Engenharia de Software

José Cascais Brás

Tópico 3

UNIVERSIDADE
LUSÓFONA

Learning Objectives

SOFTWARE PROCESS MODELS

PROCESS ACTIVITIES

COPING WITH CHANGE

PROCESS IMPROVEMENT

AGENDA

1. THE SOFTWARE PROCESS MODEL
2. SOFTWARE PROCESS MODELS
3. PROCESS ACTIVITIES
4. COPING WITH CHANGE
5. PROCESS IMPROVEMENT
6. KEY POINTS

PROCESS ACTIVITIES

01

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.

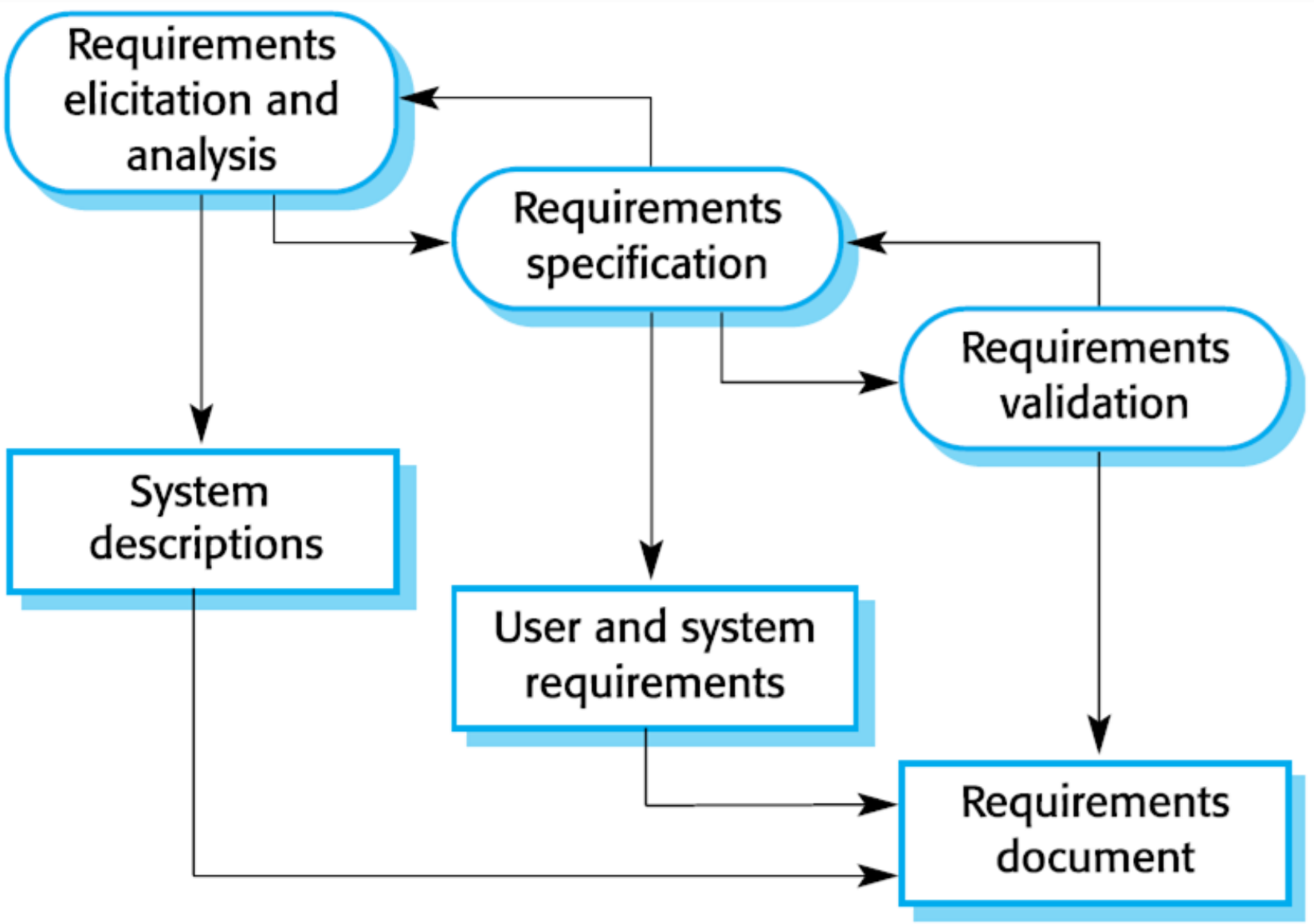
02

The four basic process activities of specification, development, validation and evolution are organized differently in **different development processes.**

03

For example, in the **waterfall model**, they are organized in sequence, whereas in **incremental development** they are interleaved.

PROCESS ACTIVITIES



PROCESS ACTIVITIES

Software Specifications

The process of establishing what services are required and the constraints on the system's operation and development.

Requirements engineering process

- **Requirements elicitation and analysis**
  What do the system stakeholders require or expect from the system?
- **Requirements specification**
  Defining the requirements in detail
- **Requirements validation**
  Checking the validity of the requirements

PROCESS ACTIVITIES

Software design and specification

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

Software design

- **Design a software structure that realizes the specification;**

Implementation

- **Translate this structure into an executable program;**

The activities of design and implementation are closely related and may be inter leaved.

PROCESS ACTIVITIES

Design activities

Architectural design

Where you identify the overall structure of the system, the principal components (subsystems or modules), their relationships and how they are distributed.

Database design

Where you design the system data structures and how these are to be represented in a database.

Interface design

where you define the interfaces between system components.

Component selection and design

Where you search for reusable components. If unavailable, you design how it will operate.

PROCESS ACTIVITIES

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..

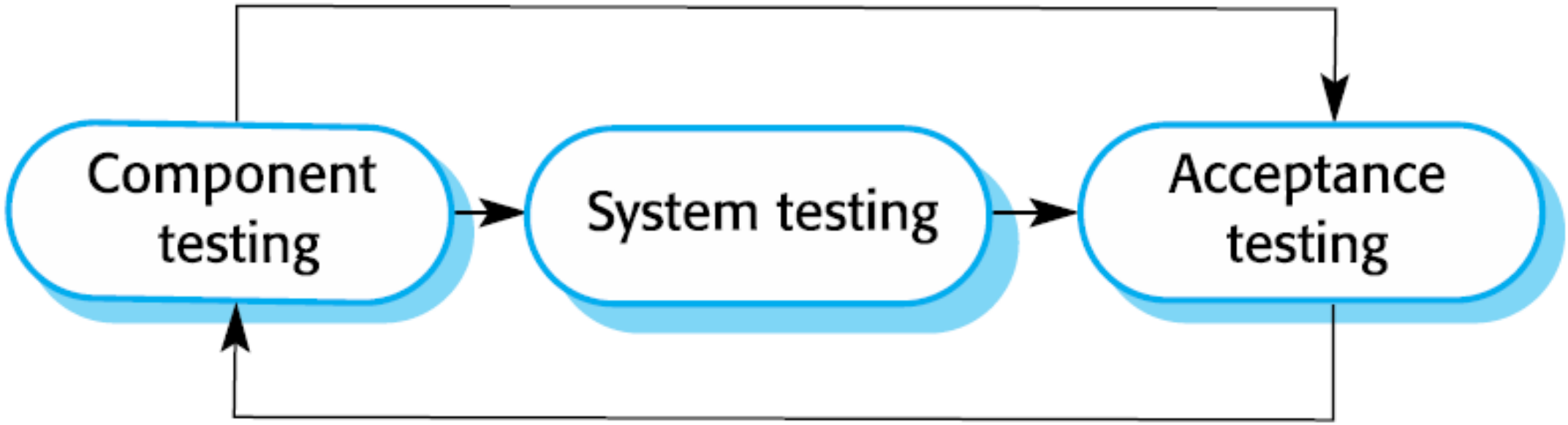
PROCESS ACTIVITIES

Software validation

- **Verification and validation** (V & V) is intended 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** involves executing the system with test cases that are derived from the specification of the real data to be processed by the system.
- **Testing** is the most commonly used V & V activity.

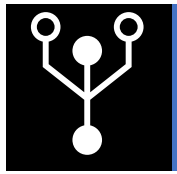
PROCESS ACTIVITIES

Stages of testing



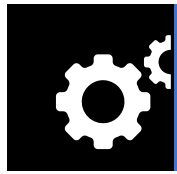
PROCESS ACTIVITIES

Testing Stages



COMPONENT TESTING

- Individual components are tested independently;
- Components may be functions or objects or coherent groupings of these entities.



SYSTEM TESTING

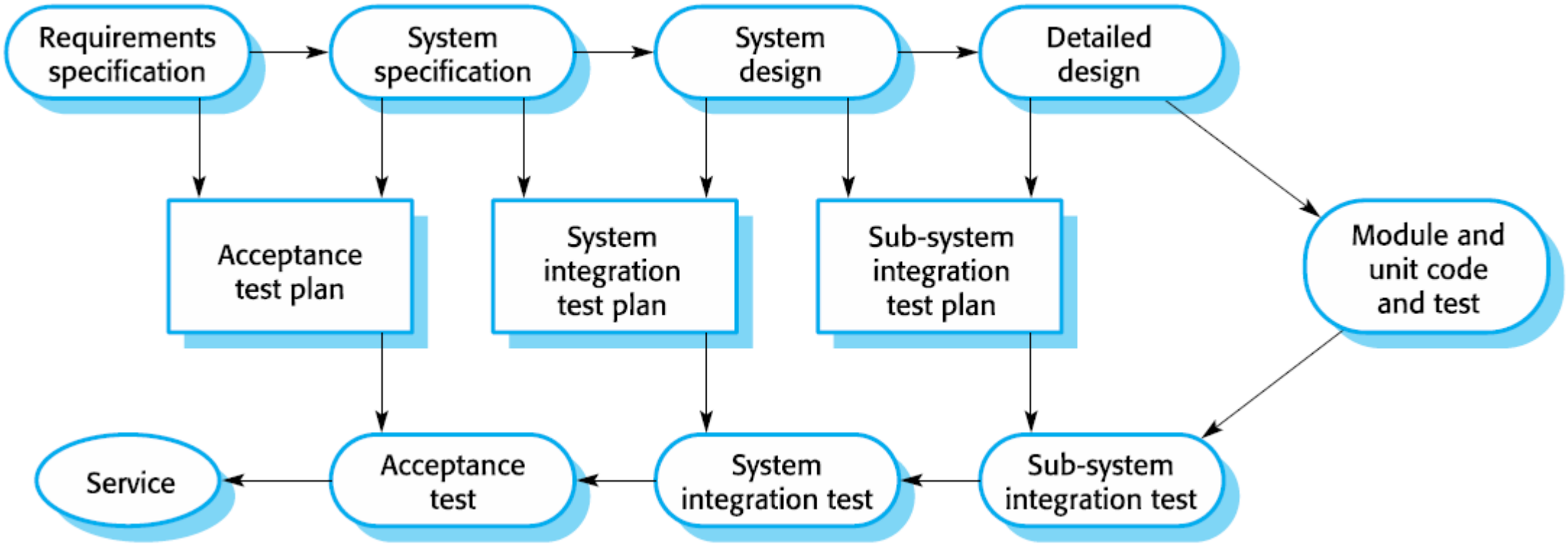
- Testing of the system as a whole. Testing of emergent properties is particularly important.

CUSTOMER TESTING

- Testing with customer data to check that the system meets the customer’s needs.

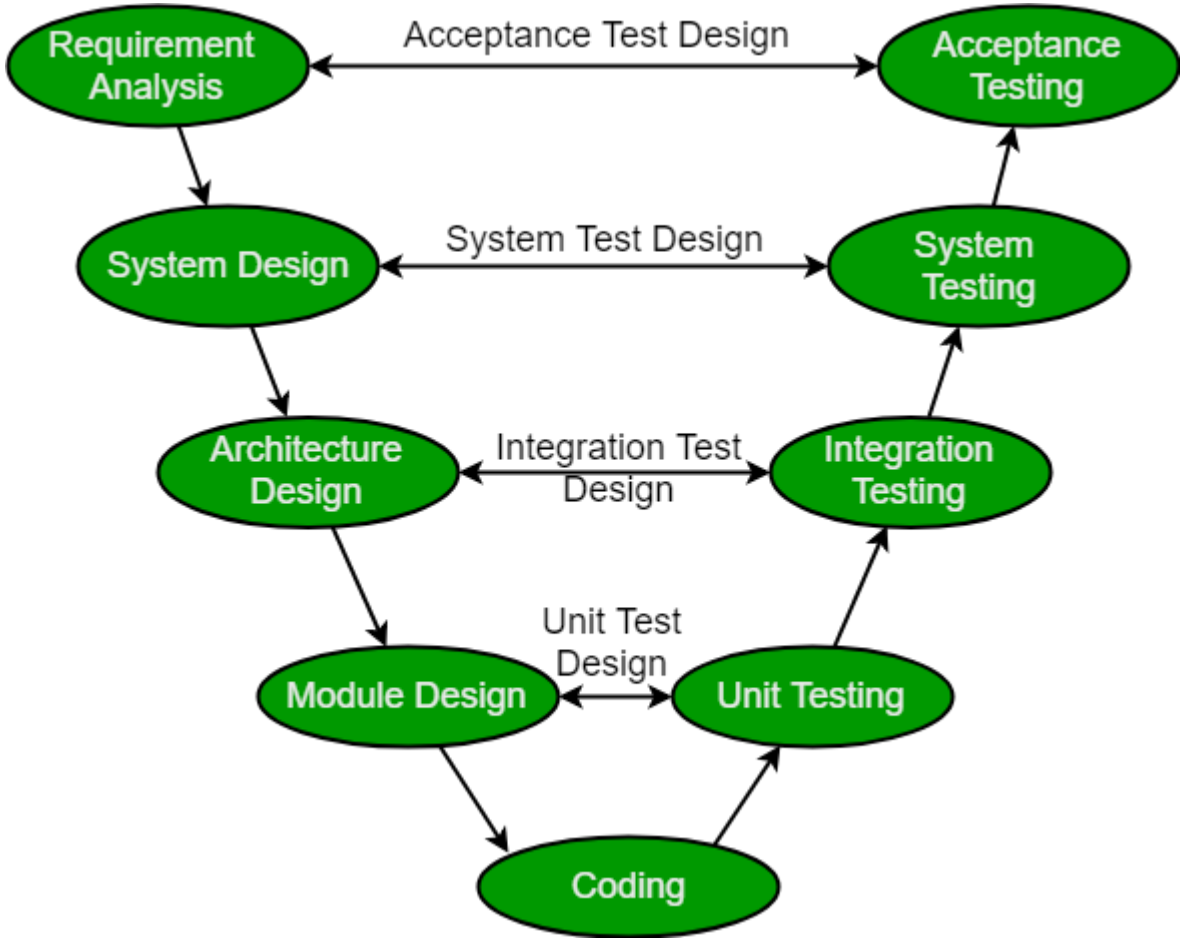
PROCESS ACTIVITIES

Testing phases in a plan-driven software process (V-model)



PROCESS ACTIVITIES

Testing phases in a plan-driven software process (V-model)



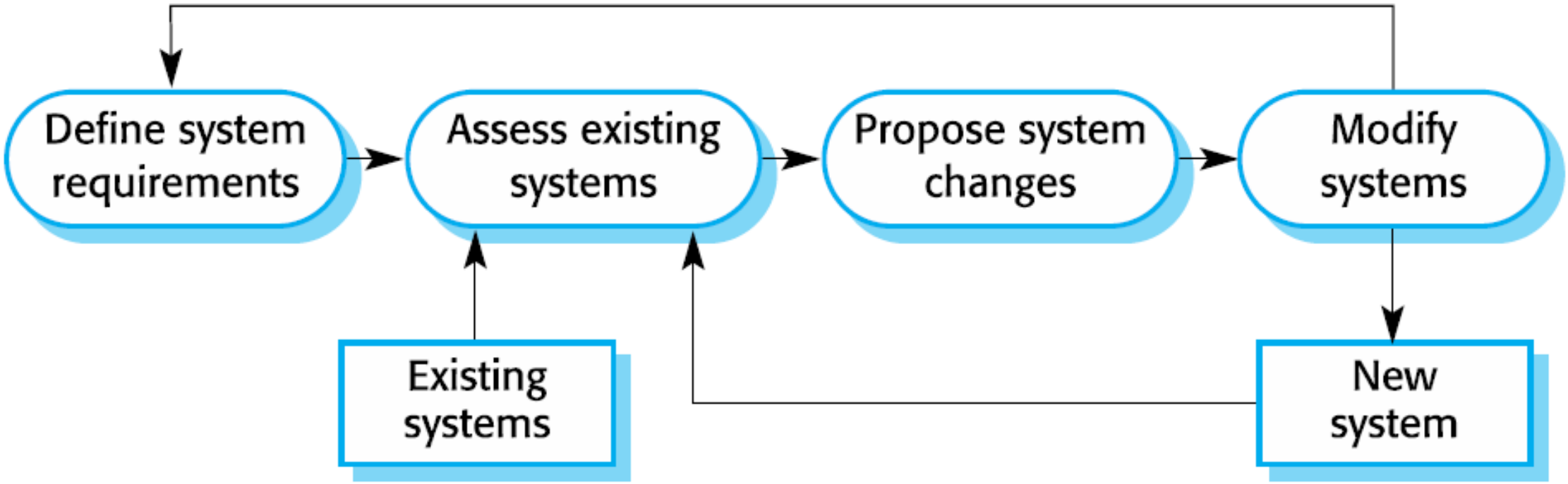
PROCESS ACTIVITIES

Software evolution

- Software is inherently flexible and can change.
- As requirements change through changing business circumstances, the software that supports the business must also evolve and change.
- Although there has been a demarcation between development and evolution (maintenance) this is increasingly irrelevant as fewer and fewer systems are completely new.

PROCESS ACTIVITIES

Software evolution



AGENDA

1. THE SOFTWARE PROCESS MODEL
2. SOFTWARE PROCESS MODELS
3. PROCESS ACTIVITIES
4. COPING WITH CHANGE
5. PROCESS IMPROVEMENT
6. KEY POINTS

COPING WITH CHANGE

- Change is inevitable in all large software projects.
  - Business changes lead to new and changed system requirements
  - New technologies open up new possibilities for improving implementations
  - Changing platforms require application changes
- Change leads to rework so the costs of change include both rework (e.g. re-analyzing requirements) as well as the costs of implementing new functionality

COPING WITH CHANGE

Reducing the costs of rework

- **Change anticipation**, where the software process includes activities that can anticipate possible changes before significant rework is required.
  - For example, a prototype system may be developed to show some key features of the system to customers.
- **Change tolerance**, where the process is designed so that changes can be accommodated at relatively low cost.
  - This normally involves some form of incremental development. Proposed changes may be implemented in increments that have not yet been developed. If this is impossible, then only a single increment (a small part of the system) may have be altered to incorporate the change.

COPING WITH CHANGE

Coping with changing requirements

System prototyping, where a version of the system or part of the system is developed quickly to check the customer's requirements and the feasibility of design decisions.

- This approach supports change anticipation.

Incremental delivery, where system increments are delivered to the customer for comment and experimentation.

- This supports both change avoidance and change tolerance.

COPING WITH CHANGE

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:

- The requirements engineering process to help with requirements elicitation and validation;
- In design processes to explore options and develop a UI design;
- In the testing process to run back-to-back tests.

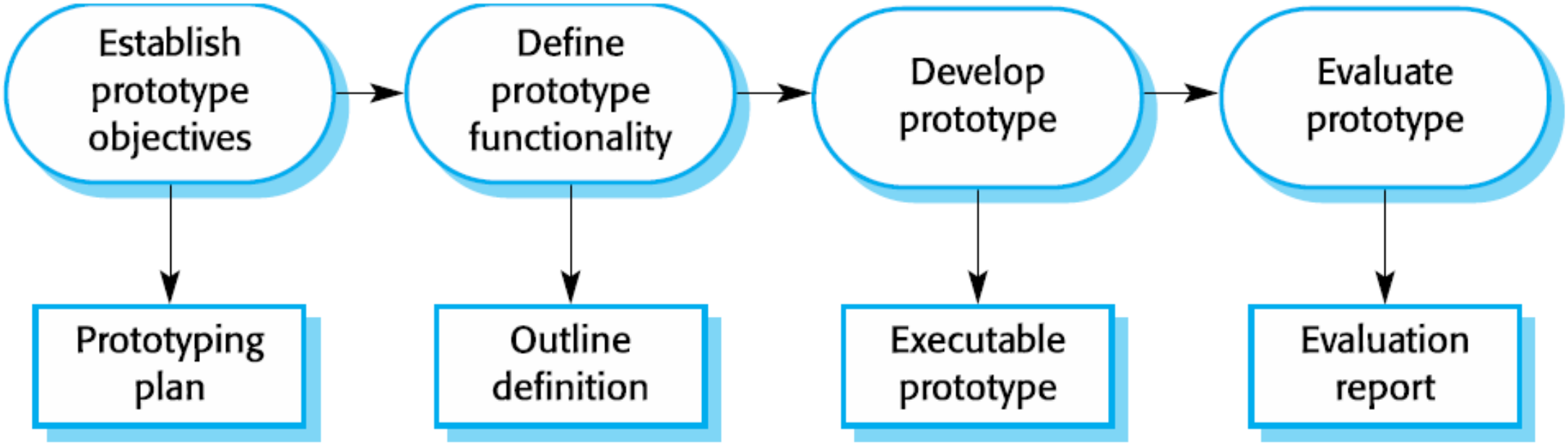
COPING WITH CHANGE

Benefits of prototyping

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

COPING WITH CHANGE

The process of prototype development



COPING WITH CHANGE

Prototype development

- ❑ May be based on rapid prototyping languages or tools
- ❑ May involve leaving out functionality
  - Prototype should focus on areas of the product that are not well-understood;
  - Error checking and recovery may not be included in the prototype;
  - Focus on functional rather than non-functional requirements such as reliability and security

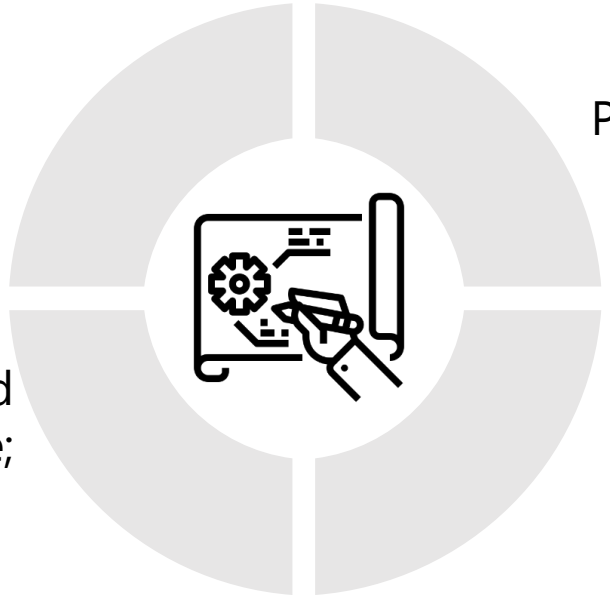
COPING WITH CHANGE

Throw-away prototypes

Prototypes should be discarded after development as they are not a good basis for a production system:

It may be impossible to tune the system to meet non-functional requirements;

The prototype structure is usually degraded through rapid change;



Prototypes are normally undocumented;

The prototype probably will not meet normal organizational quality standards.

COPING WITH CHANGE

Incremental delivery

- Rather than deliver the system as a single delivery, the development and delivery is broken down into increments with each increment delivering part of the required functionality.
- User requirements are prioritized and the highest priority requirements are included in early increments.
- Once the development of an increment is started, the requirements are frozen though requirements for later increments can continue to evolve.

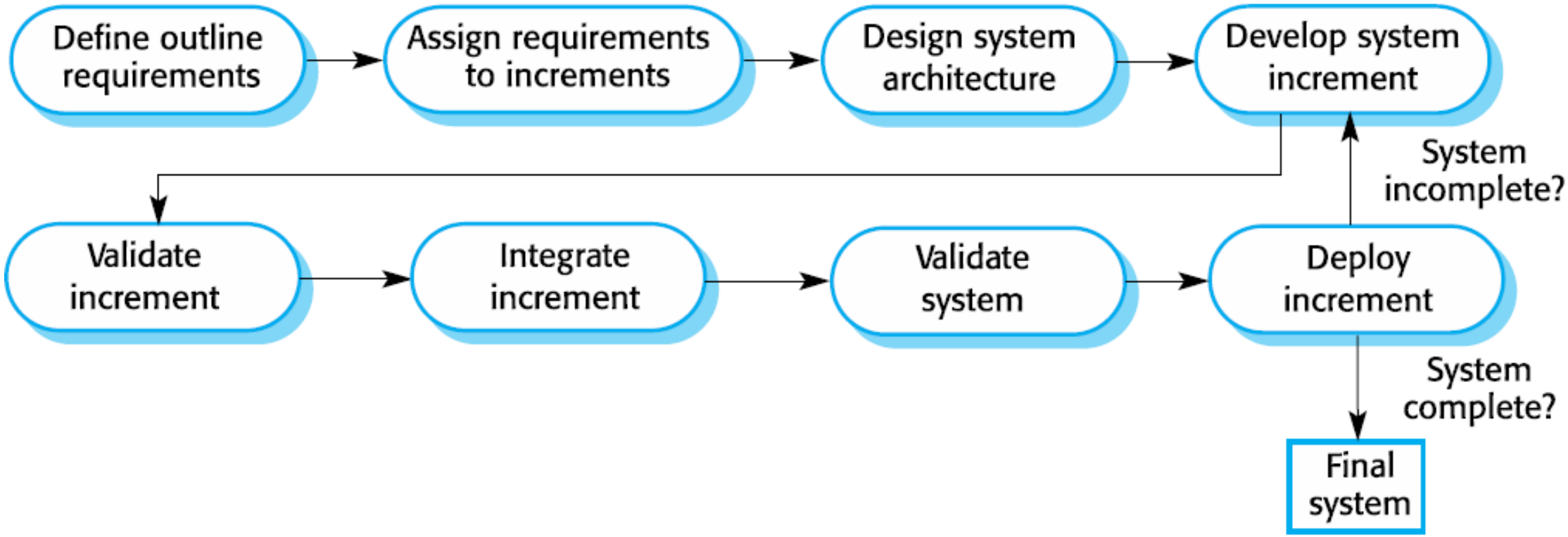
COPING WITH CHANGE

Incremental development and delivery

- **Incremental development**
  - Develop the system in increments and evaluate each increment before proceeding to the development of the next increment;
  - Normal approach used in agile methods;
  - Evaluation done by user/customer proxy.
- **Incremental delivery**
  - Deploy an increment for use by end-users;
  - More realistic evaluation about practical use of software;
  - Difficult to implement for replacement systems as increments have less functionality than the system being replaced.

COPING WITH CHANGE

Incremental delivery



COPING WITH CHANGE

Incremental delivery advantages

- Customer value can be delivered with each increment, so system functionality is available earlier.
- Early increments act as a prototype to help elicit requirements for later increments.
- Lower risk of overall project failure.
- The highest priority system services tend to receive the most testing.

COPING WITH CHANGE

Incremental delivery problems

Most systems require a set of basic facilities that **are used by different parts** of the system.

- As requirements are not defined in detail until an increment is to be implemented, it can be hard to identify common facilities that are needed by all increments.

The essence of iterative processes is that the specification is developed in conjunction with the software.

- However, this conflicts with the procurement model of many organizations, where the complete system specification is part of the system development contract.

AGENDA

1. THE SOFTWARE PROCESS MODEL
2. SOFTWARE PROCESS MODELS
3. PROCESS ACTIVITIES
4. COPING WITH CHANGE
5. PROCESS IMPROVEMENT
6. KEY POINTS

PROCESS IMPROVEMENT

- Many software companies have turned to software process improvement as a way of enhancing the quality of their software, reducing costs or accelerating their development processes.
- Process improvement means understanding existing processes and changing these processes to increase product quality and/or reduce costs and development time.

PROCESS IMPROVEMENT

The process **maturity approach**, which focuses on improving process and project management and introducing good software engineering practice.

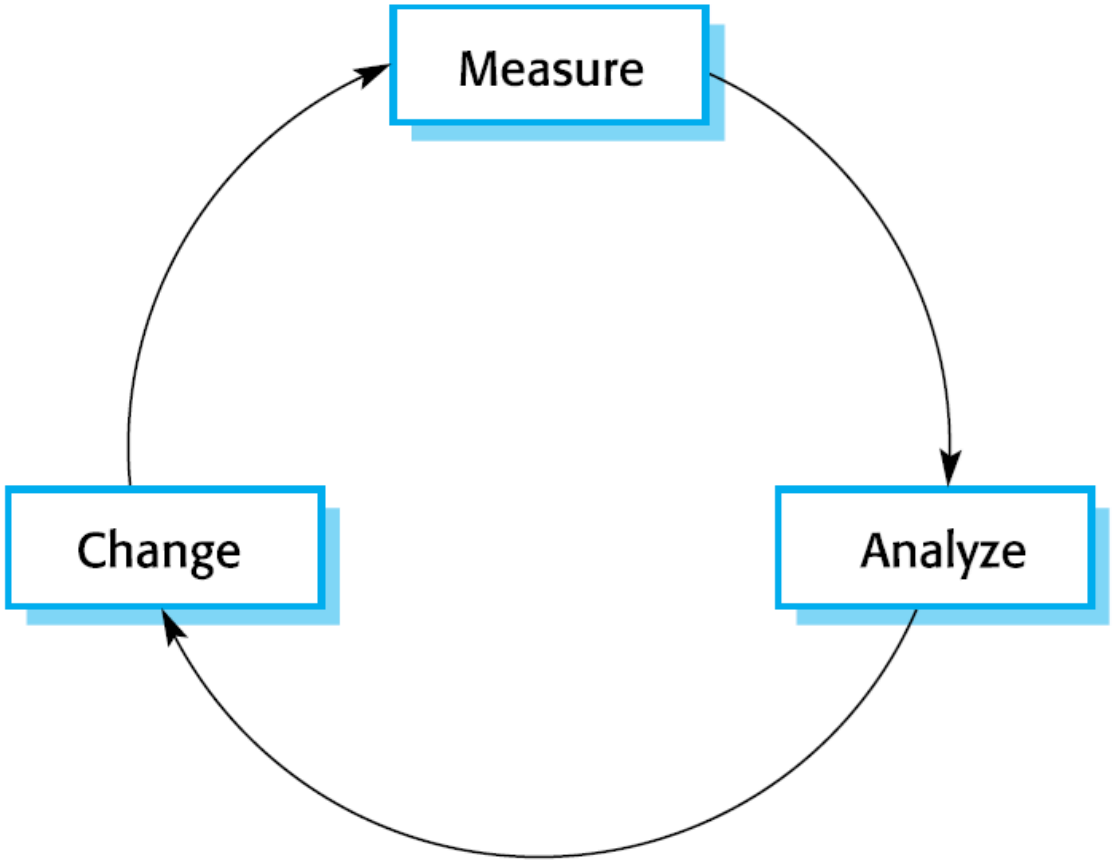
- The level of **process maturity reflects** the extent to which good technical and management practice **has been adopted** in organizational software development processes.

The agile approach, which focuses on iterative development and the reduction of overheads in the software process.

- The primary characteristics of agile methods are rapid delivery of functionality and responsiveness to changing customer requirements.

PROCESS IMPROVEMENT

The process improvement cycle



PROCESS IMPROVEMENT

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.

PROCESS IMPROVEMENT

Process Measurement

Wherever possible, quantitative process data should be collected

- However, where organizations do not have clearly defined process standards this is very difficult as you don't know what to measure. A process may have to be defined before any measurement is possible.

Process measurements should be used to assess process improvements

- But this does not mean that measurements should drive the improvements. The improvement driver should be the organizational objectives.

PROCESS IMPROVEMENT

Process Metrics

Time taken for process activities to be completed

- ❑ E.g. Calendar time or effort to complete an activity or process.

Resources required for processes or activities

- ❑ E.g. Total effort in person-days.

Number of occurrences of a particular event

- ❑ E.g. Number of defects discovered.

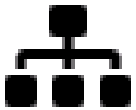
PROCESS IMPROVEMENT

CMMI Development is an integrated set of best practices that improves an organization's capability to develop quality products and services that meet the needs of customers and end users.

Improve Time-to-Market — ensure products and services are delivered quickly and efficiently with little to no re-work

Increase Quality — improve product development quality and consistency to reduce defects

Reduce Cost — lower costs through improved planning, scheduling, and budgeting processes.

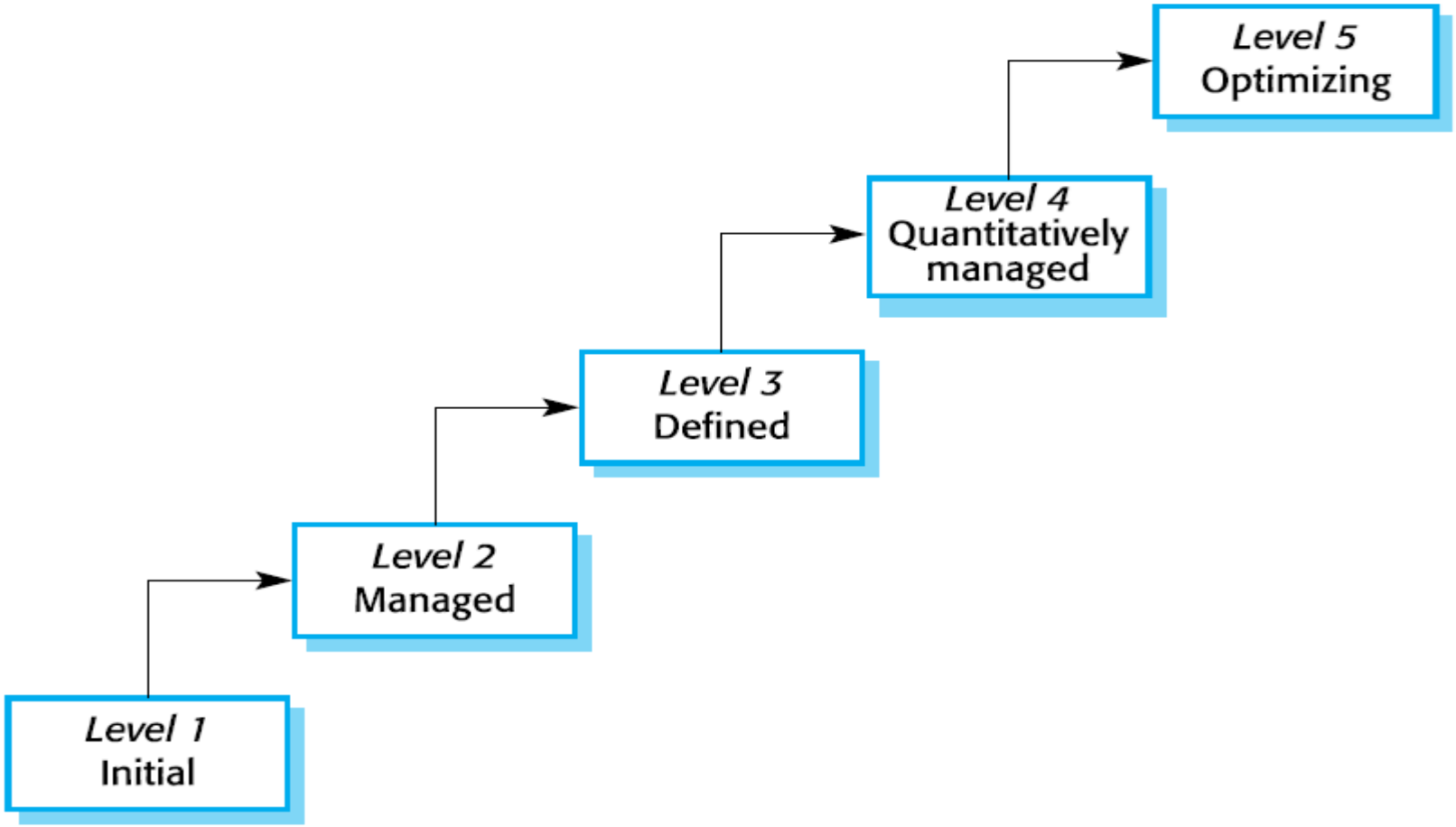


Gain Organizational Agility — leverage revenue-enhancing and cost-cutting opportunities to deliver products and services quickly, effectively, and consistently.

Improve Product Lifecycle Management — meet customer expectations across the entire product lifecycle from delivery to maintenance and operations.

PROCESS IMPROVEMENT

CAPABILITY MATURITY LEVELS



PROCESS IMPROVEMENT

01 Initial

- ❑ Essentially uncontrolled

02 Repeatable

- ❑ Product management procedures defined and used

03 Defined

- ❑ Process management procedures and strategies defined and used

04 Managed

- ❑ Quality management strategies defined and used

05 Optimizing

- ❑ Process improvement strategies defined and used

PROCESS IMPROVEMENT

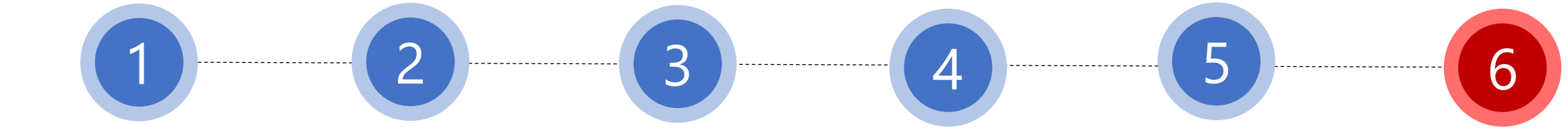
THE SEI CAPABILITY MATURITY MODEL

The four capability levels, each a layer in the foundation for ongoing process improvement, are designated by the numbers 0 through 3:

A capability level for a process area is achieved when all of the generic goals are satisfied up to that level. The fact that capability levels 2 and 3 use the same terms as generic goals 2 and 3 is intentional because each of these generic goals and practices reflects the meaning of the capability levels of the goals and practices.

CMM was developed and is promoted by the Software Engineering Institute (**SEI**), a research and development center promote by the U.S. Department of Defense (DOD).

AGENDA



1. THE SOFTWARE PROCESS MODEL
2. SOFTWARE PROCESS MODELS
3. PROCESS ACTIVITIES
4. COPING WITH CHANGE
5. PROCESS IMPROVEMENT
6. KEY POINTS

KEY POINTS

- Software processes are the activities involved in producing a software system. Software process models are abstract representations of these processes.
- General process models describe the organization of software processes.
  - Examples of these general models include the ‘waterfall’ model, incremental development, and reuse-oriented development.
- Requirements engineering is the process of developing a software specification.

KEY POINTS

- Design and implementation processes are concerned with transforming a requirements specification into an executable software system.
- Software validation is the process of checking that the system conforms to its specification and that it meets the real needs of the users of the system.
- Software evolution takes place when you change existing software systems to meet new requirements. The software must evolve to remain useful.
- Processes should include activities such as prototyping and incremental delivery to cope with change.

KEY POINTS

- Processes may be structured for iterative development and delivery so that changes may be made without disrupting the system as a whole.
- The principal approaches to process improvement are agile approaches, geared to reducing process overheads, and maturity-based approaches based on better process management and the use of good software engineering practice.
- The SEI process maturity framework identifies maturity levels that essentially correspond to the use of good software engineering practice.

Q&A