

## CHAPTER 2: SOFTWARE PROCESSES (W2)

# Outline

2

- IEEE Std. 610 – Definition of terms
- Software Engineering Principles
- Waterfall Model
- V-Model
- Barry Boehm's Spiral Model
- Rational Unified Process (RUP)
- Agile Development
- Software Product Lines
- Fundamental Process Activities
- Large Project Org Chart
- RACI Explained
- Software Process Improvement
- CMMI Explained
- Key Points
- Exercises

# Class Objectives

3

The objective of this class is to introduce a concept of a software process – a set of activities for software production.

At the end of this class you will:

- understand the concepts of **software processes**;
- become familiar with the common **process models** and when they can be used;
- understand the fundamental **process activities** that include requirements specification, software development, software testing, and evolution;
- become familiar with the **CMMI** process maturity framework;
- understand the notion of software **process improvement**.

# Key Software Engineering Principles

4

Some key principles apply to all types of software systems, irrespective of the development techniques used:

- **Process**. Systems should be developed using a managed and understood development process.
- **Dependability**
- **Requirements**
- **Reuse**



In this lecture we discuss the most common types of software processes and fundamental software activities.

# IEEE Std.610 – Definition of Terms

5

## Process

A sequence of steps performed to achieve a given purpose; for example, the software development process [*delivers a software product*].

## Process Model

A visual *model* or representation of the sequential flow and control logic of a set of related activities or actions (see Slide 9 for detail).

## Software Life Cycle

The period of time that begins when a software product is conceived and ends when the software is no longer available for use. The software life cycle typically includes a concept phase, requirements phase, design phase, implementation phase, test phase, installation and checkout phase, operation and maintenance phase, and, sometimes, retirement phase.

## Requirement

- (1) A condition or capability needed by a user to solve a problem or achieve an objective.
- (2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.
- (3) A documented representation of a condition or capability as in (1) or (2)

# IEEE Std.610 – Definition of Terms

## Requirements Specification

A document that specifies the requirements for a system or component. Typically included are user requirements, functional requirements, performance requirements, interface requirements, design requirements, and development standards.

## Testing

- (1) The process of operating a system or component under specified conditions, observing or recording the results, and making an evaluation of some aspect of the system or component.
- (2) The process of analyzing a software item to detect the differences between existing and required conditions (that is, bugs) and to evaluate the features of the software items.

## Quality Assurance (QA)

- (1) A planned and systematic pattern of all actions necessary to provide adequate confidence that an item or product conforms to established technical requirements.
- (2) A set of activities designed to evaluate the process by which products are developed or manufactured.

**Software Testing** focuses on evaluating a software product, whereas the **Quality Assurance** discipline focuses on evaluating the processes by which software products are produced.

# Fundamental Software Engineering Activities

7

- A software process is a set of related activities that deliver a software product.
- There are many types of software systems, but there is no universally applicable software process.
- Project teams select a software process based on the context of their project that includes the type of software being developed, requirements of end-users, regulatory requirements, project timelines, etc.
- However, at a high level all processes must include the four fundamental activities:
  - **Software specification**, defining the functionality of a system.
  - **Software development**, delivering a working system.
  - **Software validation**, providing visibility into the degree to which the system conforms to its requirements.
  - **Software evolution**, dealing with the changing needs of end-users and aligning the system functionality with the customer needs.
- On real-life projects, each of these activities is usually defined as a process area that includes multiple practices that we will discuss in the class.

# What Process Models Include

- Process models include more than just activities or practices.
- The other important information includes:
  - Products or deliverables of process activities (tasks),
  - Definitions of roles and responsibilities of the people performing the tasks, which can be defined as a RACI table;
  - Pre- and post-conditions for process phases, a.k.a., Toll-Gates or Entry/Exit Criteria.
- As there is no universal process that is right for all software projects, most companies select a process framework and then customize it for their needs and context:
  - In the case of safety-critical and regulatory projects the most formal and structured software processes are required.
  - In the case of developing business applications with rapidly changing business needs, a more flexible, agile process can be a better fit.



# Most Common Software Process Models

The most common software process models, discussed in this class, include:

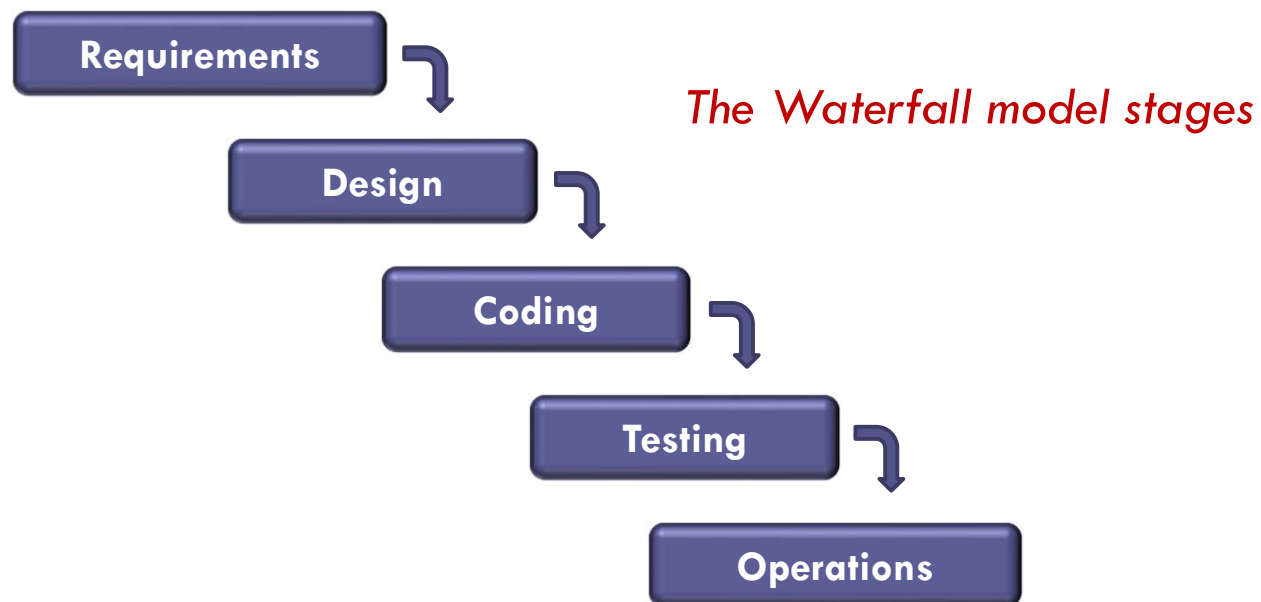
- **The Waterfall model.** A model in which the activities, including a concept phase, requirements phase, design phase, implementation phase, test phase, and installation and checkout phase, are performed in that order, possibly with overlap but with little or no iteration.
- **Iterative and Incremental Development.** A software development approach in which requirements definition, design, implementation, and testing occur in an overlapping, iterative (rather than sequential) manner, resulting in incremental delivery of the overall software product. Most common examples:
  - Rational Unified Process (RUP)
  - Agile development
- **Software Product Lines (SPL).** SPL focuses on software reuse and refers to software engineering methods, tools and techniques for creating a collection of similar software systems from a shared set of software assets using a common means of production.

*The following slides will discuss each of these models in detail.*

# The Waterfall Model History

10

- The first formal description of the waterfall model is often cited as a 1970 article by Winston W. Royce, although Royce did not use the term *waterfall* in that article.
- The term "waterfall" was introduced in the mid-1970s.
- In 1985, the United States Department of Defense included the Waterfall model in their standard DOD-STD-2167A for working with software development contractors. The standard required selected contractors to follow the Waterfall model.



# The Waterfall Model Stages Explained

11

The stages of the waterfall model reflect the fundamental software development activities:

- **Requirements analysis and definition.** The end-user needs are analyzed and then defined as a detailed system specification.
- **System design.** The design process allocates the system specification to software and hardware, delivers a system architecture and design specifications.
- **Coding and unit testing.** During this stage, the software design is implemented as executable programs. The unit testing is performed by developers to verify that a program is executable and meets its specifications.
- **Testing** (independent system testing and user acceptance testing). A complete, integrated system is tested to ensure that it meets its requirements.
- **Operation and Maintenance.** This stage focuses on installation and support in production of a complete system.

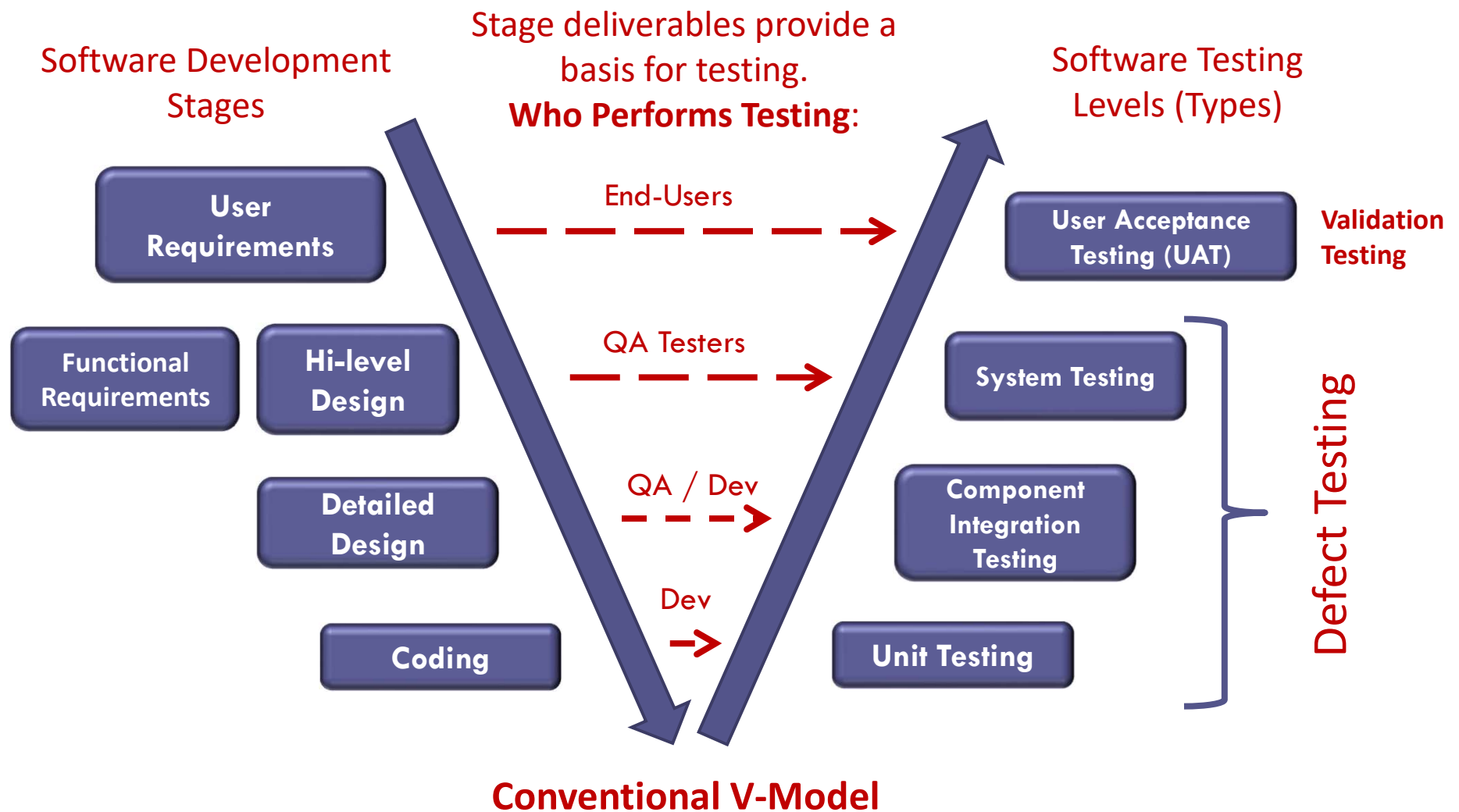
# The V-Model: Extension of the Waterfall Model

12

- With the growing complexity of software systems, the software testing process evolved and included various types of testing.
- To better align the test types with the Waterfall model stages, a V-Model emerged in the early 1990s.
- The **V-Model** represents a software development process, which can be considered an extension of the Waterfall model.
- Instead of moving down in a linear way, the process steps are bent upwards after the coding phase, to form the typical V shape.
- The V-Model demonstrates the relationships between each phase of the development lifecycle and its associated level of testing.

# Conventional Test Levels (Types)

13



# The Waterfall Model Criticism

14

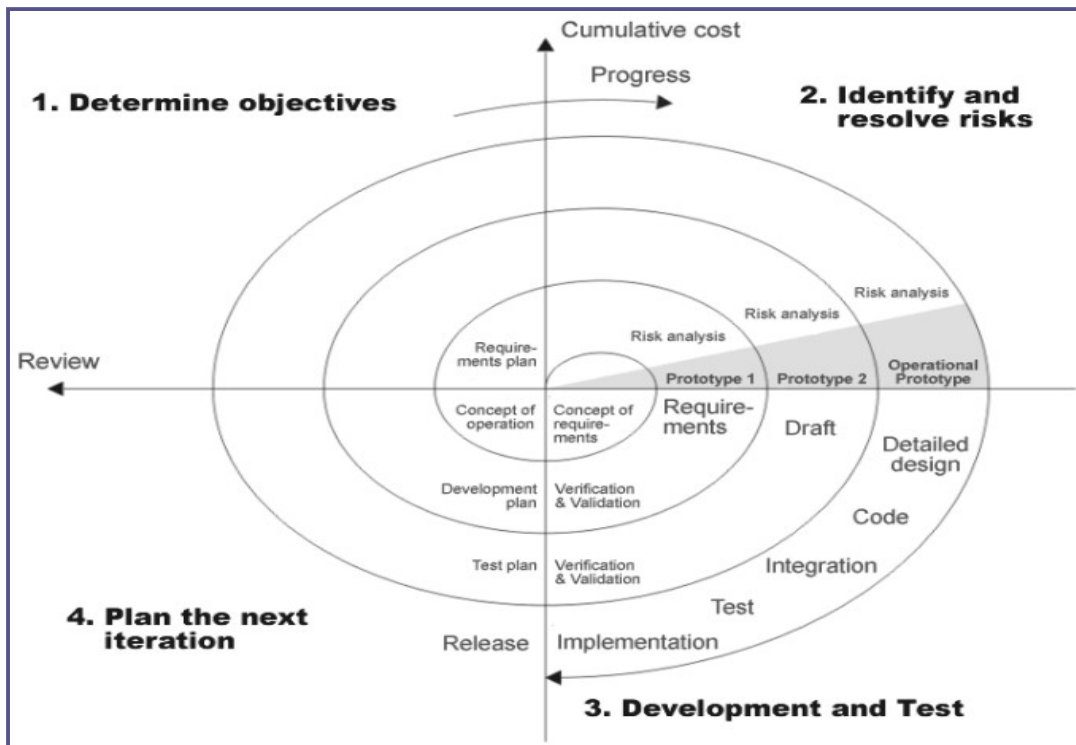
- When the Waterfall model was used for large projects, it frequently proved ineffective for the following reasons:
  - **clients may not know exactly what their requirements are** before they see working software, so they change their requirements, leading to redesign, redevelopment, and retesting, and increased costs;
  - it is **difficult to accurately estimate in advance project resources**, cost and schedule;
  - **developers may not be aware of project risks**, future difficulties when designing a new software product or feature;
- In response to these Waterfall Model problems, other process frameworks emerged that were based on the concept of **iterative and incremental development**.

# Barry Boehm's Spiral Model

15

- The **Spiral Model** was introduced and published by Barry Boehm in 1986.
- The Spiral Model is a risk-driven process model generator for software projects.
- Based on the unique risk patterns of a given project, the spiral model guides a team to adopt elements of one or more process models, such as incremental or waterfall.

- Each loop in the spiral represents a phase of the software process.
- The model combines change avoidance with change tolerance.
- It includes risk management activities to reduce these risks.
- It introduced prototypes.
- It includes lifecycle milestones.



# Advantages of Incremental Development

16

- The cost of accommodating changing customer requirements is reduced.
- The amount of analysis and documentation that has to be redone is much less than is required with the Waterfall model.
- It is easier to get customer feedback on the development work that has been done.
- Customers can comment on demonstrations of the software and see how much has been implemented.
- More rapid delivery and deployment of useful software to the customer is possible.
- Customers are able to use and gain value from the software earlier than is possible with the Waterfall process.



# Rational Unified Process (RUP) Explained

17

- The **Rational Unified Process** (RUP) is an iterative software development process framework created by the Rational Software Corporation, a division of IBM since 2003.
- RUP has its roots in the work that Ivar Jacobson did at Ericsson in the late 1960s.
- There are three main characteristics of the RUP process model:
  - **Use-case driven.** RUP places strong emphasis on building systems based on a thorough understanding of how the delivered system will be used, which is captured by developing use cases and scenarios.
  - **Architecture-centric.** RUP focuses on the early development and baselining of a software architecture in order to facilitate parallel development, system maintainability, and minimize rework.
  - **Iterative and Incremental.** An iterative approach advocates an increasing understanding of the problem through successive refinements and an incremental growth of an effective solution over multiple cycles.

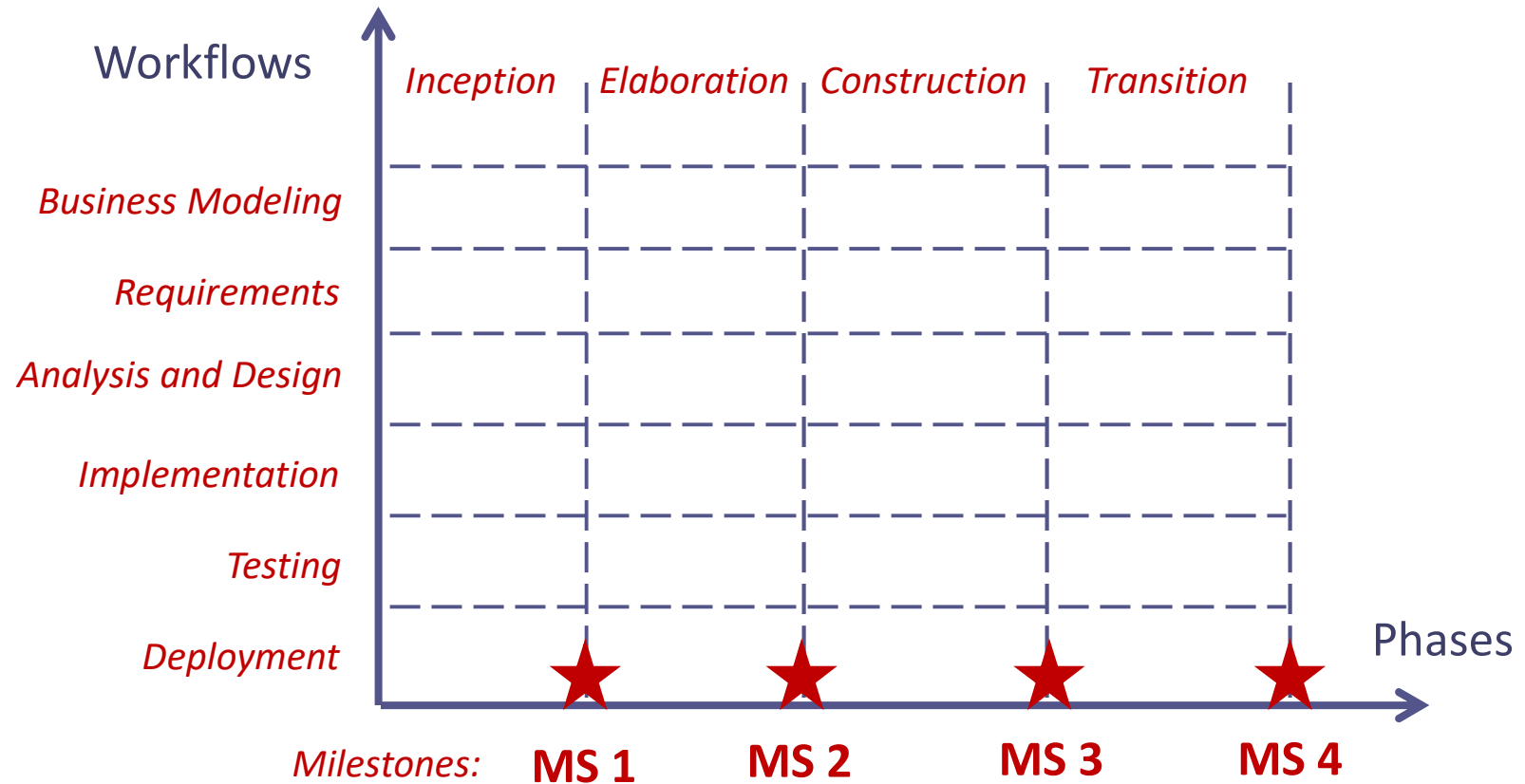
# RUP: Four Phases and Milestones

- In RUP, the development of a software system is represented as a series of iterative cycles.
- A cycle (iteration) ends with the release of a system version to customers.
- Each cycle includes four phases:
  - **Inception**. The primary goal of this phase is to establish the case for the viability of the proposed system. Deliverables include a business case, system scope, candidate architecture. The milestone associated with this phase is **Life-Cycle Objective (MS1)**.
  - **Elaboration**. The primary goal of this phase is to establish the ability to build the new system given the budget, schedule, and resources. The milestone associated with this phase is **Life-Cycle Architecture (MS2)**.
  - **Construction**. The primary goal of this phase is to build a system capable of operating. The milestone associated with this phase is Initial **Operational Capability (MS3)**.
  - **Transition**. The primary goal of this phase is to roll out the fully functional system to end-users. The milestone associated with this phase is **Product Release (MS4)**.

# Two Dimensions of the RUP Framework

19

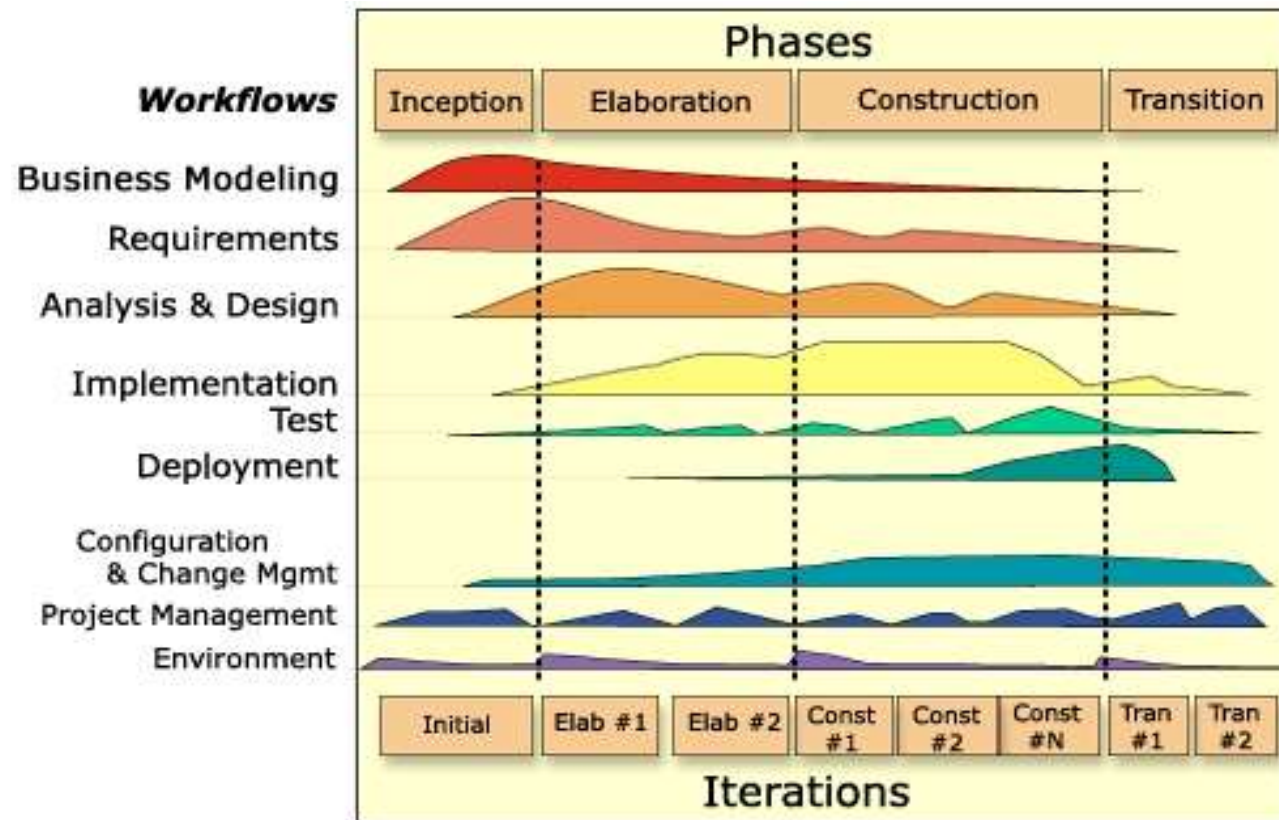
- In RUP, workflows cut across the set of four phases.
- Each workflow is a set of activities that project members perform.



# RUP Workflows and Iterations

20

- In RUP, each phase is divided into and repeats across multiple iterations.
- An iteration is a mini-project that is part of the phase.
- Different phases have different focuses across related workflows.



# Agile Development Explained

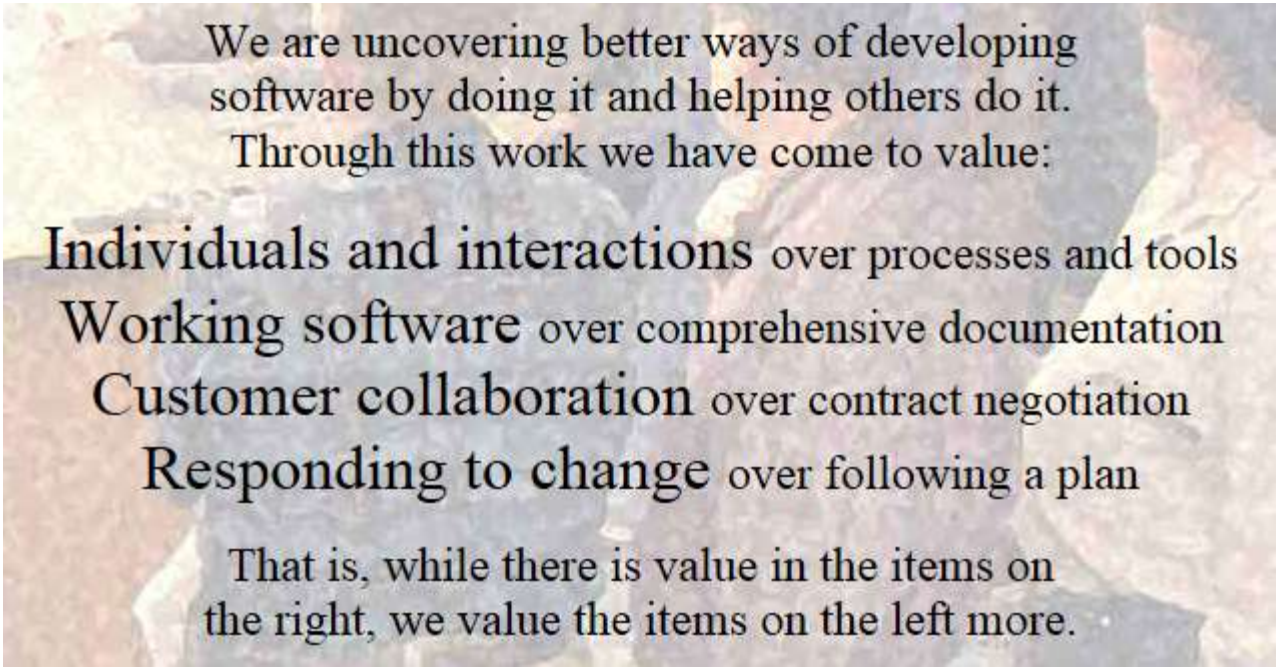
21

- **Agile Software Development** is a set of software development methods in which requirements and solutions evolve through collaboration between self-organizing, cross-functional teams. It promotes adaptive planning, evolutionary development, early delivery, continuous improvement, and encourages rapid and flexible response to change.
- A collection of *lightweight* software development methods evolved in the mid-1990s in response to the perceived *heavyweight* waterfall-oriented methods.
- Although these methods originated before the publication of the Agile Manifesto in 2001, they are now collectively referred to as Agile development.

# Agile Manifesto

22

- In February 2001, seventeen software developers met at the Snowbird Resort in Utah to discuss lightweight development methods.
- They published the *Manifesto for Agile Software Development*:



We are uncovering better ways of developing software by doing it and helping others do it.  
Through this work we have come to value:

**Individuals and interactions** over processes and tools  
**Working software** over comprehensive documentation  
**Customer collaboration** over contract negotiation  
**Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

# Waterfall Model vs. Iterative Development

23

- A common way to deal with a complex task, e.g., developing a large system, is to break it into smaller and easier to manage parts.
- The main difference between the Waterfall and Iterative development models is how we break up a project into smaller chunks.
- Regardless of which model is used, a project team performs the same four types of activities – **Specification, Design, Validation, Evolution**.

## Waterfall Model

- Breaks down a project based on activity:
- requirements, design, coding, testing

## Iterative Model

- Breaks down a project by subset of functionality.
- Each iteration goes through the same activities.

# Software Reuse

24

- Software reuse is a recognized solution to the three primary drivers of the software business – *faster, better, cheaper*.
- This topic has been the focus of the IT industry for several decades.
- **Software Product Lines (SPL)** is the software discipline that focuses on software reuse.
- SPL practices go beyond product creation into maintenance and evolution, lowering the overall complexity of product line development, increasing the scalability of product line portfolios.



# Software Product Lines Terminology

25

## Product Family

A product family is a set of related products, which we call a ***product line***.

## Software Product Line

A set of software systems sharing a common, managed set of features that represent specific needs of a particular market segment; these shared features are developed from a common set of core assets [software reuse].

## Core Assets

Core assets form the basis for the software product line and include the architecture, reusable components, domain models, requirements specifications, etc.

## Domain Model

A domain model is a specialized body of knowledge. An area of expertise, or a collection of related functionality.

# Product Family Examples

26

## Keurig Product Line - Coffee Maker Models



## Nissan Product Line - Car Models



# SPL Essential Activities

27

The SPL engineering discipline is based on three essential activities:

- Architecture
- Components
- Production Plan



- Product Line Scope
- Core Assets
- Production Plan

Oversees the core asset development and the product development activities.

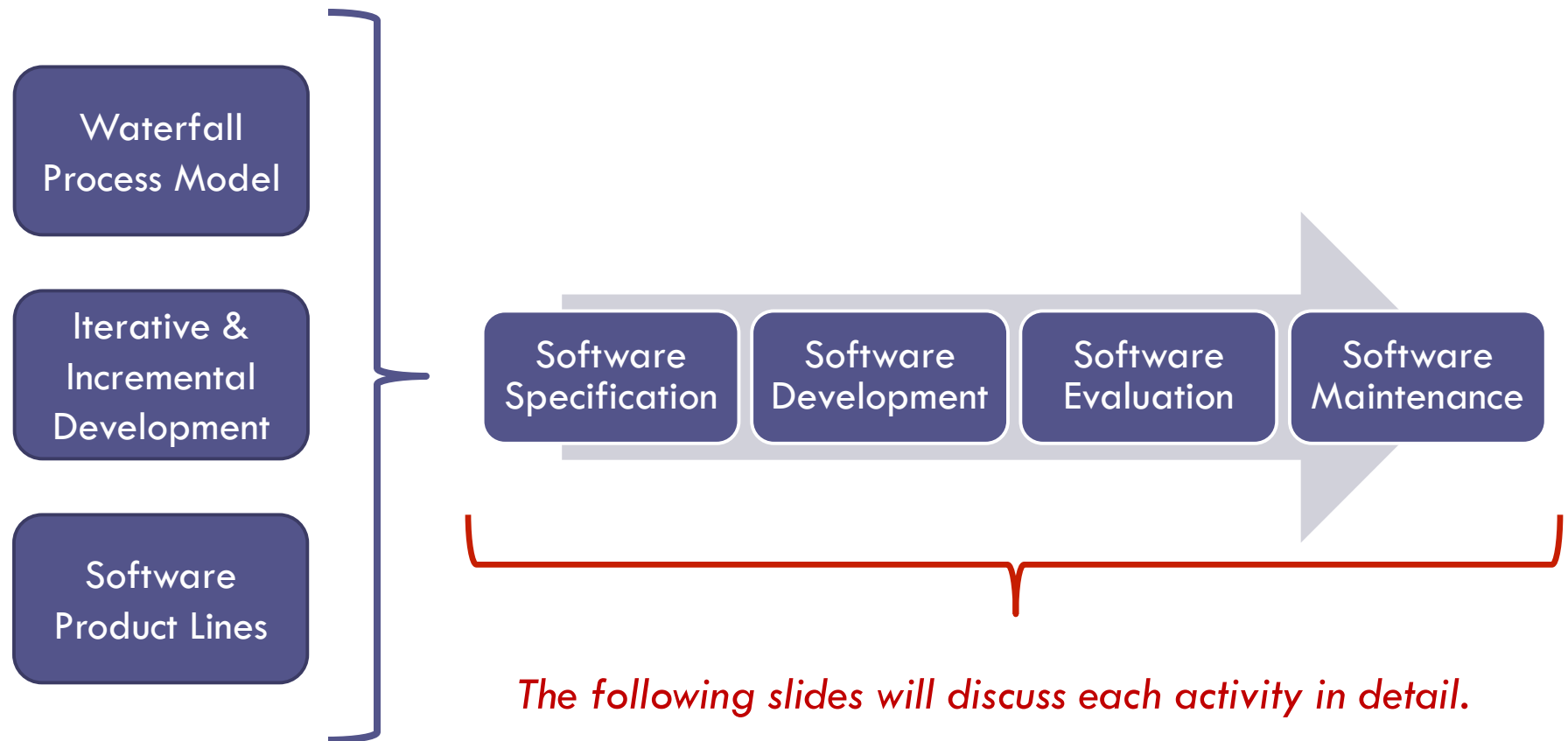
# SPL Essential Activities Explained

28

Activity	Description
Core Asset Development	The goal of core asset development is to establish a production capability for products. This activity is iterative. It includes developing an architecture and components, shared by the products in the family, as well as the production plan describing how the products are produced from the core assets.
Product Development	The product development activity depends on the three outputs from the core asset development – product line scope, core assets, and the production plan.
Management	Management is a critical part of the SPL process. Activities must be provided resources, and they must be coordinated and supervised. Technical management oversees the core asset development and the product development activities and ensures that both groups follow the process defined for the product line.

# Fundamental Activities Common to All Processes

29



# Process Activities – Software Specification

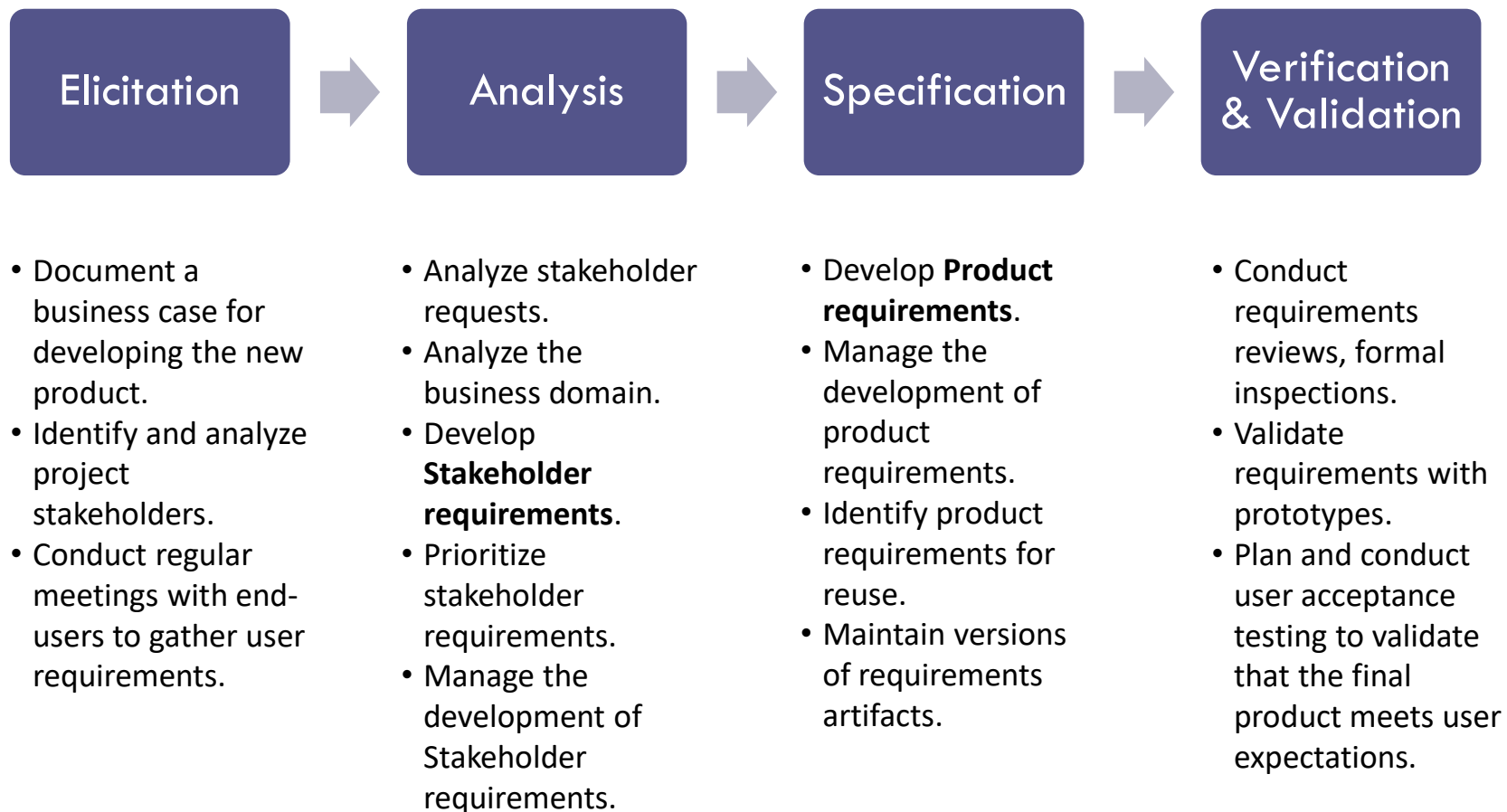
30

- A **software specification** activity is the process of establishing what services are required and the constraints on the system's operation and development.
- The Requirements Engineering process includes:
  - Feasibility study
    - Determines whether it is technically and financially feasible to build the system.
  - Requirements elicitation and analysis
    - Determines what the system stakeholders require or expect from the system.
    - Determines what kind of requirements are needed.
  - Requirements specification
    - Defines the requirements in detail.
  - Requirements validation
    - Checks the validity of the requirements.

# Requirements Engineering (RE) Process

31

The RE process includes four conventional phases:



# Process Activities – Software Development

32

**Software Development** is the process of converting the system specification into an executable system, it includes:

- **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.



# Process Activities – Software Design

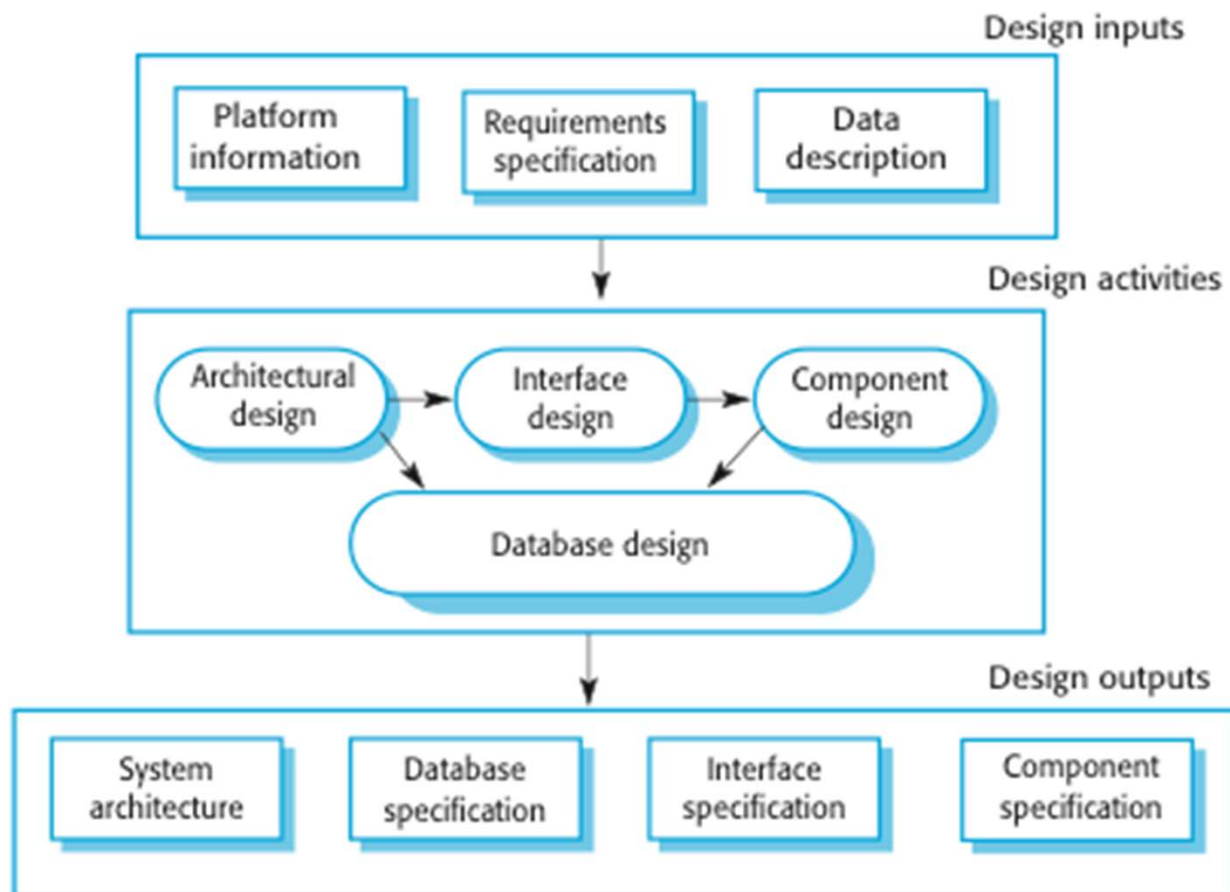
33

Software Design activities can have different objectives, most common of them include the following:

- *Architectural design*, where you identify the overall structure of the system, the principal components (sometimes called sub-systems or modules), their relationships and how they are distributed.
- *Database design*, where you design the system data structure and how it will be represented in a database.
- *Interface design*, where you define the interfaces between system components.
- *Component design*, where you take each system component and design how it will operate.

# Process Activities – Software Design

34



# Process Activities – Software Validation

35

**Software validation** , more commonly called **software testing**, is intended to show that a) system conforms to its requirements, and b) meets the expectations of end-users.

During a project cycle, a typical system undergoes various types of testing:

- Unit and component integration testing, performed by developers,
- System testing, performed by testers,
- End-to-End testing with other systems,
- User acceptance testing, performed by users or business analysts,
- Performance testing , performed by professional performance engineers,
- Operational Acceptance Testing, performed by IT operations.

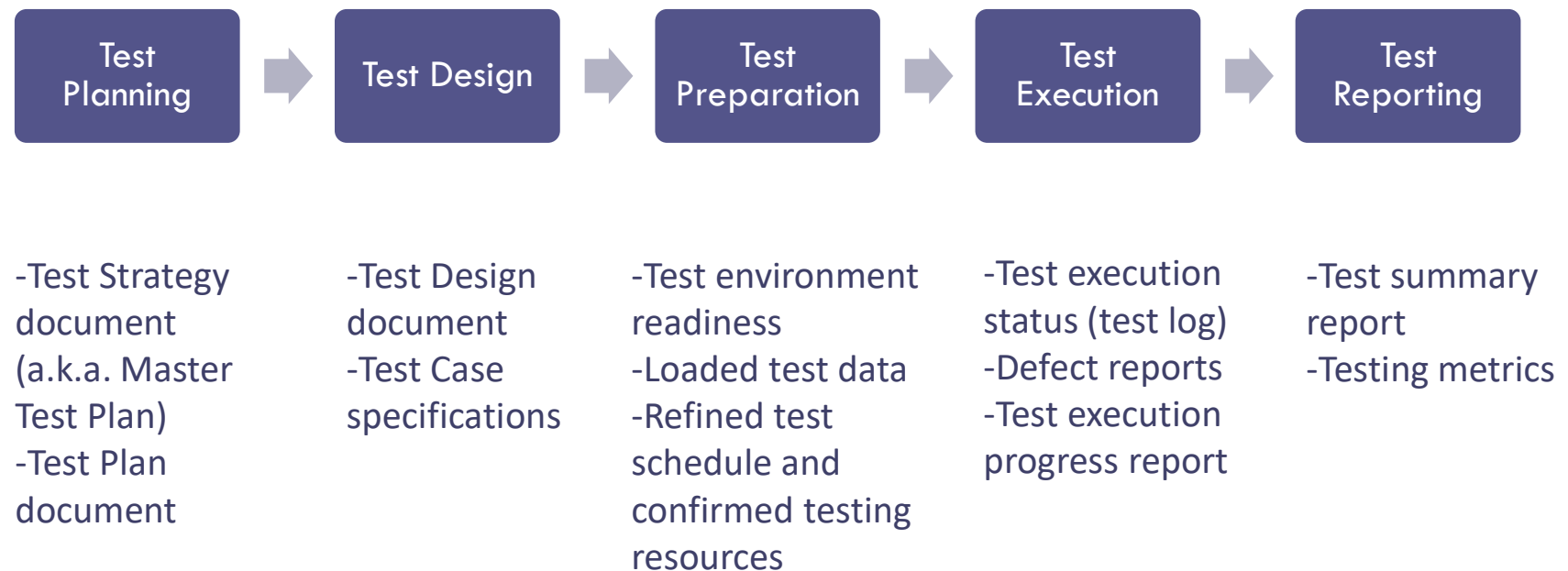
See  
Slide 13

Commonly, separate types of tests are performed in separate environments.

# Conventional Test Process Lifecycle

36

Regardless of the testing type (a.k.a. test level), the test process is planned and executed as the following phases:



# Process Activities – Software Evolution

37

- Software evolution, a.k.a. **software maintenance** is an important part of a software lifecycle.
- Most applications in IT departments are existing systems that evolve driven by continuous business condition changes.
- A new application might be developed within 6-9 months. Once in production, it can then evolve and be used for decades.
- A critical activity of software maintenance is performing *change impact analysis (CIA)*.

## Change Impact Analysis

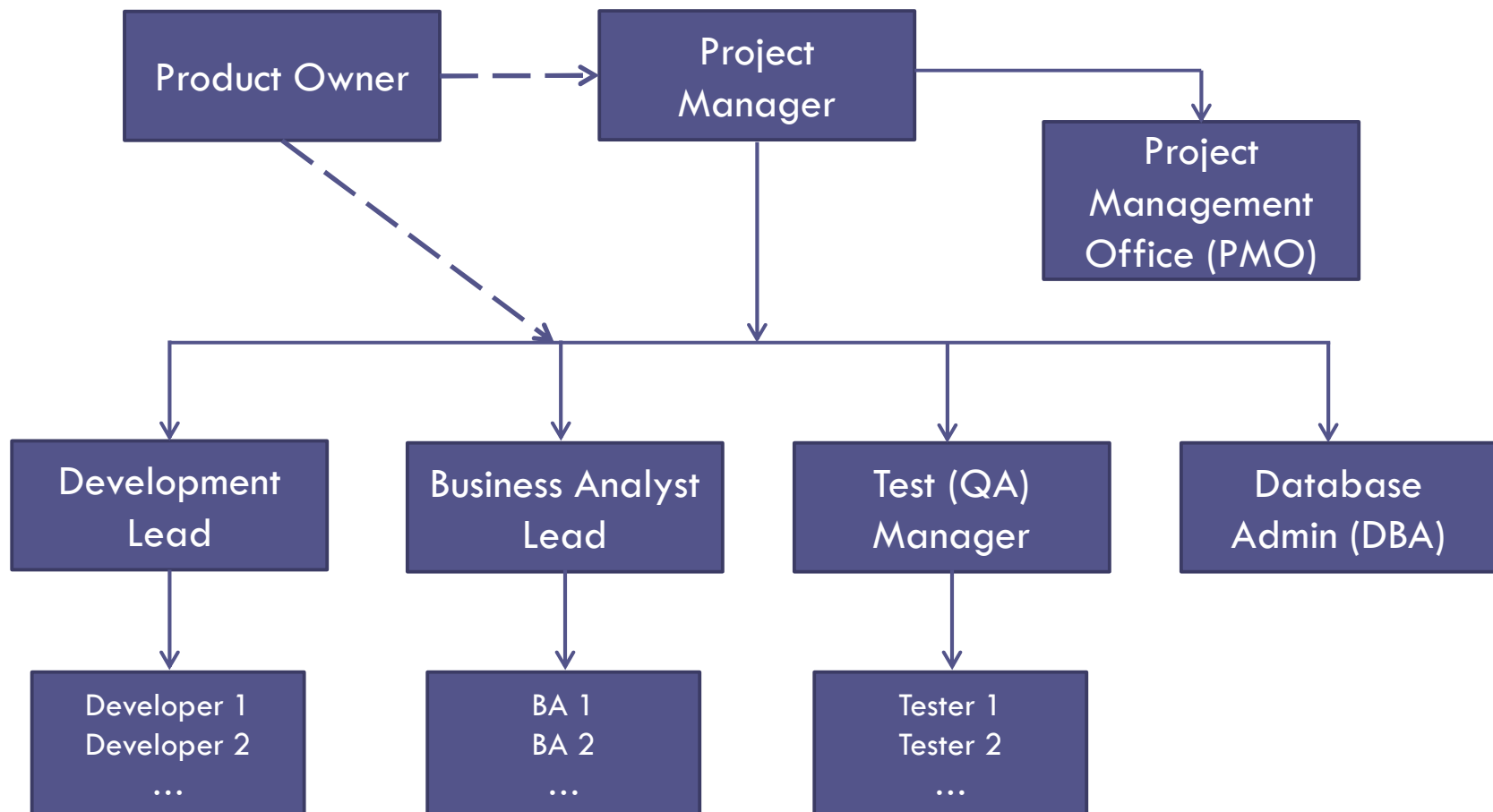
“CIA is the process of identifying the potential consequences of a change, or estimating what needs to be modified to accomplish a change.”

# Who Performs Process Activities?

## Large Project Organization Chart

38

- A project team can include different groups of software professionals with different skills.
- Different project team members are responsible for different tasks and deliverables.



# RACI Table Example

39

- **RACI** means:

- **(R)**esponsible,
- **(A)**ccountable,
- **(C)**onsulted,
- **(I)**nformed.

- RACI table is a formal representation of project team or business stakeholders' roles and responsibilities.

Process Area	Project Tasks	Project Roles								
		Project Manager	PMO	Dev Lead	Developer	BA Lead	Business Analyst	Test Manager	Tester	DBA
Project Management	Develop a project plan	A,R	C	C	I	C	I	C	I	I
	Provide cost estimate	A,R	R	I	I	I	I	I	I	I
	Hire resources	A,R	C	R	C	R	C	R	C	C
	Establish a project portal on SharePoint	A	R	C	I	C	I	C	I	I
	Maintain a project risk and issue log	A	R	C	C	C	C	C	C	C
	Provide project status reports	A	R	C	I	C	I	C	I	I
Requirements	Perform requirements analysis	A	I	C	I	R	I	I	I	I
	Gather business requirements	A	I	C	I	R	C	I	I	I
	Produce functional requirements	A	I	C	C	R	R	C	I	I
Design	Produce high-level design specs	A	I	R	C	C	C	I	I	C
	Produce data model	A	I	C	C	C	I	I	I	R
	Produce detailed design specs	A	I	R	R	C	C	I	I	R
Coding	Establish a code repository	A	I	R	R	I	I	I	I	I
	Develop component code	A	I	R	R	I	I	I	I	I
Testing	Develop a test plan	A	I	C	I	C	C	R	C	C
	Establish a test repository	A	I	C	I	I	I	R	C	I
	Develop test specifications	A	I	I	I	I	I	R	R	I
	Execute testing, report defects	A	I	I	I	I	I	R	R	I
	Conduct defect review calls	A	I	C	I	C	I	R	C	C
	Produce, deliver defect metrics	A	C	C	I	C	I	R	C	I
	Support test environments	A	I	R	R	I	I	C	I	R
Deployment	Produce a deployment plan	A	I	R	R	I	I	I	I	R
	Produce deployment procedures	A	I	R	R	I	I	I	I	R
	Deploy software into production	A	I	R	R	I	I	C	C	R

# Software Process Improvement

40

- To the extent of the software project criticality, the project management should not only be concerned with selecting the right software development process, but with continuously improving the selected software process as well.
- In the 1930s, Walter Shewhart began work in process improvement with his principles of statistical quality control. These principles were refined by Phillip Crosby [1979], W. Edwards Deming in [1986], and Joseph Juran [1988] and became known as **Total Quality Management (TQM)**.
- TQM consists of organization-wide efforts to install and make permanent a climate in which an organization continuously improves its ability to deliver high-quality products and services to customers.
- Common process improvement frameworks include ISO 9000, Lean Manufacturing, Six Sigma, and CMMI.



# PDCA Method

41

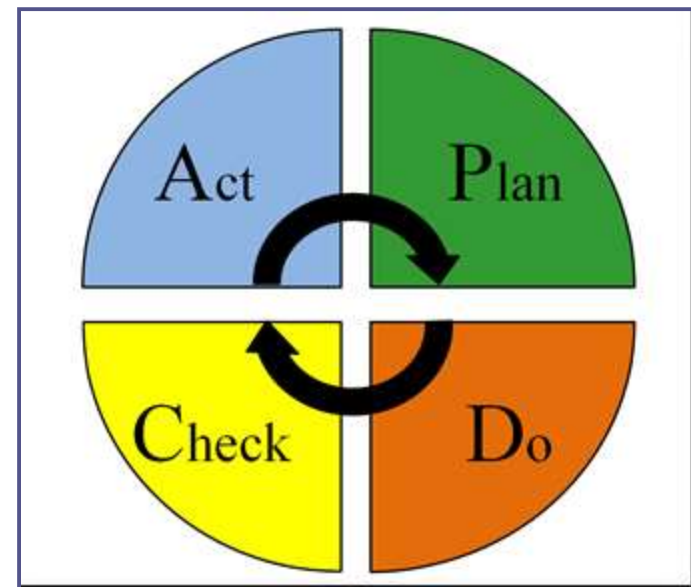
- **PDCA (plan–do–check–act)** is an iterative four-step management method used in business for the control and continuous improvement of processes and products.
- PDCA was made popular by Dr W. Edwards Deming, who is considered by many to be the father of modern quality control and process improvement.

**PLAN.** Establish the objectives and processes necessary to deliver results in accordance with the expected output (the target or goals). Make assumptions and start on a small scale to test improvement ideas.

**DO.** Implement the plan, execute the new process. Collect data for performing analysis in the following "CHECK" and "ACT" steps.

**CHECK.** Study the actual results (measured and collected in "DO" above) and compare against the expected results (targets or goals from the "PLAN") to ascertain any differences.

**ACT.** If the CHECK shows that the PLAN that was implemented in DO is an improvement to the prior standard (baseline), then that becomes the new standard (baseline) for how the organization should ACT going forward (new standards are enACTed).



# CMMI – the Process Maturity Framework

42

- The **Capability Maturity Model Integration (CMMI)**, is a process improvement framework that provides a clear definition of what an organization should do to promote behaviors that lead to improved process performance.
- The CMMI was developed by the Software Engineering Institute at Carnegie Mellon University (CMU) with representation from the defense industry, government, and academia, and is now operated and maintained by the CMMI Institute, an operating unit of CMU.
- CMMI is comprised of a set of **Process Areas** and provides the ability to assess the process maturity at five levels.

## Maturity Level

Degree of process improvement across a predefined set of process areas in which all goals in the set are attained. (See also “capability level” and “process area.”)

## Process Area

A cluster of related practices in an area that, when implemented collectively, satisfies a set of goals considered important for making improvement in that area.

# CMMI: 5 Levels of Process Maturity

43

- **Level 1 “Initial”**. Processes are usually ad hoc and chaotic. The organization usually does not provide a stable environment to support processes. Success in these organizations depends on the competence and heroics of the people in the organization and not on the use of proven processes.
- **Level 2 “Managed”**. The projects have ensured that processes are planned and executed in accordance with policy; the projects employ skilled people who have adequate resources to produce controlled outputs; involve relevant stakeholders; are monitored, controlled, and reviewed.
- **Level 3 “Defined”**. Processes are well characterized and understood, and are described in standards, procedures, tools, and methods. The organization’s set of standard processes, which is the basis for maturity level 3, is established and improved over time.
- **Level 4 “Quantitatively Managed”**. The organization and projects establish quantitative objectives for quality and process performance and use them as criteria in managing projects. Quantitative objectives are based on the needs of the customer, end users, organization, and process implementers.
- **Level 5 “Optimizing”**. An organization continually improves its processes based on a quantitative understanding of its business objectives and performance needs.

# CMMI: Process Areas

44

- The CMMI defines an incremental path to improve the process maturity and suggests 22 subsets of process areas to focus at each maturity level.
- Marked by the asterisks ( ★ ) are the process areas directly related to your project assignment.



Name	Abbr.	ML
Configuration Management	CM	2
Measurement and Analysis	MA	2
Project Monitoring and Control	PMC	2
Project Planning	PP	2
Process and Product Quality Assurance	PPQA	2
Requirements Management	REQM	2
Supplier Agreement Management	SAM	2
Decision Analysis and Resolution	DAR	3
Integrated Project Management	IPM	3
Organizational Process Definition	OPD	3
Organizational Process Focus	OPF	3
Organizational Training	OT	3
Product Integration	PI	3
Requirements Development	RD	3
Risk Management	RSKM	3
Technical Solution	TS	3
Validation	VAL	3
Verification	VER	3
Organizational Process Performance	OPP	4
Quantitative Project Management	QPM	4
Causal Analysis and Resolution	CAR	5
Organizational Performance Management	OPM	5

# Key Points

45

- Software processes are conceptual models that represent the activities involved in producing and maintaining a software system.
- Most common process models include the waterfall model, incremental development and reusable product line development.
- Requirements Engineering is the process of developing software specifications.
- Design and implementation processes are concerned with transforming requirements into an executable software system.
- Software validation and testing is the process of checking that the system conforms to its requirements and the end-user needs.
- Software evolution and maintenance takes place to cope with continuously evolving business needs.
- Process improvement focuses on improving the existing software processes to address management and end-user concerns, e.g., cost of development, product quality, time-to-market, etc.

# Exercises

46

1. What are the four fundamental software engineering activities?
2. What are the three most common software process models?
3. What are the conventional stages of the waterfall model?
4. Explain what the V-model is.
5. What are the three main characteristics of the RUP model?
6. Explain why incremental development is the most effective process model for developing business applications.
7. Explain the main difference between the Waterfall and Iterative development models.
8. What are the phases of the conventional requirements engineering process?
9. What are the phases of the conventional software testing process?
10. What are the common roles on a software development project?
11. What is a RACI table? How is it used?
12. Explain, what the PDCA method is.