Chapter 2: Software Processes

Sabrinah Yonell C. Yap

TOPICS COVERED

- Software process models
- Process activities
- Coping with change
- The Rational Unified Process

THE SOFTWARE PROCESS

- A structured set of activities required to develop a software system.
- Many different software processes but all involve:
  - **SPECIFICATION** – defining what the system should do
  - **DESIGN AND IMPLEMENTATION** – defining the organization of the system and implementing the system
  - **VALIDATION** – checking that it does what the customer wants
  - **EVOLUTION** – changing the system in response to changing customer needs.

SOFTWARE PROCESS DESCRIPTIONS

- Specifying a data model, designing a user interface and the ordering of these activities.
- Process descriptions may also include:
  - **PRODUCTS**- outcomes of a process activity;
  - **ROLES**- lists the responsibilities of the people involved in the process;
  - **PRE- AND POST- CONDITIONS** - statements that are true before and after a process activity has been enacted or a product produced.

PLAN-DRIVEN & AGILE PROCESSES

PLAN-DRIVEN PROCESS

- Processes are planned in advance
- Strictly follows the Gantt Chart

AGILE PROCESS

- Incremental planning
- Flexibility to follow changing customer's requirements

IN INDUSTRY PRACTICE

- Mixture of both plan-driven and agile process, *on how the shoe fits*
  - With this, there's **NO RIGHT OR WRONG PROCESS**

SOFTWARE PROCESS MODELS

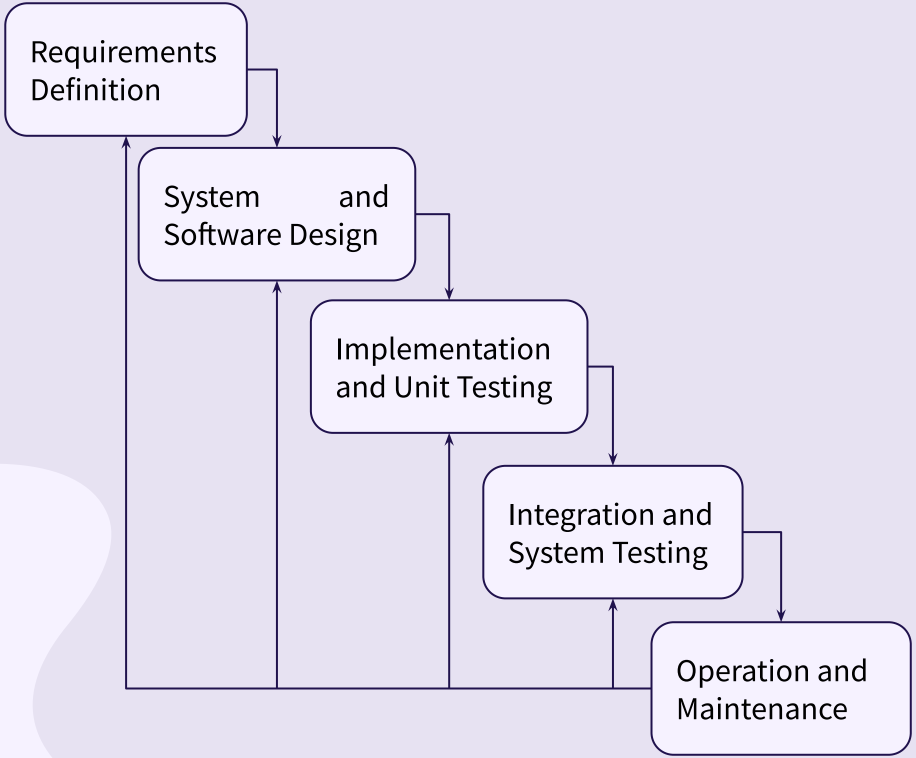
- Waterfall model
- Incremental development
- Reuse-oriented software engineering

THE WATERFALL MODEL

- A planned-driven model that has a separate and distinct phases of specification and development

PHASES

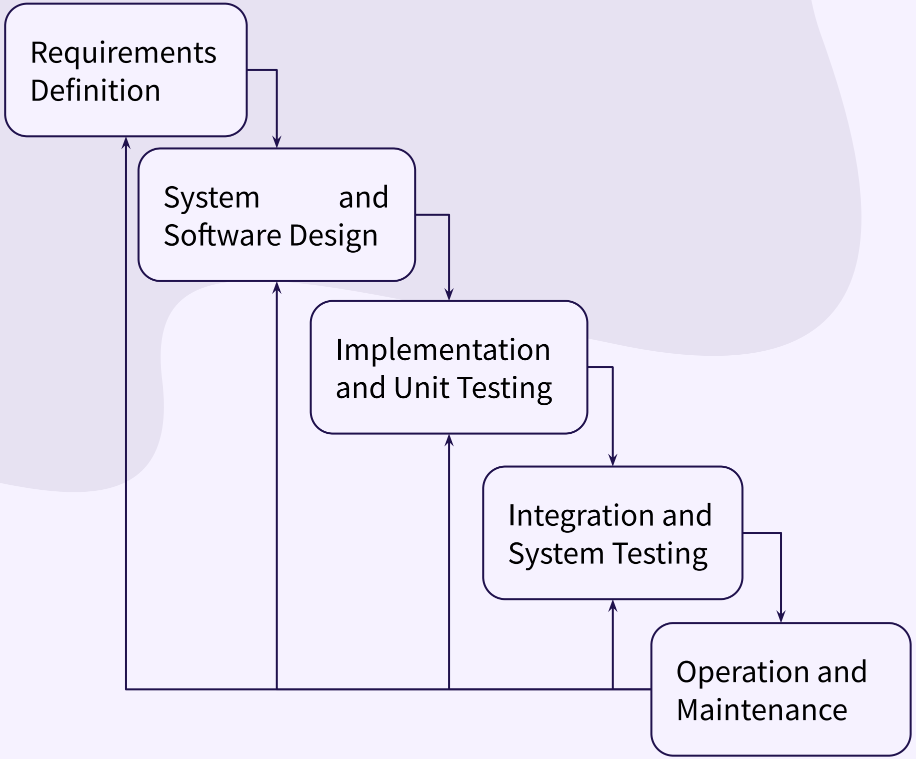
- Requirements analysis and definition
- System and software design
- Implementation and unit testing
- Integration and system testing
- Operation and maintenance


Requirements Definition
System and Software Design
Implementation and Unit Testing
Integration and System Testing
Operation and Maintenance

PROBLEMS OF THE WATERFALL MODEL

- **DIFFICULTY OF ACCOMMODATING CHANGES** after the process is underway.

- **NON-FLEXIBILITY OF PARTITIONS** thereby making it hard to respond to ever-changing customer requirements



Requirements Definition

System and Software Design

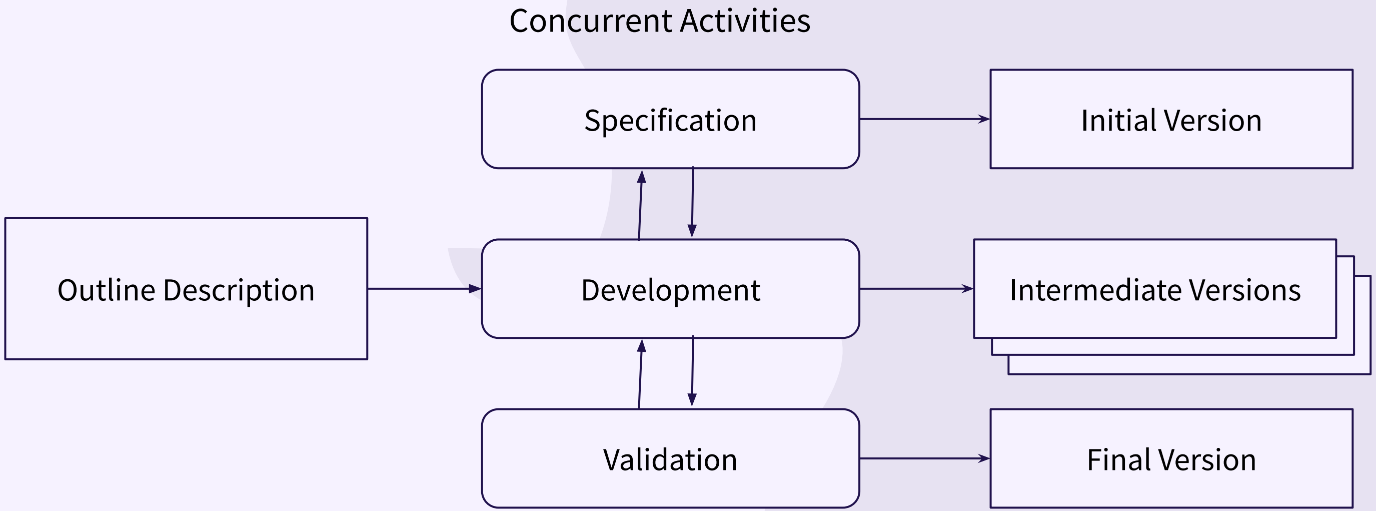
Implementation and Unit Testing

Integration and System Testing

Operation and Maintenance

THE INCREMENTAL MODEL

- A planned-driven or agile model that interleaves specification, development and validation



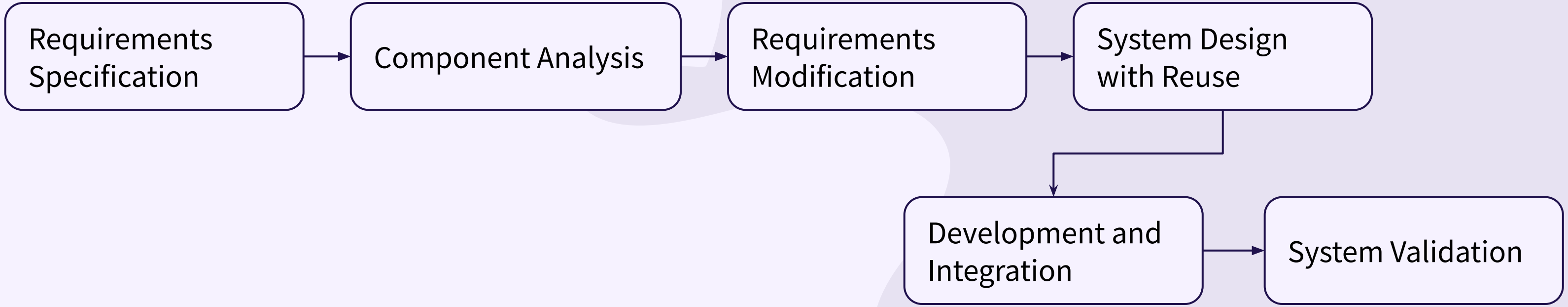
THE INCREMENTAL MODEL

PROS AND CONS

PROS	CONS
Reduced cost of changing customer requirements	The process is not visible.
Easier customer feedback on finished development work	System structure tends to degrade as new increments are added.
More rapid delivery and deployment	

REUSE-ORIENTED SOFTWARE ENGINEERING

- Based on systematic reuse where systems are integrated from existing components or COTS (Commercial-off-the-shelf) systems.


Requirements Specification
Component Analysis
Requirements Modification
System Design with Reuse
Development and Integration
System Validation

TYPES OF SOFTWARE COMPONENT

- Web services that are developed according to service standards and which are available for remote invocation.
- Collections of objects that are developed as a package to be integrated with a component framework such as .NET or J2EE.
- Stand-alone software systems (COTS) that are configured for use in a particular environment.

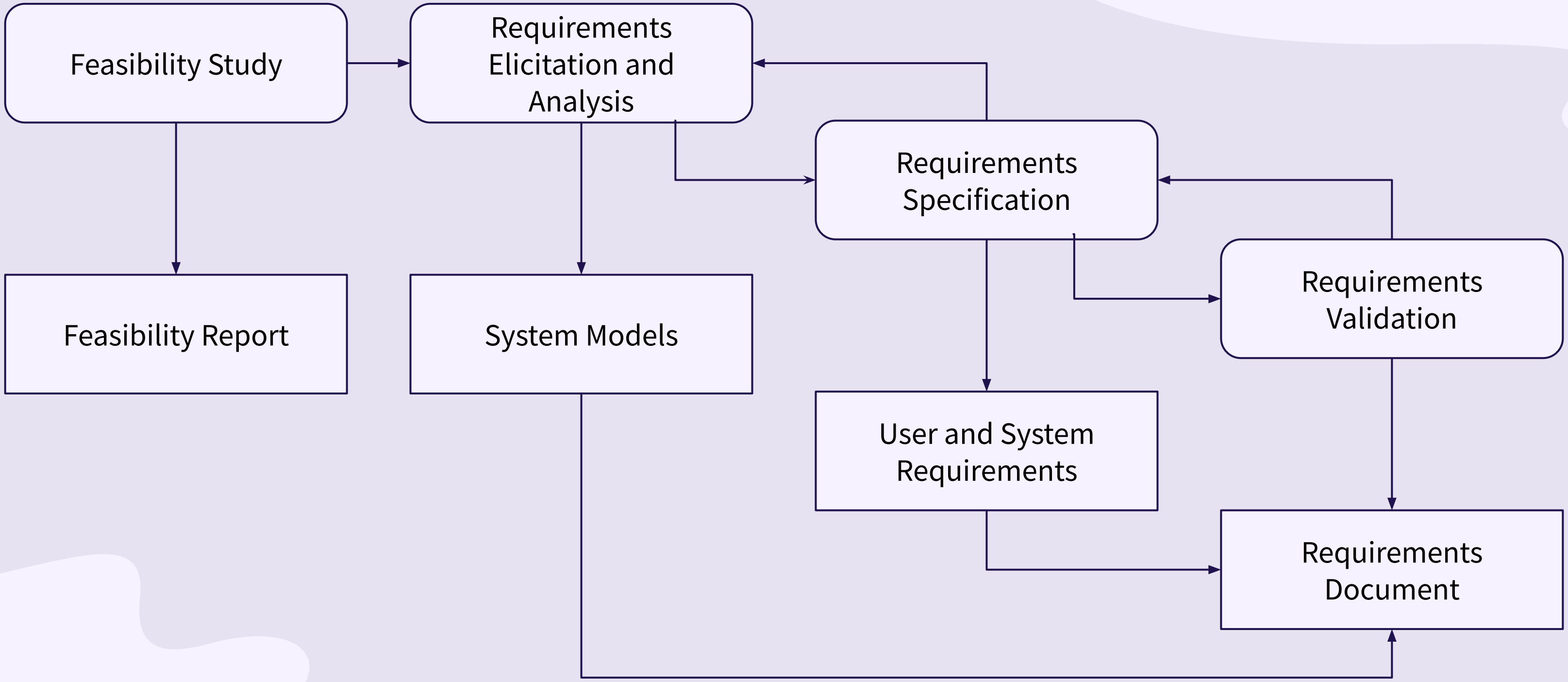
PROCESS ACTIVITIES

- 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 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 inter-leaved.

SOFTWARE SPECIFICATION

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

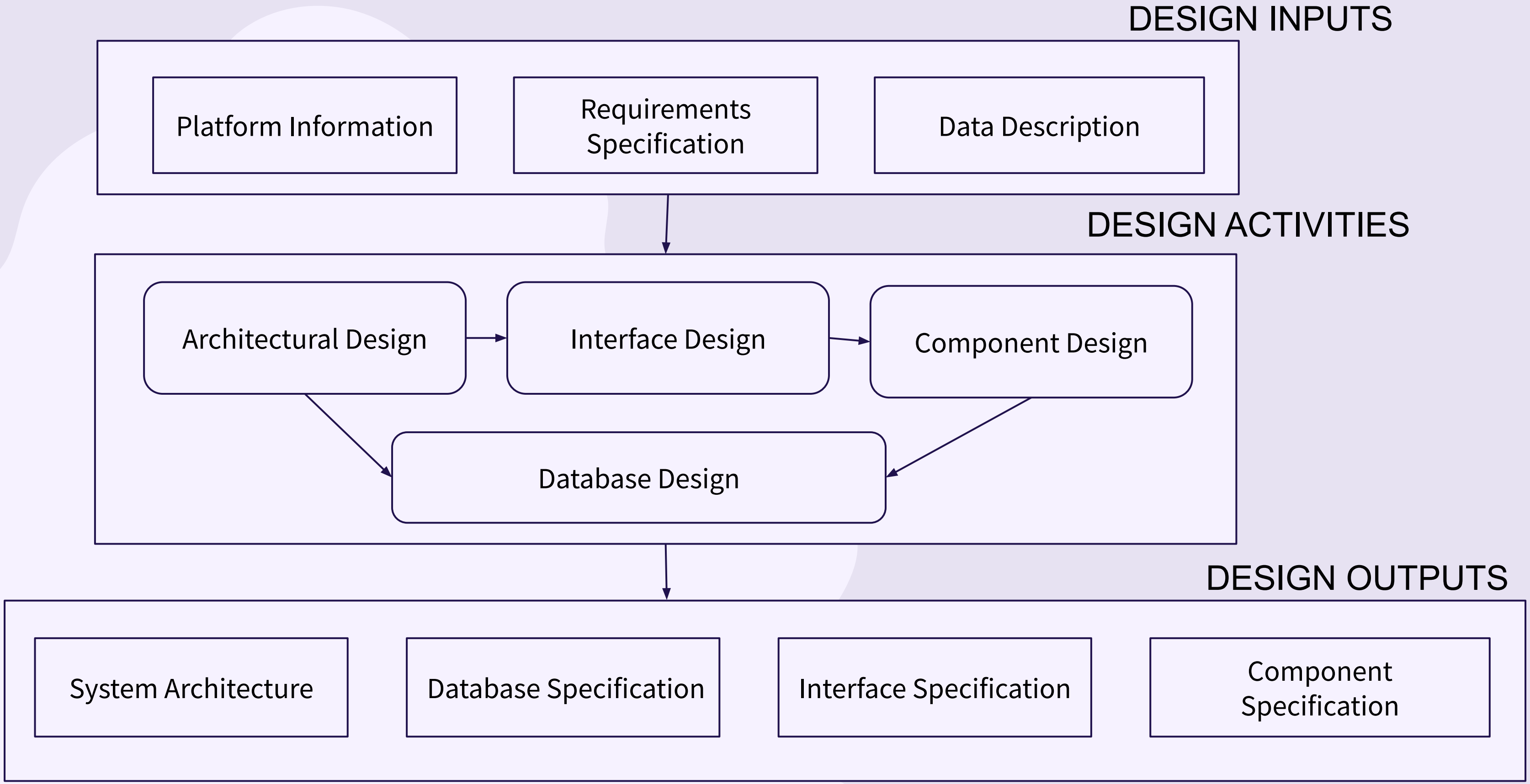
THE REQUIREMENTS ENGINEERING PROCESS



SOFTWARE DESIGN AND IMPLEMENTATION

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


DESIGN INPUTS

Platform Information

Requirements Specification

Data Description

DESIGN ACTIVITIES

Architectural Design

Interface Design

Component Design

Database Design

DESIGN OUTPUTS

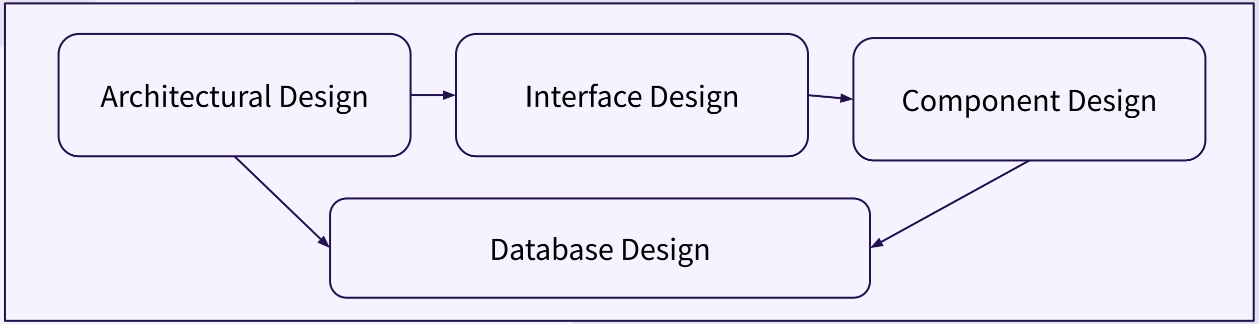
System Architecture

Database Specification

Interface Specification

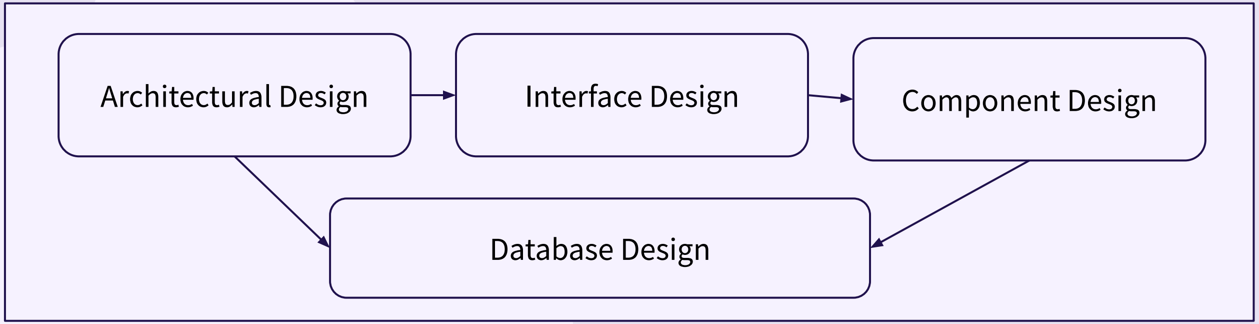
Component Specification

DESIGN ACTIVITIES



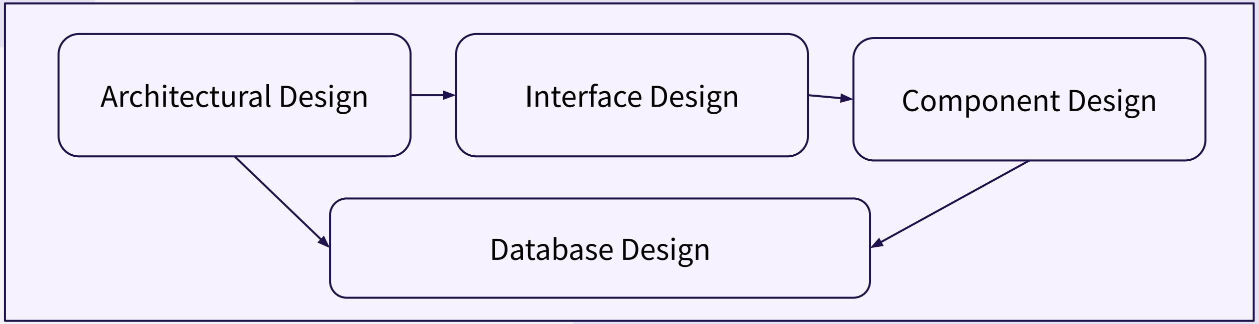
ARCHITECTURAL DESIGN- identification of the overall structure of the system, the principal components, their relationships and how they are distributed.

DESIGN ACTIVITIES



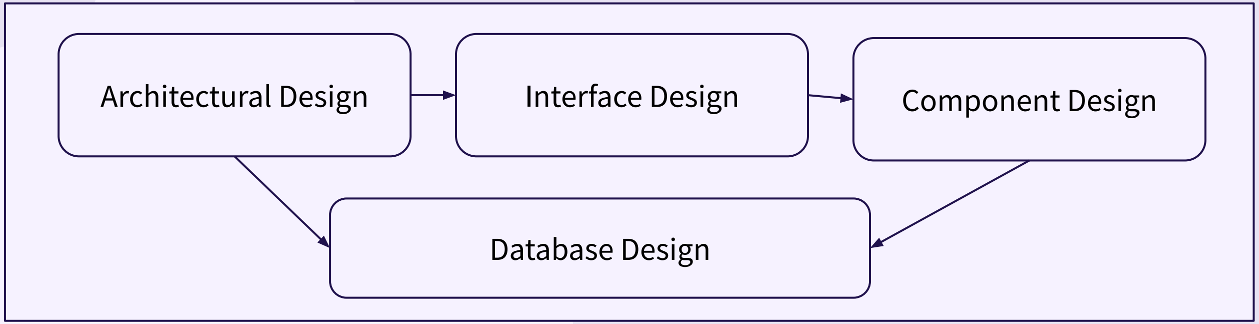
INTERFACE DESIGN- defining the interfaces between system components.

DESIGN ACTIVITIES



COMPONENT DESIGN- taking each system component and design how it will operate.

DESIGN ACTIVITIES

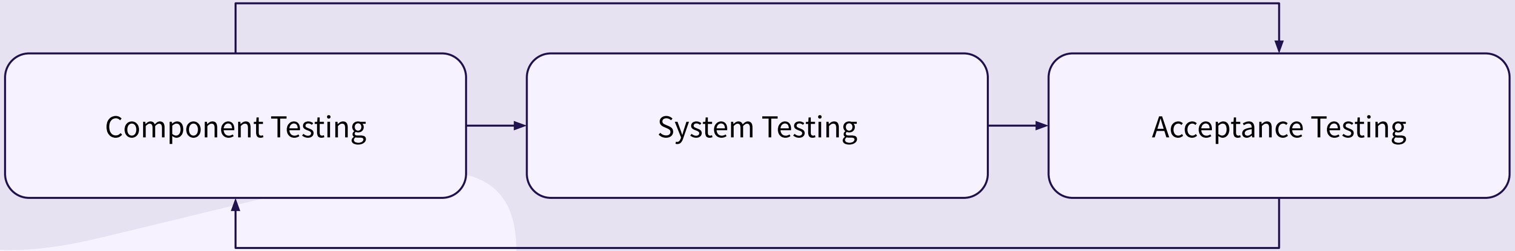


DATABASE DESIGN- designing the system data structures and how these are to be represented in a database.

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.

STAGES OF TESTING

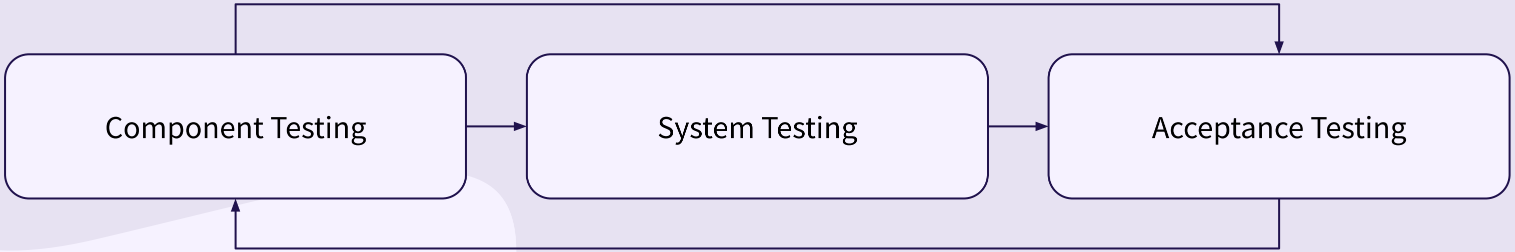


Component Testing

System Testing

Acceptance Testing

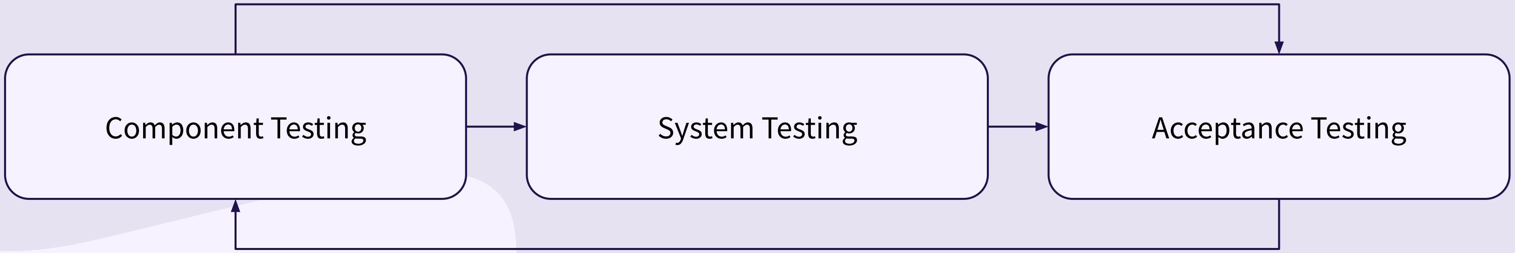
STAGES OF TESTING



DEVELOPMENT OR COMPONENT TESTING

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

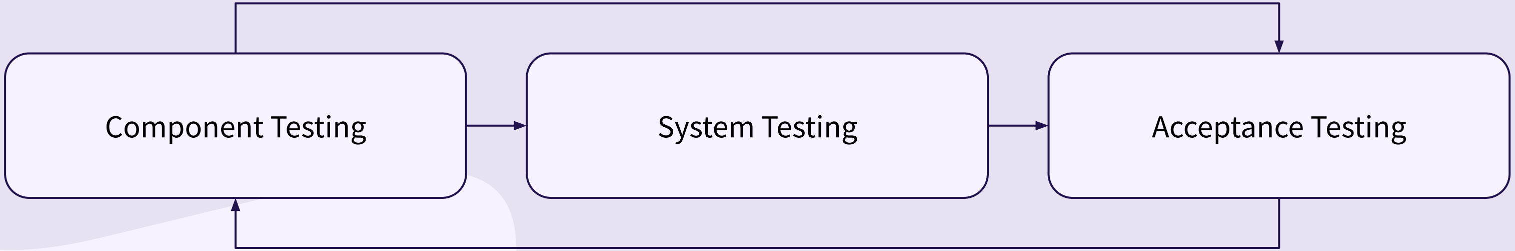
STAGES OF TESTING



SYSTEM TESTING

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

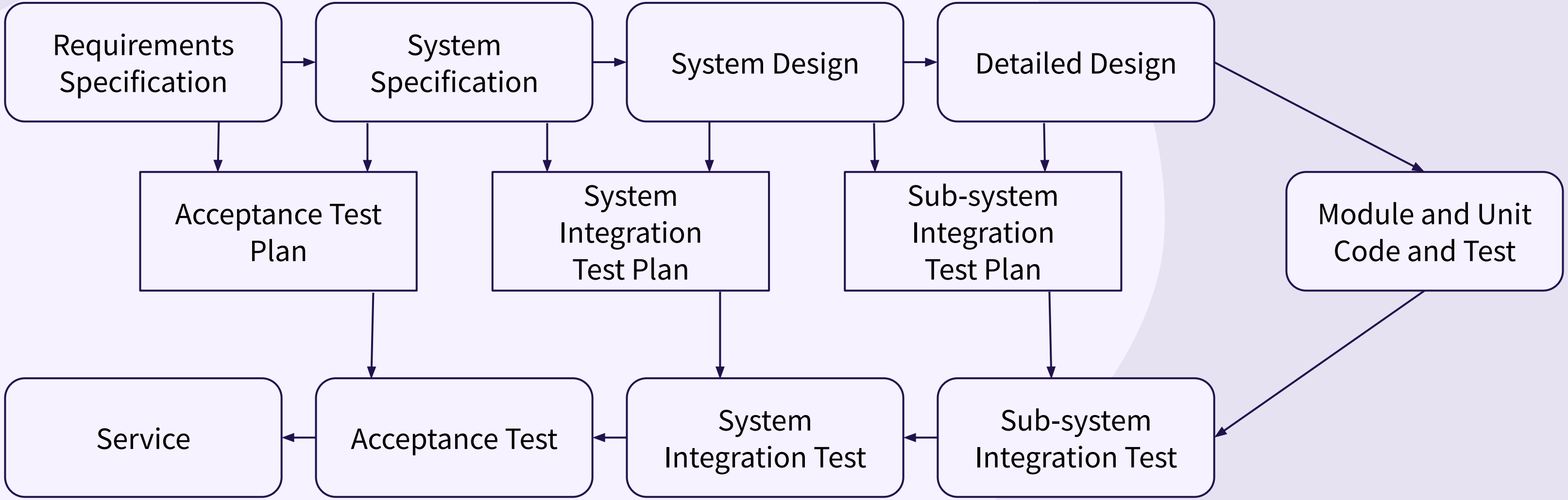
STAGES OF TESTING



ACCEPTANCE TESTING

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

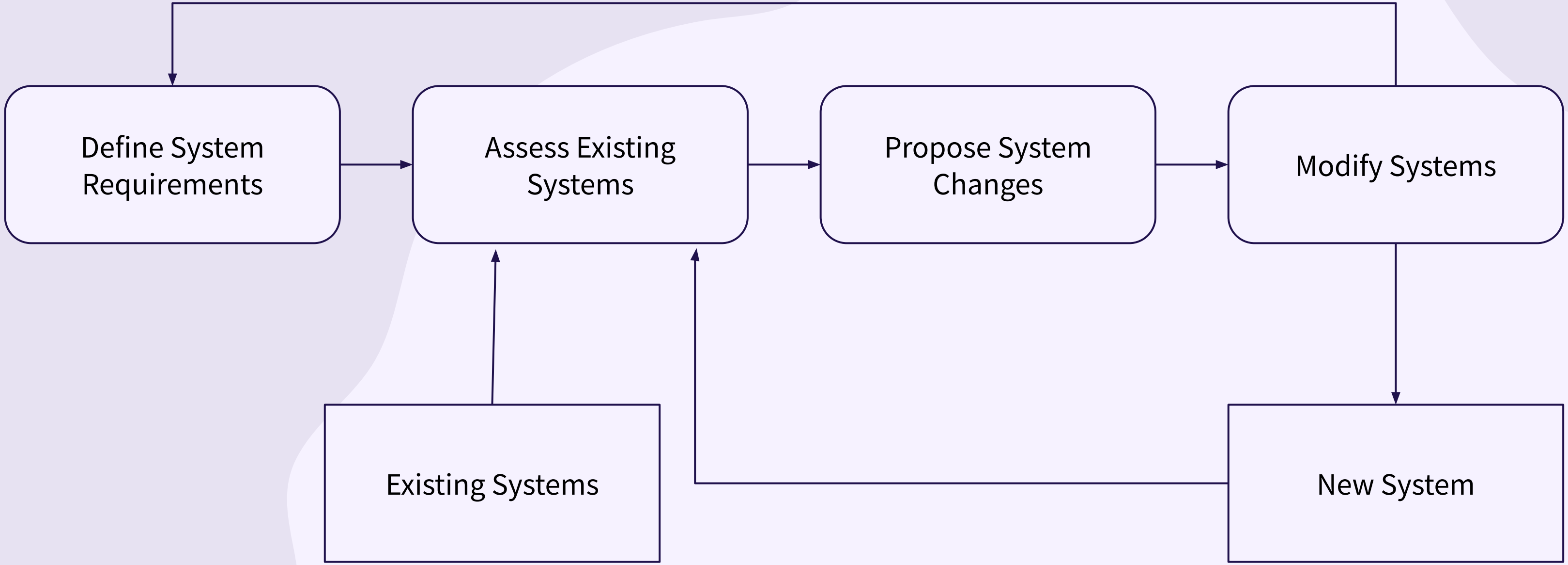
TESTING PHASES IN A PLAN-DRIVEN SOFTWARE PROCESS


Requirements Specification
System Specification
System Design
Detailed Design
Acceptance Test Plan
System Integration Test Plan
Sub-system Integration Test Plan
Module and Unit Code and Test
Service
Acceptance Test
System Integration Test
Sub-system Integration Test

STAGES OF TESTING

- 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.

SYSTEM EVOLUTION



Define System Requirements

Assess Existing Systems

Propose System Changes

Modify Systems

Existing Systems

New System

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-analysing requirements) as well as the costs of implementing new functionality

REDUCING THE COSTS OF REWORK

- Change avoidance, 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.

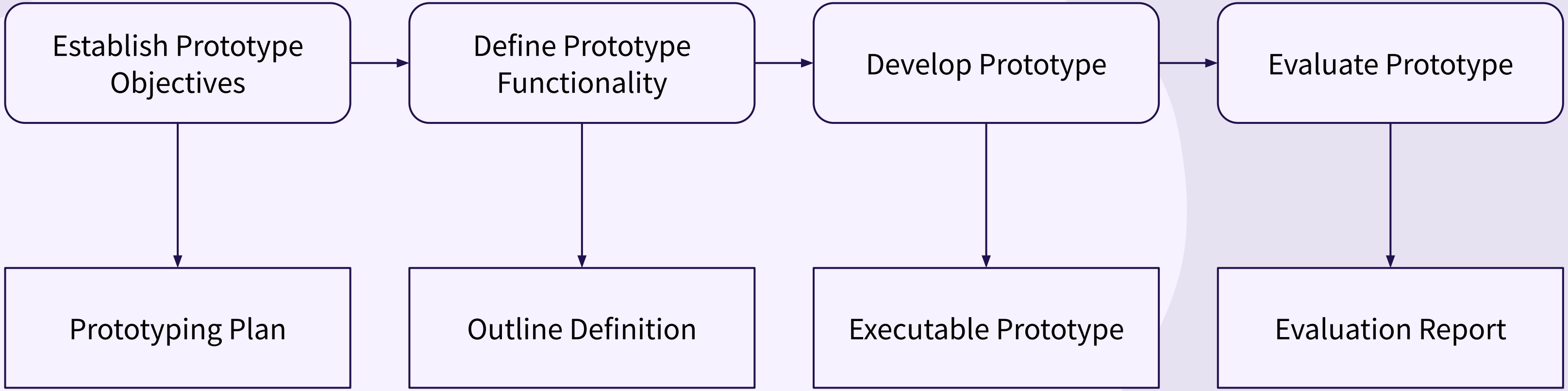
SOFTWARE PROTOTYPING

- 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.

BENEFITS OF PROTOTYPING

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

THE PROCESS OF PROTOTYPE DEVELOPMENT



Establish Prototype Objectives

Define Prototype Functionality

Develop Prototype

Evaluate Prototype

Prototyping Plan

Outline Definition

Executable Prototype

Evaluation Report

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

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;
    - Prototypes are normally undocumented;
    - The prototype structure is usually degraded through rapid change;
    - The prototype probably will not meet normal organisational quality standards.

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 prioritised 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.

INCREMENTAL DEVELOPMENT AND DELIVERY

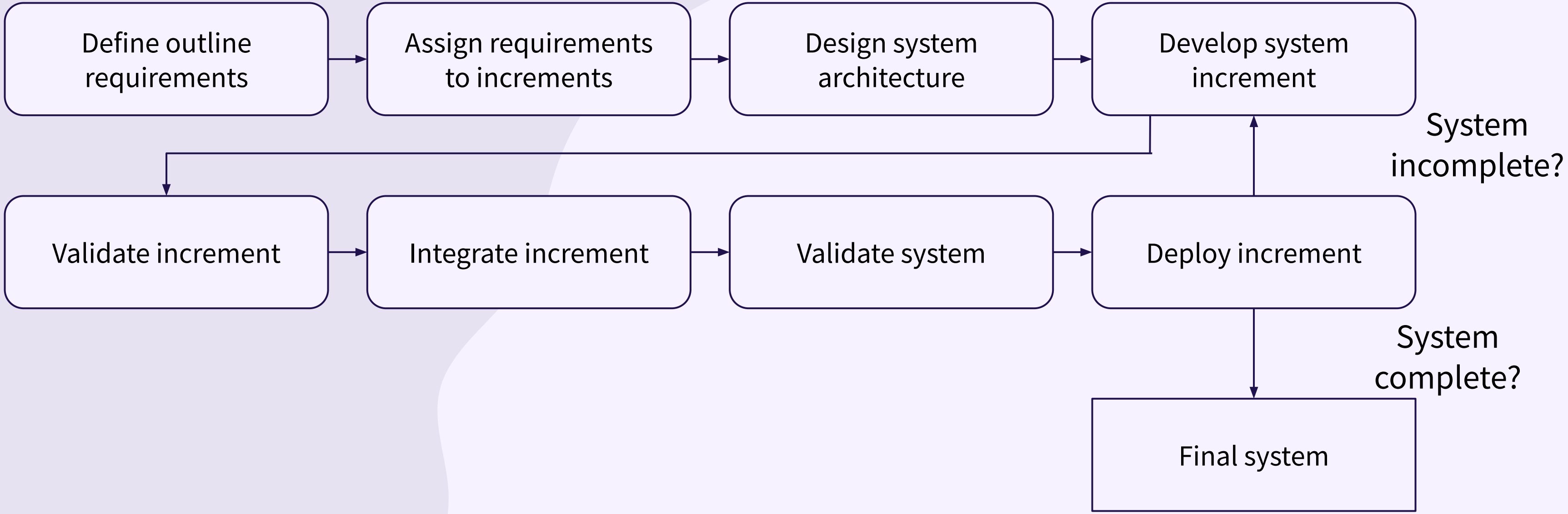
INCREMENTAL DEVELOPMENT

- ○ 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.

INCREMENTAL DELIVERY



Define outline requirements

Assign requirements to increments

Design system architecture

Develop system increment

Validate increment

Integrate increment

Validate system

Deploy increment

System incomplete?

System complete?

Final system

INCREMENTAL DELIVERY

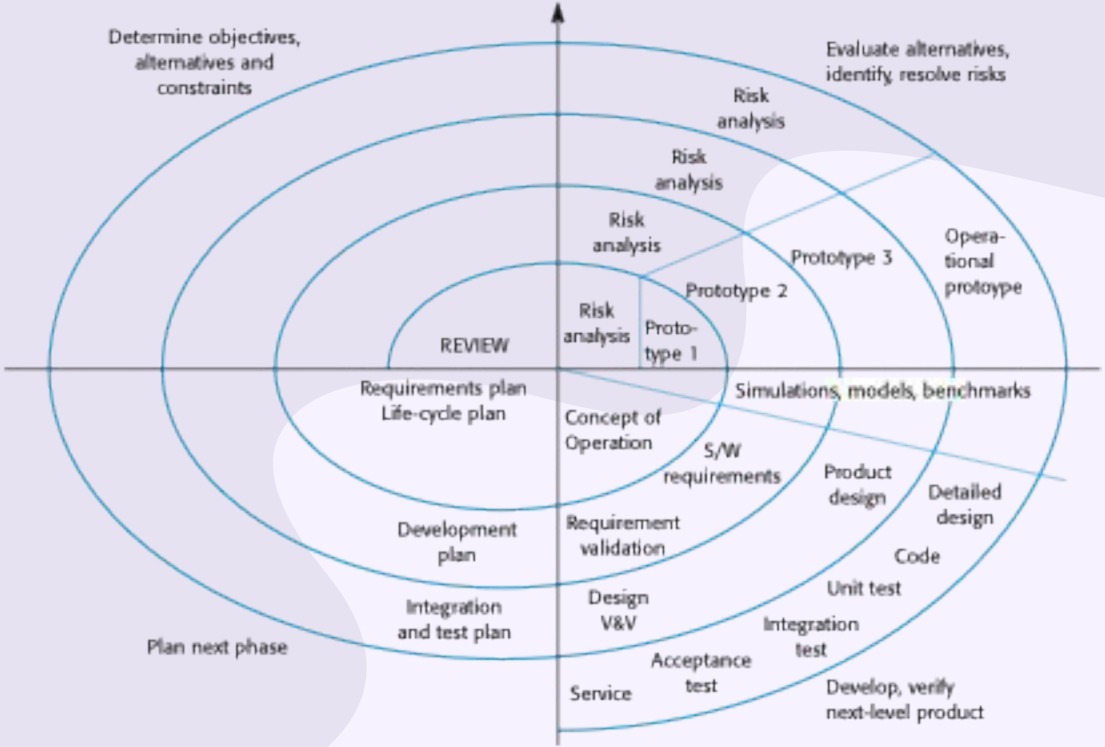
PROS AND CONS

PROS	CONS
Customer value can be delivered with each increment so system functionality is available earlier.	Most systems require a set of basic facilities that are used by different parts of the system.
Early increments act as a prototype to help elicit requirements for later increments.	The essence of iterative processes is that the specification is developed in conjunction with the software.
Lower risk of overall project failure.	
The highest priority system services tend to receive the most testing.	

BOEHM’S SPIRAL MODEL

- Process is represented as a spiral rather than as a sequence of activities with backtracking.
- Each loop in the spiral represents a phase in the process.
- No fixed phases such as specification or design - loops in the spiral are chosen depending on what is required.
- Risks are explicitly assessed and resolved throughout the process.

BOEHM'S SPIRAL MODEL OF THE SOFTWARE PROCESS



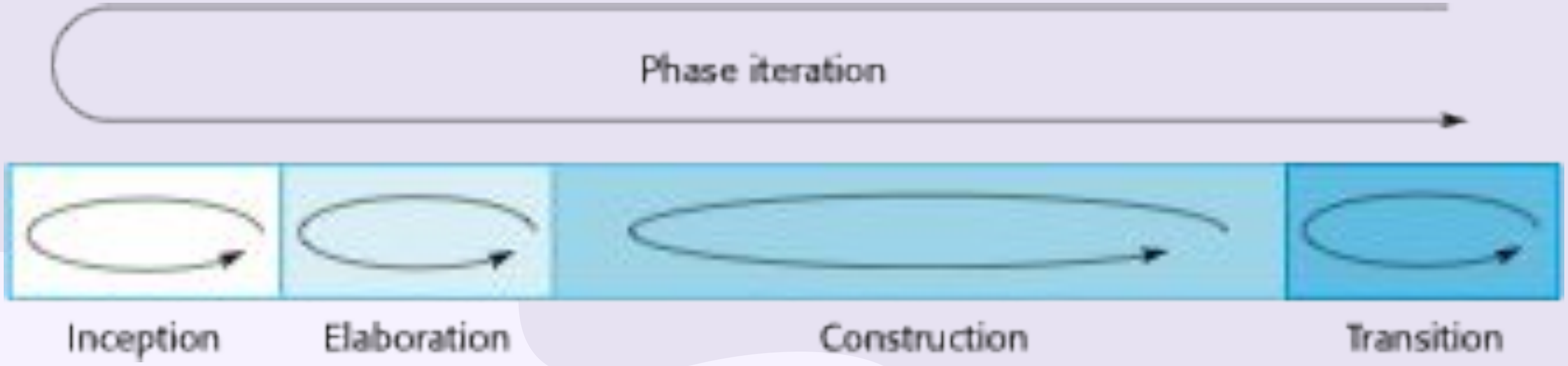
SPIRAL MODEL SECTORS

- **OBJECTIVE SETTING**- specific objectives for the phase are identified.
- **RISK ASSESSMENT AND REDUCTION**- risks are assessed and activities put in place to reduce the key risks.
- **DEVELOPMENT AND VALIDATION**- a development model for the system is chosen which can be any of the generic models.
- **PLANNING**- the project is reviewed and the next phase of the spiral is planned.

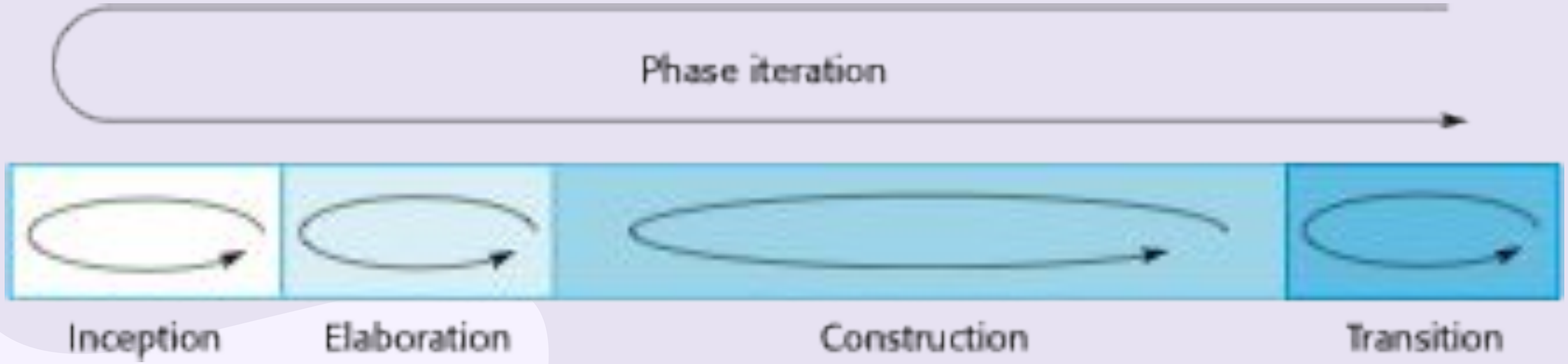
THE RATIONAL UNIFIED PROCESS

- A modern generic process derived from the work on the UML and associated process.
- Normally described from 3 perspectives
  - A dynamic perspective that shows phases over time;
  - A static perspective that shows process activities;
  - A proactive perspective that suggests good practice.

PHASES IN THE RATIONAL UNIFIED PROCESS

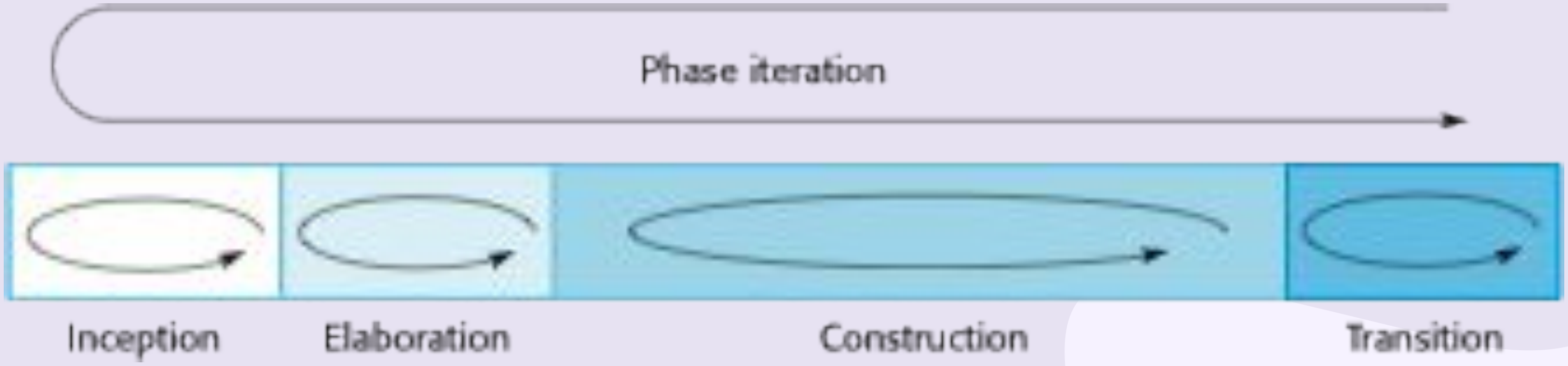


PHASES IN THE RATIONAL UNIFIED PROCESS



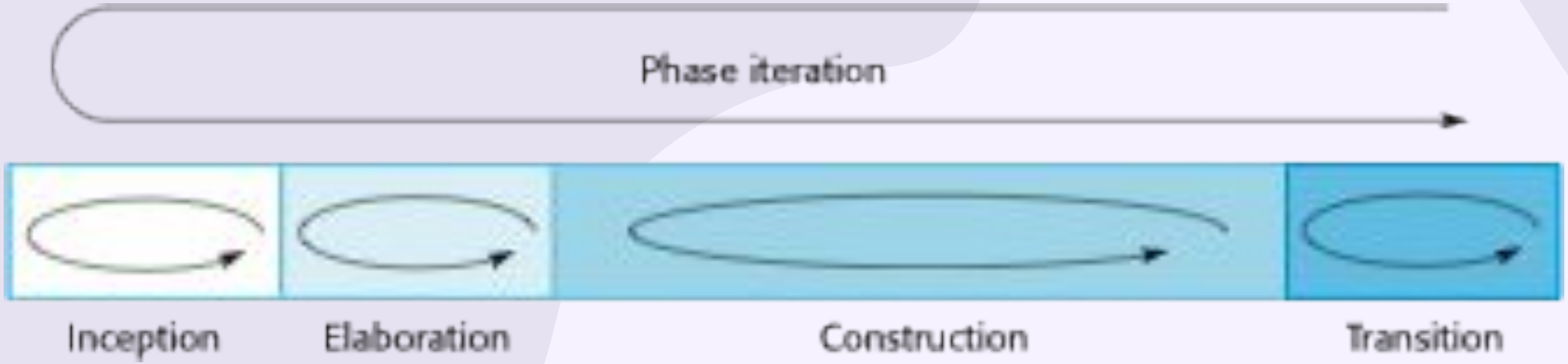
INCEPTION- establish the business case for the system.

PHASES IN THE RATIONAL UNIFIED PROCESS



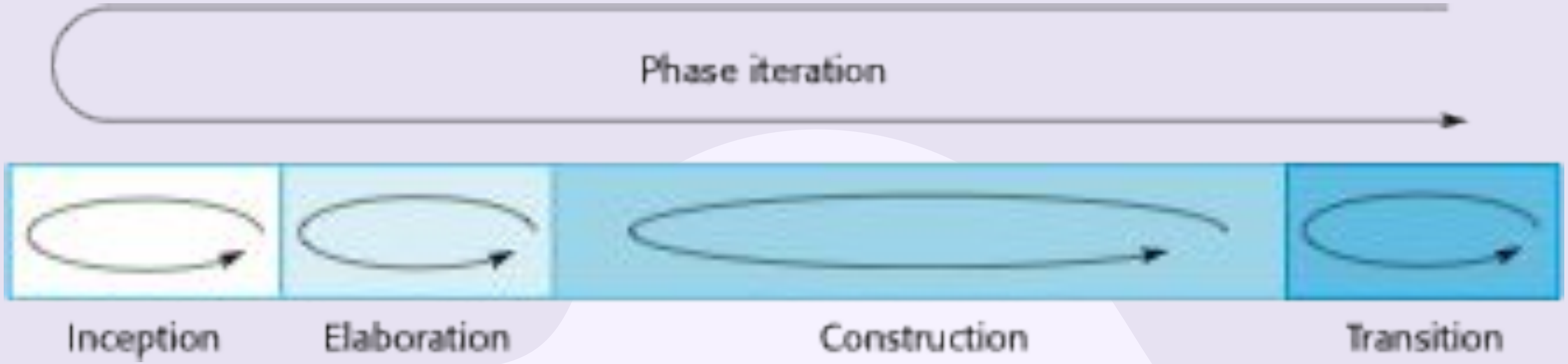
ELABORATION- develop an understanding of the problem domain and the system architecture.

PHASES IN THE RATIONAL UNIFIED PROCESS



CONSTRUCTION- system design, programming and testing.

PHASES IN THE RATIONAL UNIFIED PROCESS



TRANSITION- deploy the system in its operating environment.

RATIONAL UNIFIED PROCESS ITERATION

- **IN-PHASE ITERATION**- each phase is iterative with results developed incrementally.
- **CROSS-PHASE ITERATION**- as shown by the loop in the RUP model, the whole set of phases may be enacted incrementally.

STATIC WORKFLOWS IN THE RATIONAL UNIFIED PROCESS

WORKFLOW	DESCRIPTION
Business modelling	The business processes are modelled using business use cases.
Requirements	Actors who interact with the system are identified and use cases are developed to model the system requirements.
Analysis and design	A design model is created and documented using architectural models, component models, object models and sequence models.
Implementation	The components in the system are implemented and structured into implementation sub-systems. Automatic code generation from design models helps accelerate this process.

STATIC WORKFLOWS IN THE RATIONAL UNIFIED PROCESS

WORKFLOW	DESCRIPTION
Testing	Testing is an iterative process that is carried out in conjunction with implementation. System testing follows the completion of the implementation.
Deployment	A product release is created, distributed to users and installed in their workplace.
Configuration and change management	This supporting workflow managed changes to the system
Project management	This supporting workflow manages the system development
Environment	This workflow is concerned with making appropriate software tools available to the software development team.

GOOD PRACTICES OF THE RATIONAL UNIFIED PROCESS

- Develop software iteratively
- Manage requirements
- Use component-based architectures
- Visually model software
- Verify software quality
- Control changes to software

Chapter 2: Software Processes

Sabrinah Yonell C. Yap