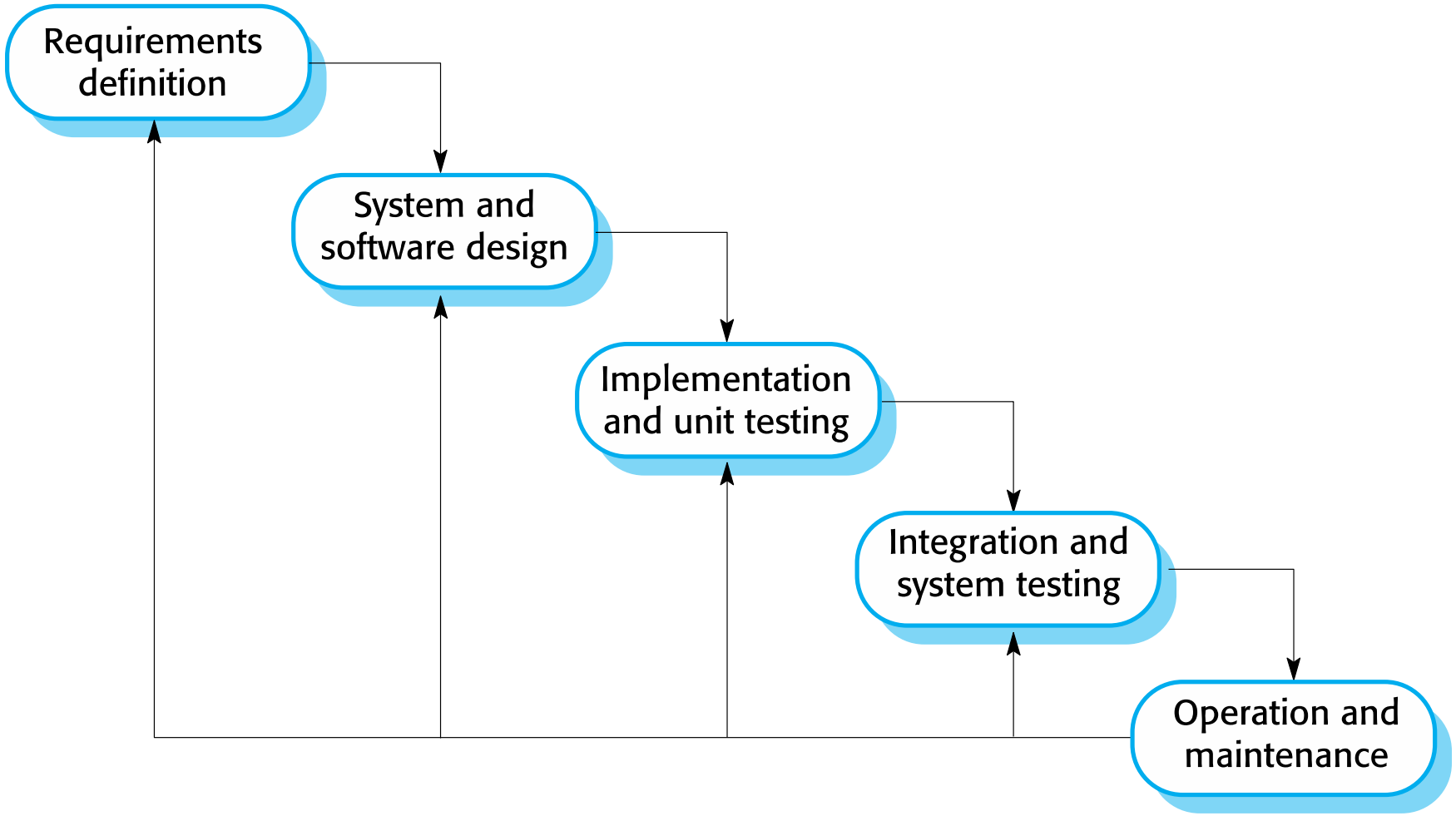
THE SOFTWARE PROCESS

- ✓ A structured set of activities required to develop a software system.
- ✓ Many different software processes but all involve:
  - Specification
  - Design and implementation
  - Validation
  - Evolution.
- ✓ A software process model
  - an abstract representation of a process

SOME SOFTWARE PROCESS MODELS

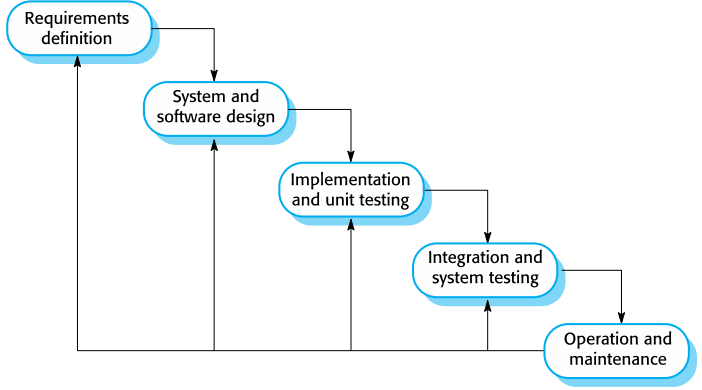
- ✓ The waterfall model
  - Plan-driven model.
  - Separate and distinct phases of specification and development.
- ✓ Incremental development
  - Specification, development and validation are interleaved.
  - May be plan-driven or agile.
- ✓ Integration and configuration
  - The system is assembled from existing configurable components.
  - May be plan-driven or agile.
- ✓ In practice, most large systems are developed using a process that incorporates elements from all of these models.

THE WATERFALL MODEL



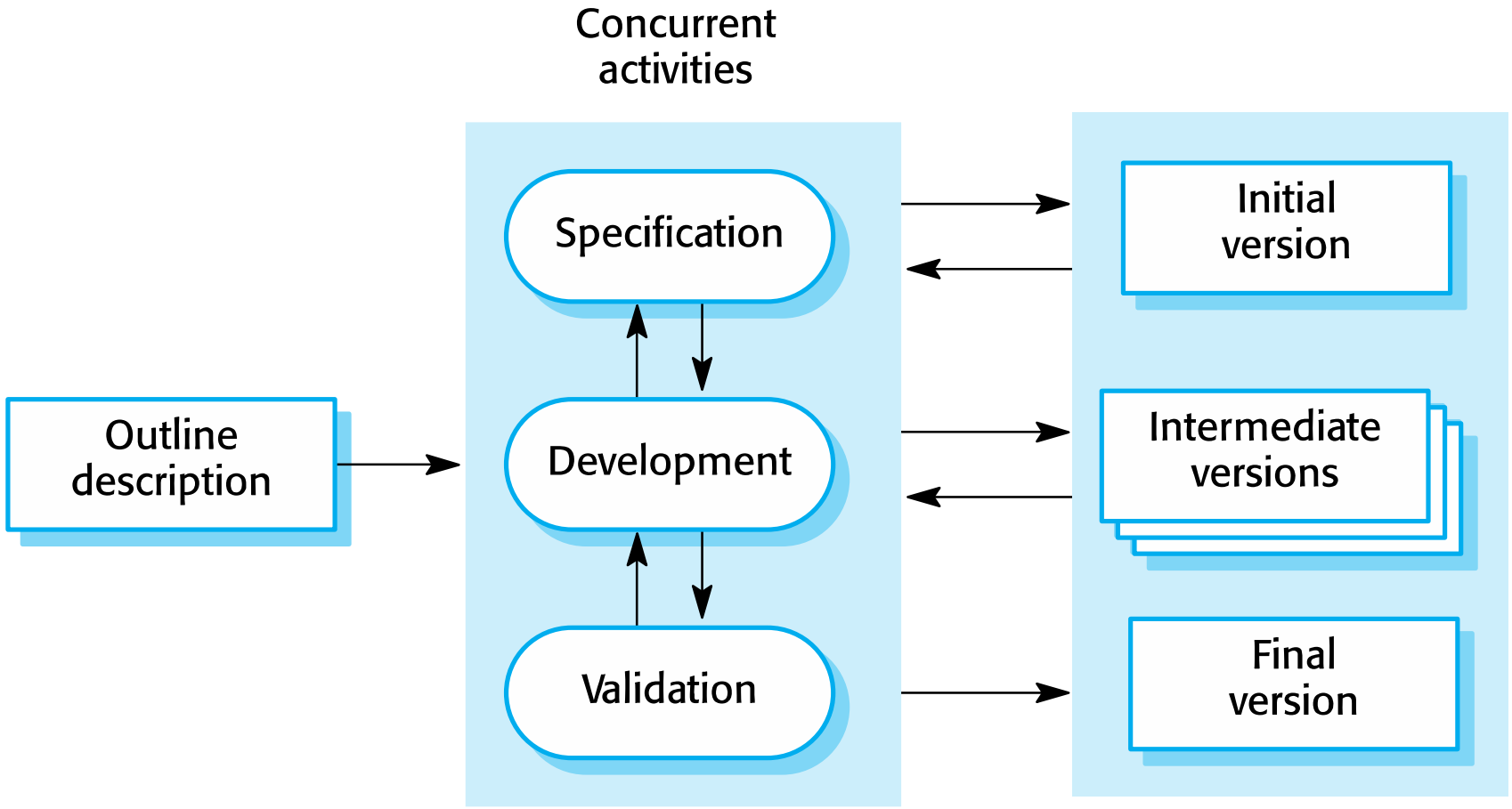
In principle, a phase has to be complete before moving onto the next phase.

WATERFALL MODEL USAGES



- ✓ The main drawback:
  - the difficulty of accommodating change after the process is underway.
- ✓ Mostly used for large systems engineering projects
  - a system is developed at several sites.
  - the plan-driven nature of the waterfall model helps coordinate the work.
- ✓ When the requirements are well-understood and changes will be fairly limited during the design process.
  - Few business systems have stable requirements.

INCREMENTAL DEVELOPMENT



INCREMENTAL DEVELOPMENT BENEFITS

- ✓ Reduce the cost of accommodating changing customer requirements
- ✓ Easier to get customer feedback on the development work that has been done.
- ✓ More rapid delivery and deployment of useful software to the customer

INCREMENTAL DEVELOPMENT PROBLEMS

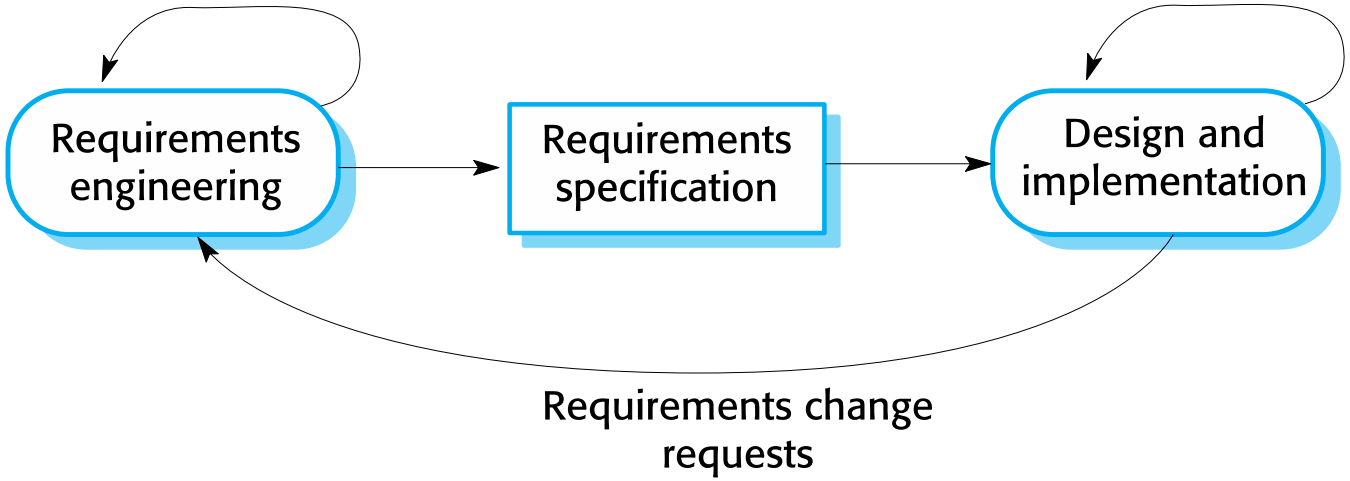
- ✓ The process is not visible.
  - Managers need regular deliverables
  - Not cost-effective to produce documents for every product version

- ✓ System structure tends to degrade as new increments are added.
  - Need time and money on refactoring to improve the software
  - Regular change tends to corrupt the structure.
  - Incorporating further software changes becomes increasingly difficult and costly.

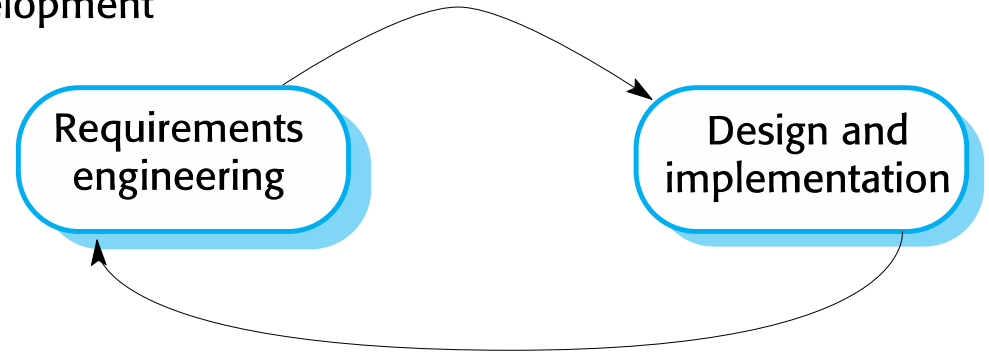
AGILE DEVELOPMENT

Plan-based development

i.e.: waterfall model, incremental development



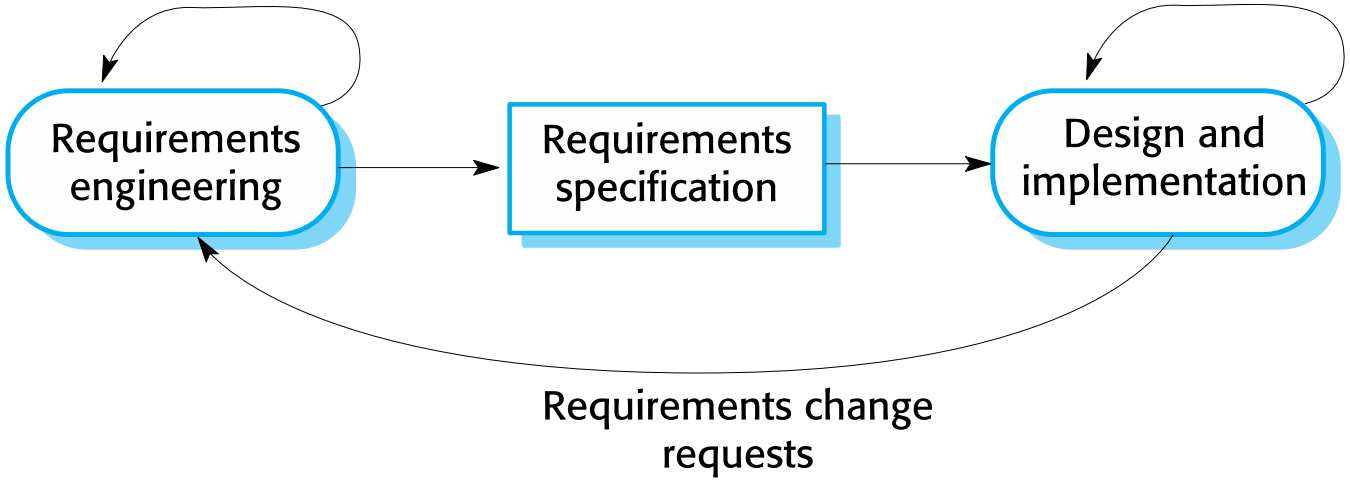
Agile development



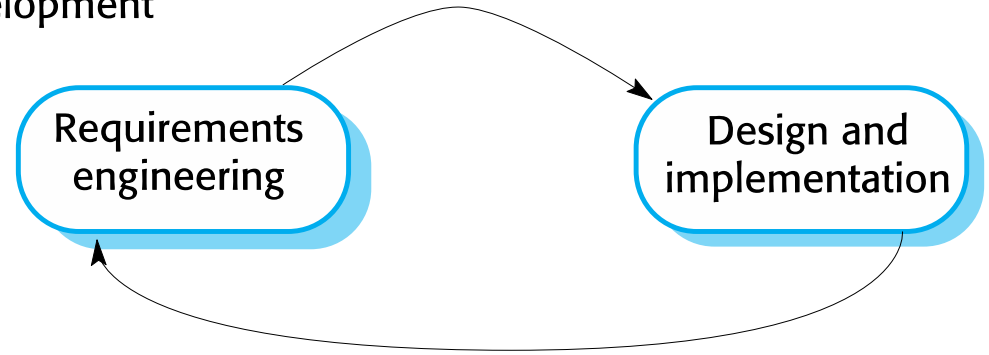
AGILE DEVELOPMENT

Plan-based development

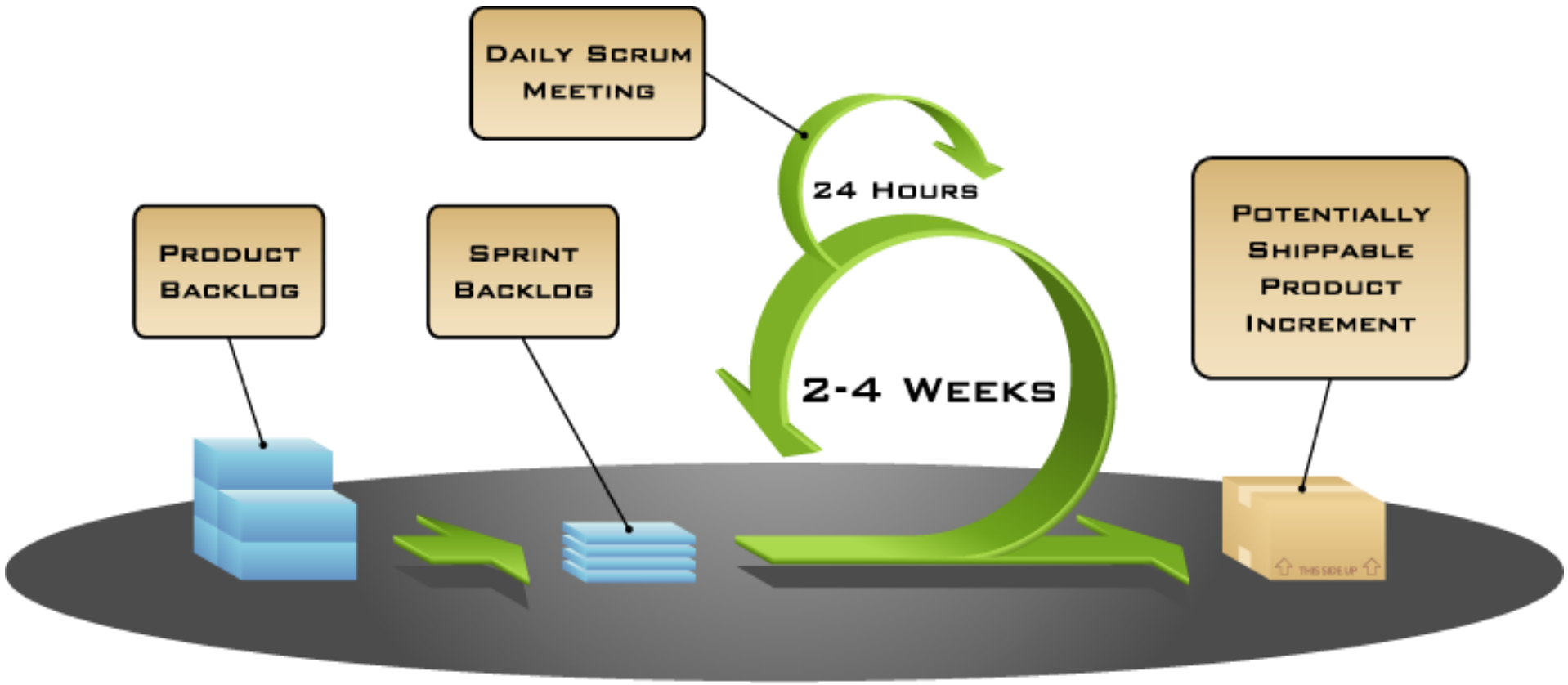
i.e.: waterfall model, incremental development



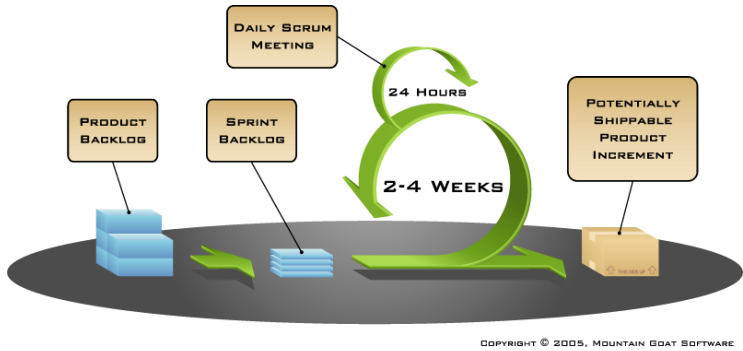
Agile development



SCRUM – THE MOST POPULAR AGILE DEVELOPMENT APPROACHES



AGILE DEVELOPMENT

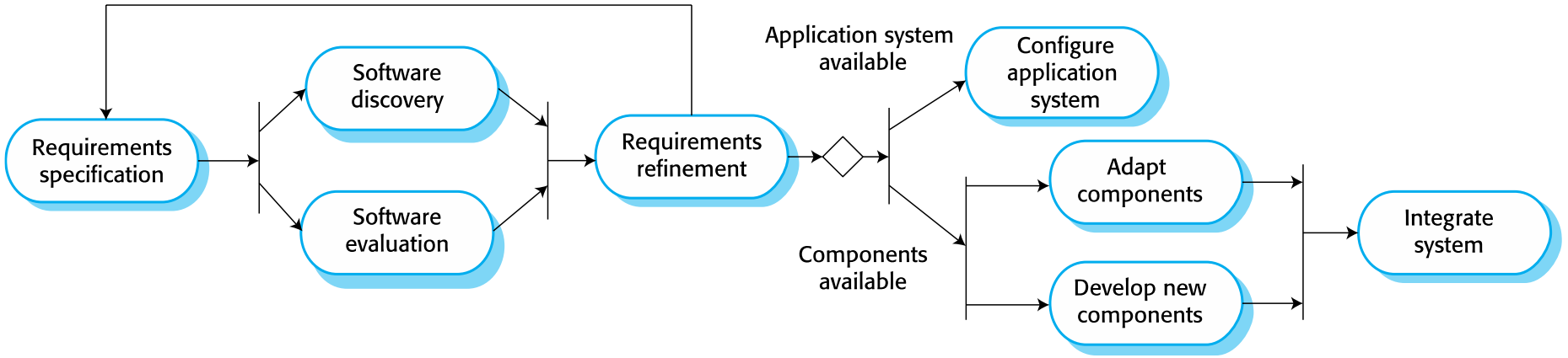


PROs	**CONs**
More flexible	Hard to predict
Product get to market faster	Final product is not released first
Better communication	Documentation gets left behind

REUSE-ORIENTED SOFTWARE ENGINEERING

- ✓ Based on software reuse where systems are integrated from existing components or application systems (COTS - Commercial-off-the-shelf) systems).
  - Stand-alone application systems (COTS)
  - Package objects / component framework such as .NET or J2EE.
  - Web services
- ✓ Reused elements may be configured to adapt their behaviour and functionality to a user's requirements
- ✓ Reuse is now the standard approach for building many types of business system

REUSE-ORIENTED SOFTWARE ENGINEERING

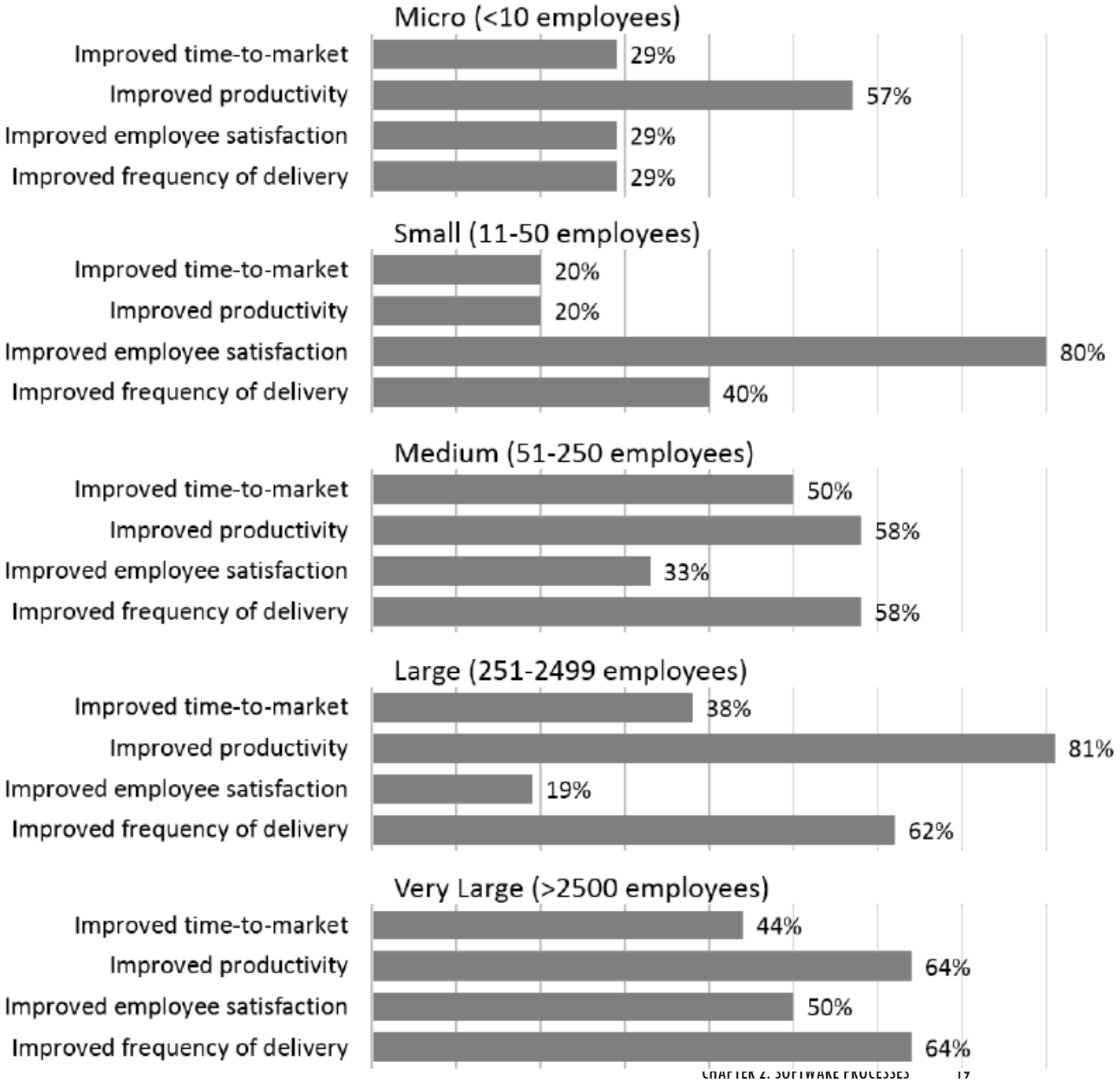


ADVANTAGES AND DISADVANTAGES

- ✓ Reduced costs and risks as less software is developed from scratch
- ✓ Faster delivery and deployment of system
- ✓ But requirements compromises are inevitable so system may not meet real needs of users
- ✓ Loss of control over evolution of reused system elements

"HYBRID DEVELOPI APPROACHES IN S SYSTEMS DEVELOP

Many companies face the pro development approach fitting



Kuhrmann, M., P. Diebold, J.
"Hybrid Software Developme
Software 36 (4): 20–31. https

A DISCUSSION CASE

https://docs.google.com/document/d/1S05m-iNqgjqhPTAkPJM6FfoXdnmv6EO4q_drPjeXqTA/edit?usp=sharing

PROCESS ACTIVITIES

PROCESS ACTIVITIES

- Real software processes are **inter-leaved sequences** of technical, collaborative and managerial activities with the overall goal of specifying, designing, implementing and testing a software system.

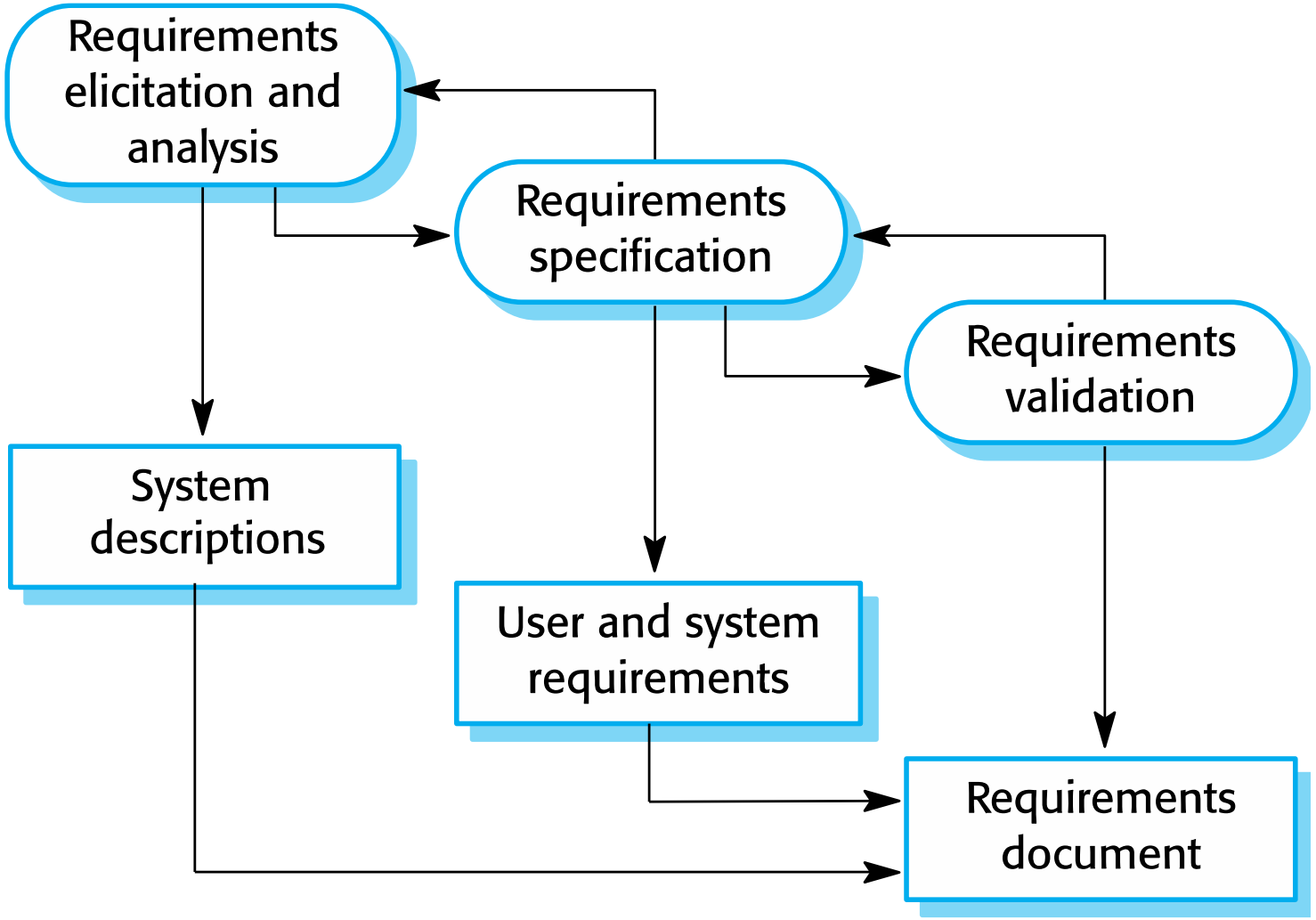
- The four basic process activities (specified in your book) of specification, development, validation and evolution are **organized differently in different development processes**.

- In the waterfall model, they are organized in **sequence**, whereas in incremental development they are i**nter-leaved**.

ACTIVITY: SOFTWARE SPECIFICATION

- ✓ The process of establishing what services are required and the constraints on the system's operation and development.
- ✓ Feasibility study
- ✓ Use: Requirements engineering process
  - Requirements elicitation and analysis
  - Requirements specification
  - Requirements validation

THE REQUIREMENTS ENGINEERING PROCESS

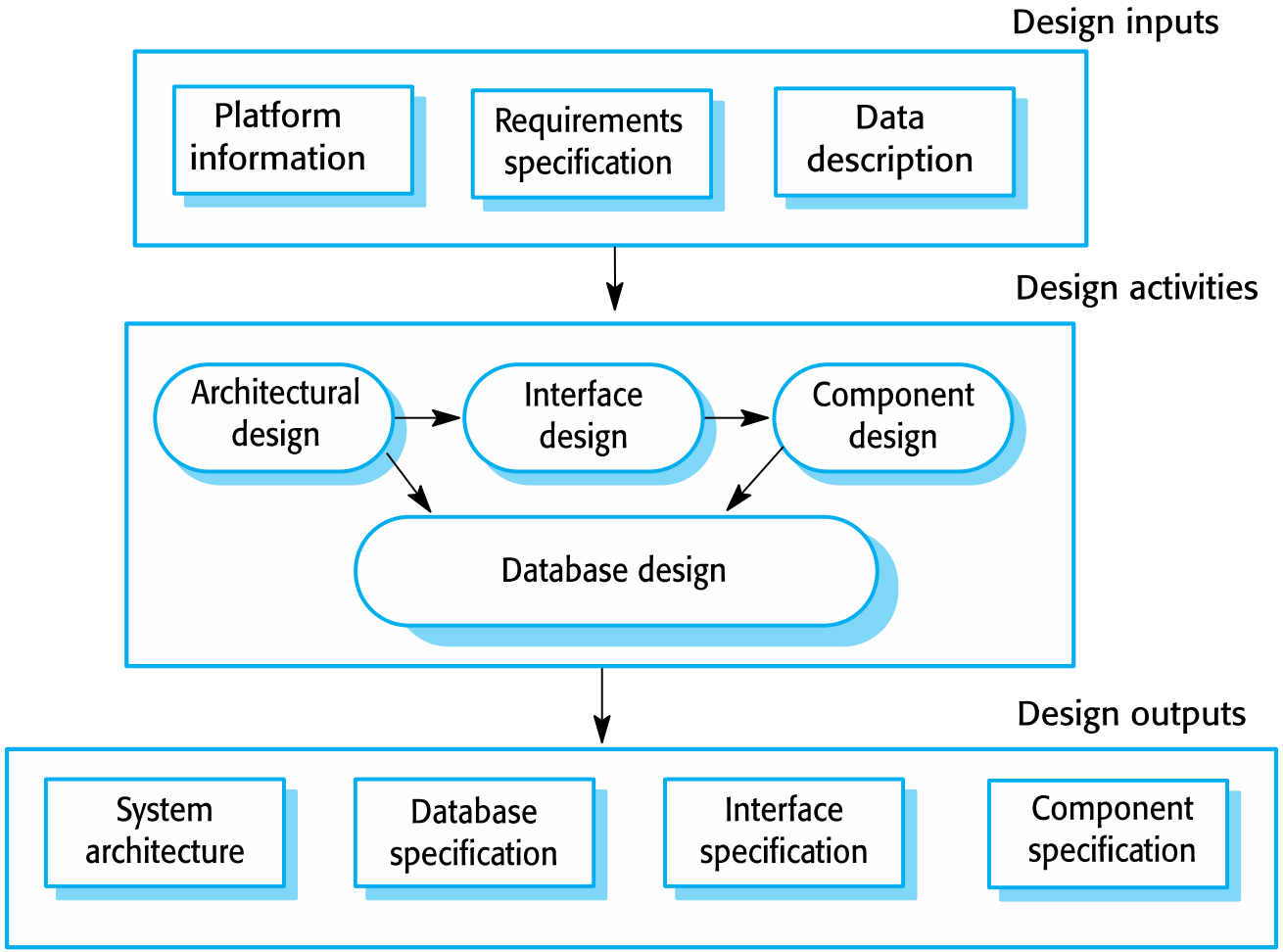


ACTIVITY: SOFTWARE DESIGN AND IMPLEMENTATION ~ SOFTWARE DEVELOPMENT

- ✓ The process of converting the system specification into an executable system.

- ✓ Two (sub) activities:
  - Software design
    - Design a software structure that realises the specification;
  - Implementation
    - Translate this structure into an executable program;
  - The activities of design and implementation are closely related and may be inter-leaved.

A GENERAL MODEL OF THE DESIGN PROCESS



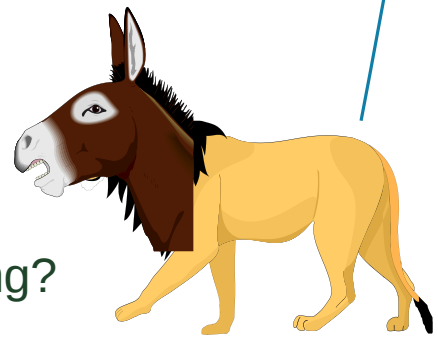
SYSTEM IMPLEMENTATION

- ✓ The software is implemented either by developing a program or programs or by configuring an application system.
- ✓ Design and implementation are interleaved activities for most types of software system.
- ✓ Programming is an individual activity with no standard process.
- ✓ Debugging is the activity of finding program faults and correcting these faults.

ACTIVITY: SOFTWARE VALIDATION

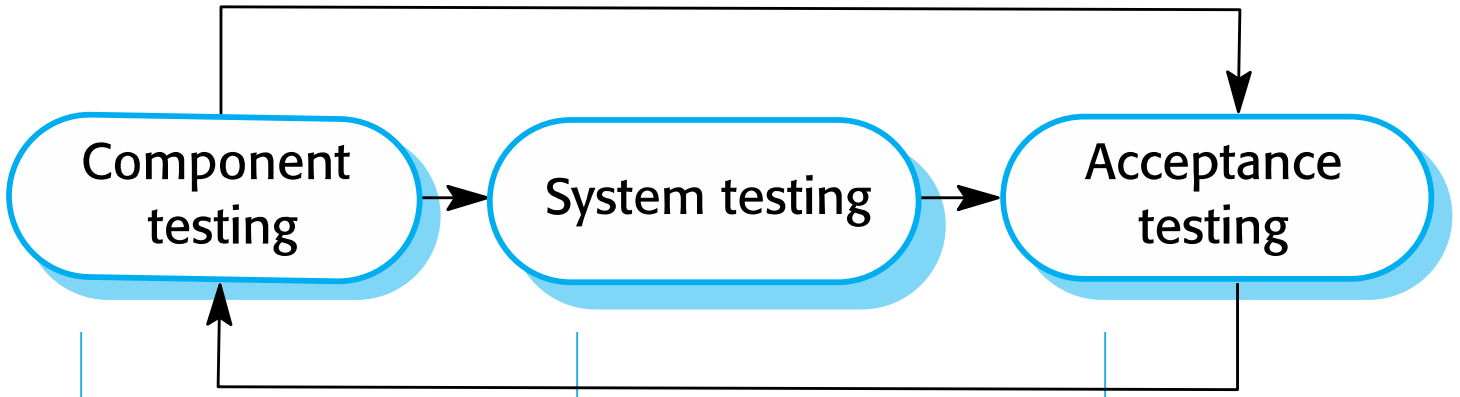
building the thing right?

- ✓ Verification and validation (V & V)
  - to show that a system conforms to its specification and meets the requirements of the system customer.
- ✓ Involves checking and review processes and system testing.
  - System testing: executing the system with test cases
  - Testing: the most commonly used V & V activity.



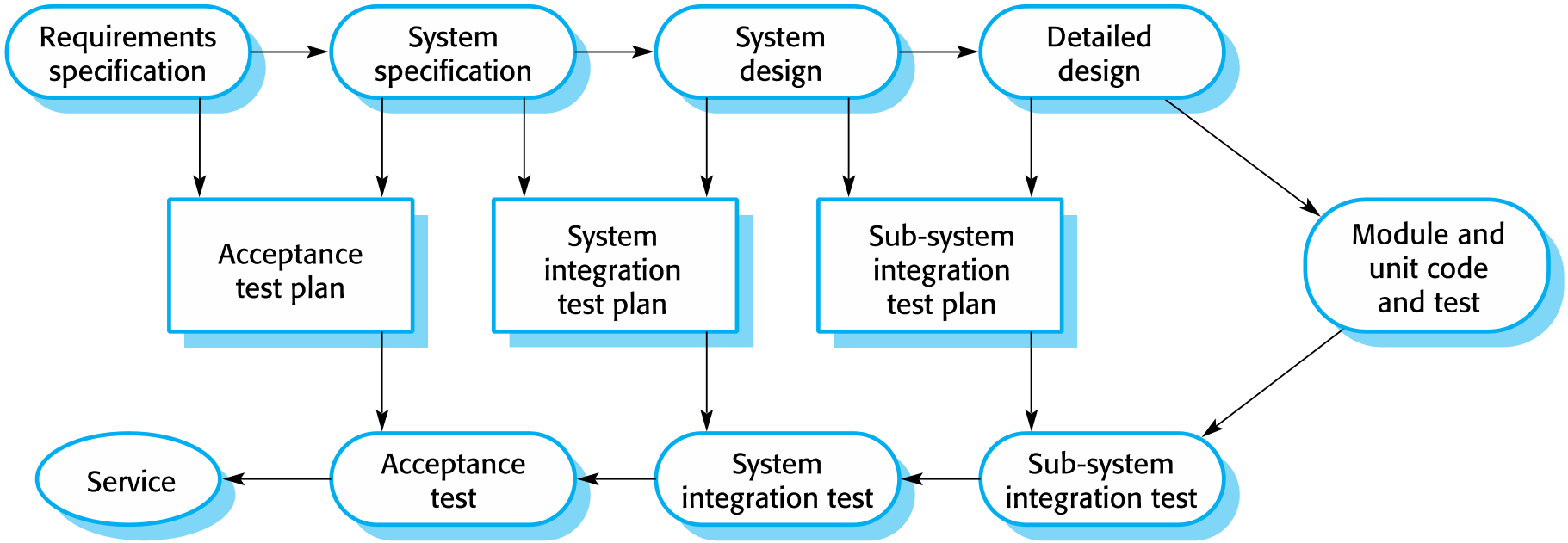
building the right thing?

STAGES OF TESTING



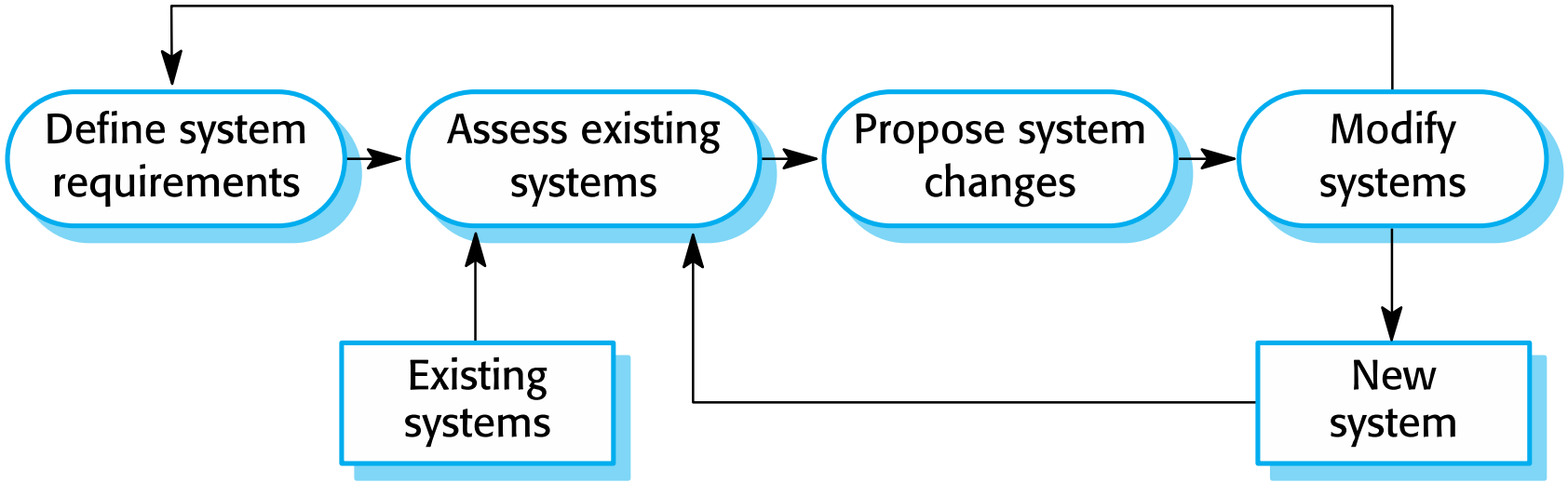
- Test individual components independently
- Testing of the system as a whole
- Testing with customer data

TESTING PHASES IN A PLAN-DRIVEN SOFTWARE PROCESS



ACTIVITY: SOFTWARE EVOLUTION

- ✓ Software is inherently flexible and can change.
- ✓ Requirements can change
  - (changing business circumstances) => the software must also evolve and change.



COPING WITH CHANGE

COPING WITH CHANGE

- ✓ Change is inevitable in all large software projects.
  - Business changes
  - New technologies
  - Changing platforms

- ✓ Change leads to rework
  - costs include rework (re-analysing requirements) and implementing new functionality

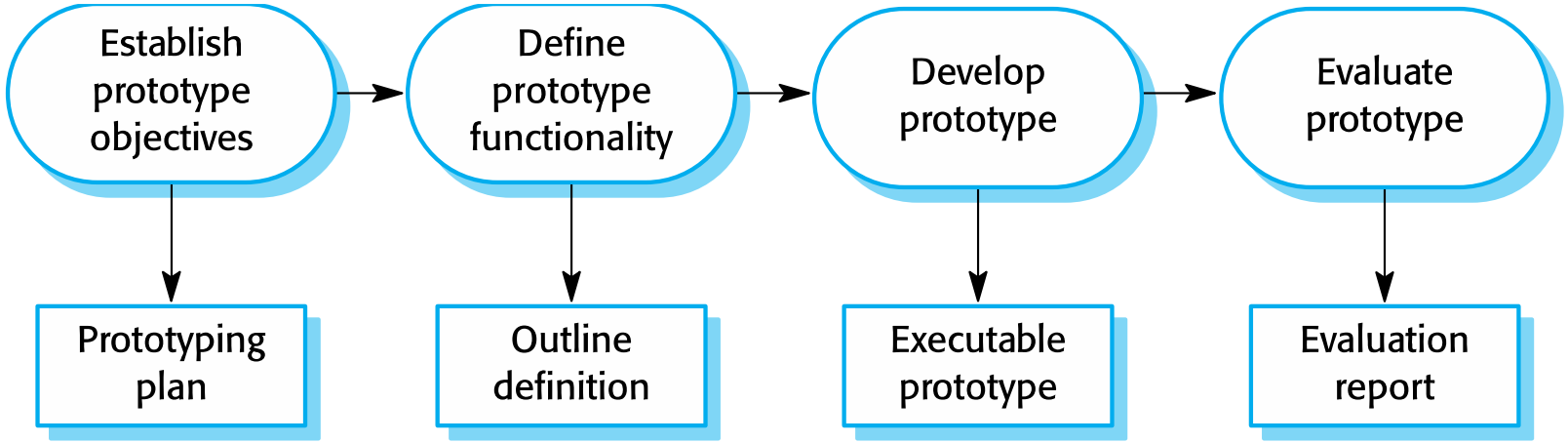
SOFTWARE PROTOTYPING

- ✓ A prototype is an initial version of a system used to demonstrate concepts and try out design options.
- ✓ A prototype can be used in:
  - requirements engineering process: requirements elicitation and validation;
  - design processes: options and develop UI design;
  - testing process: run back-to-back tests.

khám phá

Benefits:
- Improved system usability.
- A closer match to users' real needs.
- Improved design quality.
- Improved maintainability.
- Reduced development effort.

THE PROCESS OF PROTOTYPE DEVELOPMENT



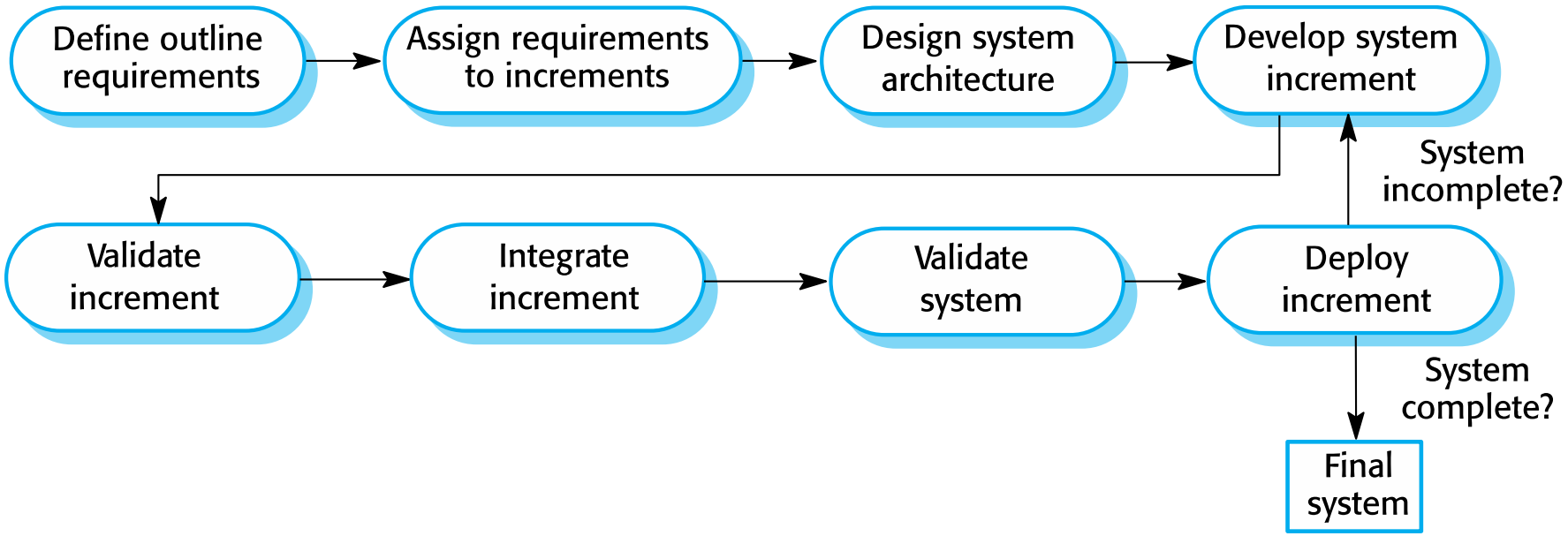
Prototype development:

- May be based on rapid prototyping languages or tools
- May involve leaving out functionality

INCREMENTAL DELIVERY

- ✓ The development and delivery is broken down into increments
  - each increment delivering part of the required functionality.
  - user requirements are prioritised and the highest priority requirements are included in early increments.
- ✓ Two approaches:
  - Incremental development: by developer
  - Incremental delivery: for end-user

INCREMENTAL DELIVERY



Advantages:

- system functionality is available earlier.
- early increments act as a prototype
- lower risk of overall project failure.
- highest priority system services receive most testing.

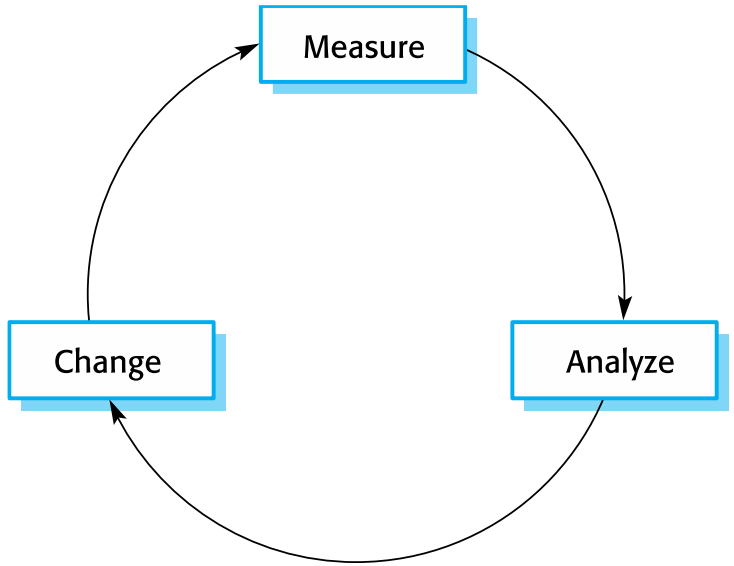
Problems:

- may require a set of basic facilities
- the specification is developed in conjunction with the software.

PROCESS IMPROVEMENT

PROCESS IMPROVEMENT

- ✓ Software process improvement
  - enhancing the quality of software,
  - reducing costs
  - or accelerating development processes.
- ✓ Process improvement
  - understanding existing processes
  - and changing these processes



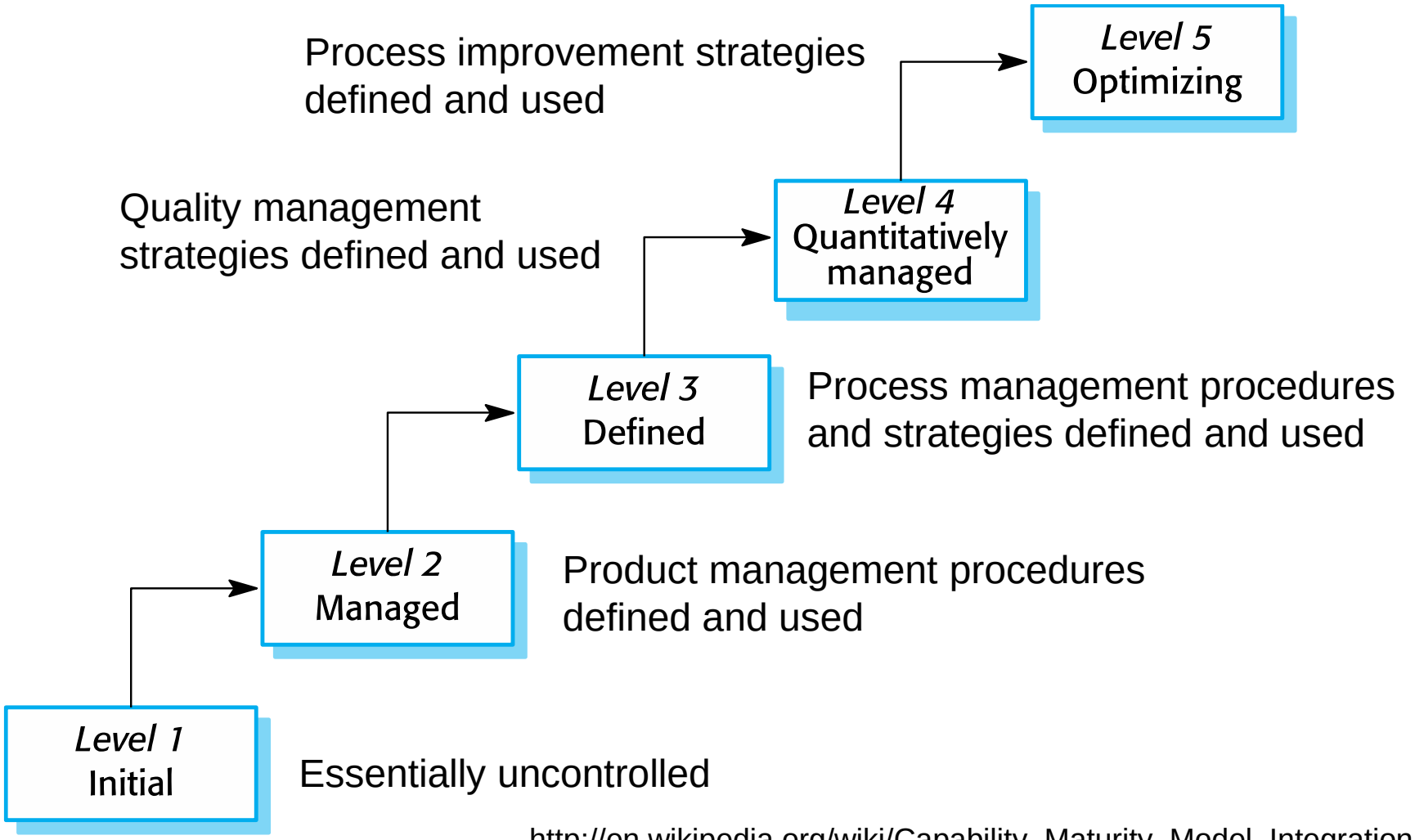
PROCESS IMPROVEMENT ACTIVITIES

- ✓ Process measurement
  - You measure one or more attributes of the software process or product. These measurements forms a baseline that helps you decide if process improvements have been effective.
- ✓ Process analysis
  - The current process is assessed, and process weaknesses and bottlenecks are identified. Process models (sometimes called process maps) that describe the process may be developed.
- ✓ Process change
  - Process changes are proposed to address some of the identified process weaknesses. These are introduced and the cycle resumes to collect data about the effectiveness of the changes.

THE CAPABILITY MATURITY MODEL (CMM)

- ✓ Capability Maturity Model Integration (CMMI) is a process level improvement training and appraisal program
- ✓ CMMI defines the most important elements that are required to build great products, or deliver great service
- ✓ It is required by many U.S. Government contracts, especially in software development.

THE CAPABILITY MATURITY MODEL (CMM)



http://en.wikipedia.org/wiki/Capability_Maturity_Model_Integration

SOFTWARE PROJECT DOCUMENTATION

Activity	Document
Validation & Verification	**SVVP** - Software Validation & Verification Plan
Quality Assurance	**SQAP** - Software Quality Assurance Plan
Configuration	**SCMP** - Software Configuration Management Plan
Project status	**SPMP** - Software Project Management Plan
Requirements	**SRS** - Software Requirements Specifications
Design	**SDD** - Software Design Document / Software Detail Design Document
Code	Source **Code**
Testing	**STD** - Software Test Document
Operation	User's **Manual**

SUMMARY

- ✓ Software processes
- ✓ Software process models
  - waterfall, incremental development, reuse-oriented development.
- ✓ Fundamental activities:
  - Requirements engineering: developing specification.
  - Design and implementation: transforming a requirements specification into an executable software system
  - Software validation: checking that the system conforms to its specification.
  - Software evolution: change existing software systems to meet new requirements

SUMMARY (CONT.)

- ✓ Coping with change
  - prototyping
  - iterative development and delivery
- ✓ Process improvement
  - agile approaches, geared to reducing process overheads,
  - maturity-based approaches based on better process management
  - and the use of good software engineering practice.
- ✓ The SEI process maturity framework (CMM)
  - identifies maturity levels that essentially correspond to the use of good software engineering practice.