

OVERVIEW Principles of Detailed Design I: What Detailed Design Is and Its Key Tasks

Detailed design takes the architecture's black-box components and refines them until they can be coded: their internal structure, their behaviour, their interfaces and their data. Architecture bridges requirements and design; detailed design bridges design and code. The lecture places it in the life cycle and lists its five key tasks: understand the architecture and the requirements allocated to your component, create the detailed design, evaluate it with technical reviews, document it in the software design document (SDD), and keep the implementation synchronised with it.

Refinement

Detailed design refines and expands the architecture until the design is complete enough to implement.

Both directions

From requirements and architecture forward to code, and from code backward (reverse engineering) to the design.

Five tasks

Understand → create → evaluate (technical reviews) → document (SDD) → monitor and control implementation.

Synchronicity

Detailed design follows the architecture and the code follows the detailed design; low synchronicity signals project risk.

HOW THE LECTURE IS ORGANIZED

Slides 1–6 What detailed design is and where it fits

Slides 7–9 Key tasks 1–2: understanding requirements, creating designs

Slides 10–12 Key tasks 3–4: technical reviews and the SDD

Slides 13–14 Key task 5: managing implementation, and summary

EXAM FOCUS

Final 252 Q1 asked for the primary focus of detailed design (the internals of components and their behaviour to meet functional requirements). Expect short questions on the five key tasks, technical-review guidelines, the SDD and synchronicity; Lecture 2 holds the SOLID questions.

HOW TO READ THIS PAGE

Every original slide is shown unchanged. **Yellow marks** on a slide point to a numbered note: hover or tap a mark to jump to its note, and hover a note to see its marks. Questions tagged **Past exam** appeared on a real paper; **Prediction** questions are likely angles, not confirmed.



Software Design and Architecture

[Principles of Detailed Design] – Chapter 05, L01

NOTES

- 1 Chapter 5 moves from architecture to **detailed design**, the second major design activity. Lecture 1 explains what it is and the tasks it involves; Lectures 2 and 3 cover the principles (SOLID) used to judge a detailed design.

SLIDE 2 / 14 Lecture Outlines

Lecture Outlines

➤ Overview of Detailed Design

- ✓ What is detailed design?
- ✓ Where does it fit?

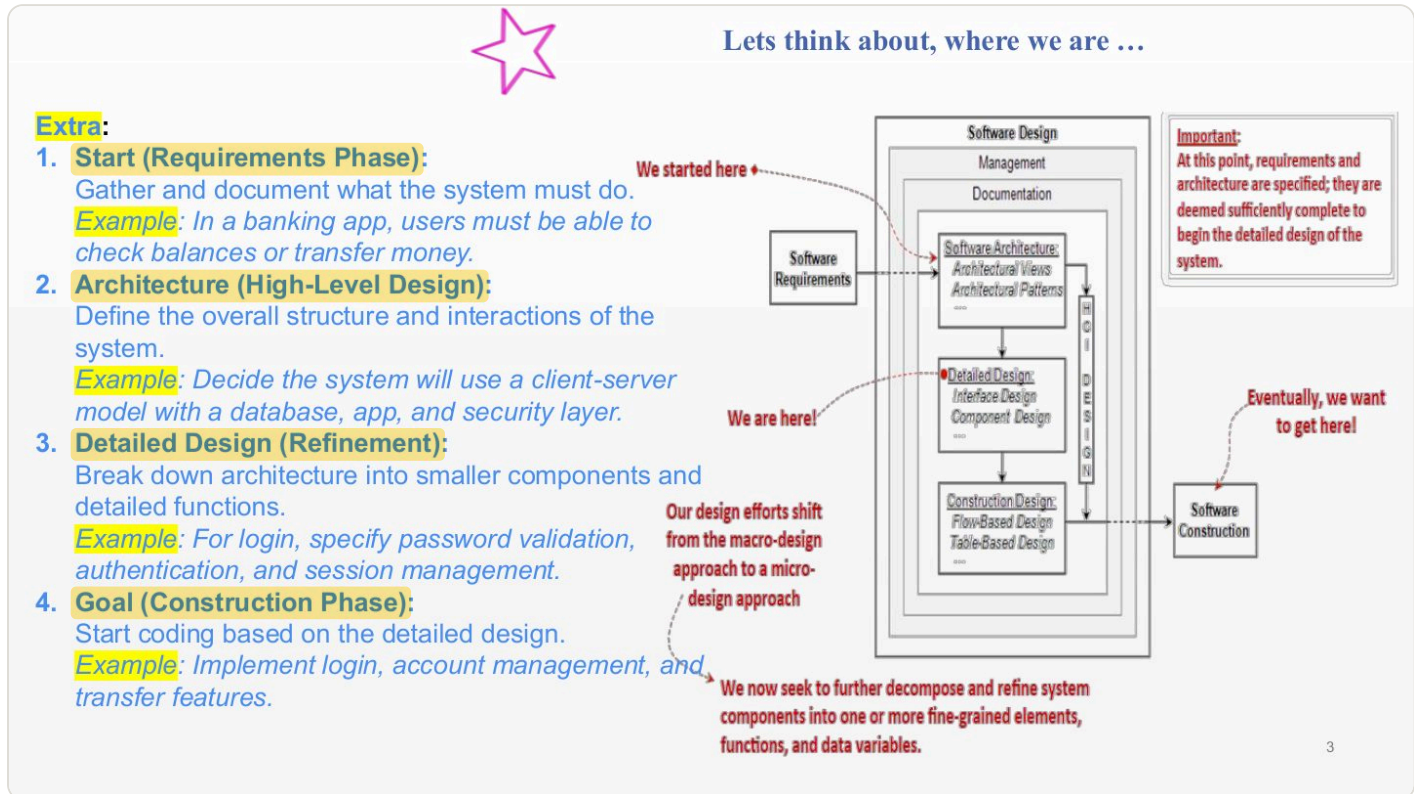
➤ Key Tasks in Detailed Design

- ✓ Understanding architecture and requirements – (chapters 3 & 4)
- ✓ Creating detailed designs
- ✓ Evaluating detailed designs
- ✓ Documenting detailed designs
- ✓ Monitoring and controlling implementation

NOTES

- 1 First: what detailed design is and where it fits in the life cycle.
- 2 Then its five key tasks, each on its own slide (7–13).

SLIDE 3 / 14 Where we are: from requirements to construction



NOTES

- 1 **Requirements:** gather and document what the system must do, e.g. banking-app users must check balances and transfer money.
- 2 **Architecture:** define the overall structure and interactions, e.g. a client-server system with a database, the app and a security layer.
- 3 **Detailed design:** break the architecture into smaller components and detailed functions, e.g. for login, specify password validation, authentication and session management. The diagram marks this "we are here": design shifts from a macro approach to a **micro** one, decomposing components into fine-grained elements, functions and data.
- 4 **Construction:** start coding from the detailed design (login, account management, transfers).

What is Detailed Design?

➤ Detailed design is:

- ✓ The process of **refining** and **expanding** the preliminary design phase *software architecture* of a system or component to the extent that the design is **sufficiently complete** to be implemented.

➤ During **Detailed Design** designers go deep into each component to define its **internal structure** and **behavioral** capabilities, and the resulting design leads to natural and efficient construction of software.

➤ **Extra:**

In this stage, designers focus on:

- ✓ **Internal structure:** Defining how each component works internally.
- ✓ **Behavior:** Specifying how each component behaves and reacts in different scenarios.

Extra:

1. Detailed Design (Refinement Phase)

•What happens here:

You don't write the actual code yet. Instead, you **specify in detail** how each part of the system will work.

•**Output:** Detailed specifications, algorithms, database schemas, interface definitions, class diagrams, etc.

•Example:

- Login requires: password validation rules (min length, special characters).
- Authentication: use hashing + salting.
- Session management: define session timeout, refresh tokens.

👉 Think of this as **blueprints before construction**.

2. Construction Phase (Implementation)

•What happens here:

Developers **translate the detailed design into actual code**.

•**Output:** Executable software (source code + builds).

•Example:

- Writing the actual `validatePassword()` function in Java/Python.
- Implementing the authentication API.
- Coding the session management module.

👉 Think of this as **building the house according to the blueprint**.

NOTES

- 1 Definition:** detailed design is the process of **refining and expanding** the preliminary design (the software architecture) of a system or component until the design is **complete enough to be implemented**.
- 2** Designers go deep into each component to define its **internal structure** and **behaviour**, so that construction becomes natural and efficient. Structure = how the component is built inside; behaviour = how it acts and reacts in different scenarios.
- 3** Detailed design produces specifications, algorithms, database schemas, interface definitions and class diagrams, but **no code yet**. Login example: password rules (minimum length, special characters), hashing plus salting for authentication, session timeout and refresh tokens. It is the blueprint before building.
- 4 Construction** then translates that design into code: writing `validatePassword()`, implementing the authentication API, coding session management. It is building the house from the blueprint.

What is the primary focus of "Detailed Design"?

- A. External properties of software components and non-functional requirements.
- B. The overall development process for both hardware and software.
- C. The internals of system components and their behavior to fulfill functional requirements.
- D. Project management tasks like staffing and tracking.

Answer: C. The internals of system components and their behavior to fulfill functional requirements.

SLIDE 5 / 14 What is Detailed Design?

What is Detailed Design?

Detailed design is closely related to architecture and construction; therefore successful designers (during detailed design) are required to have or acquire full understanding of the system's requirements and architecture.

- ✓ Just as architecture provides the bridge between requirements and design, detailed design provides the bridge between design and code.
- ✓ They must also be proficient in particular design strategies (e.g., object-oriented), programming languages, and methods and processes for software quality control.

NOTES

- 1 Detailed design sits between **architecture** and **construction**, so detailed designers need a **full understanding** of both the requirements and the architecture they are refining.
- 2 The bridge analogy: architecture bridges **requirements and design**; detailed design bridges **design and code**.
- 3 Designers must also be proficient in **design strategies** (e.g. object-oriented), programming languages, and quality-control methods, because their output must be directly implementable.

Prediction

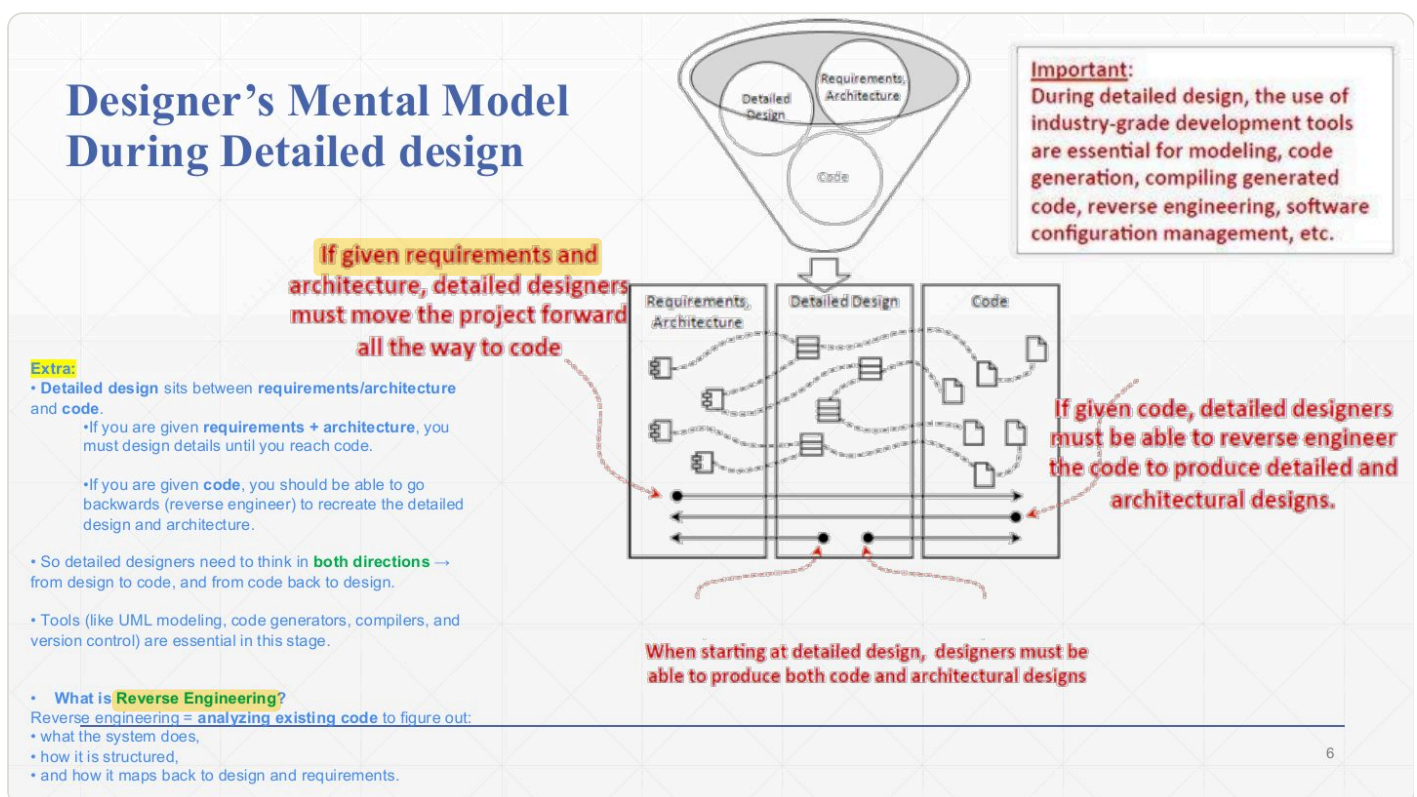
Q1

Detailed design provides the bridge between:

- A. Design and code
- B. Requirements and architecture
- C. Testing and deployment
- D. Users and managers

Answer: A. Design and code

SLIDE 6 / 14 Designer's Mental Model During Detailed Design



NOTES

- Forward:** given requirements and architecture, detailed designers must move the project all the way to code (the funnel: requirements/architecture → detailed design → code).
- Backward:** given code, they must be able to **reverse engineer** it into detailed and architectural designs. Starting at detailed design, you must be able to produce both the code and the architecture.
- Reverse engineering** means analysing existing code to recover what the system does, how it is structured, and how it maps back to design and requirements (useful for legacy systems and maintenance).
- Tools** are essential at this stage: UML modelling, code generation, compilers, reverse-engineering tools and configuration management.

Key Tasks in Detailed Design

➤ The major tasks identified for carrying out the detailed design activity include:

1. Understanding the architecture and requirements
2. Creating detailed designs
3. Evaluating detailed designs
4. Documenting software design
5. Monitoring and controlling implementation

NOTES

- 1 **1. Understand** the architecture and the requirements allocated to your components (slide 8).
- 2 **2. Create** the detailed designs: interfaces, GUI, internal structure and behaviour, data (slide 9).
- 3 **3. Evaluate** them, mainly through technical reviews (slide 10).
- 4 **4. Document** them, mainly in the SDD (slides 11–12).
- 5 **5. Monitor and control implementation**, keeping code and design in sync (slide 13). The tasks form a loop: a review or a coding discovery can send you back to an earlier task.

EXAM ANGLE

Prediction

Q1

List the five key tasks of detailed design.

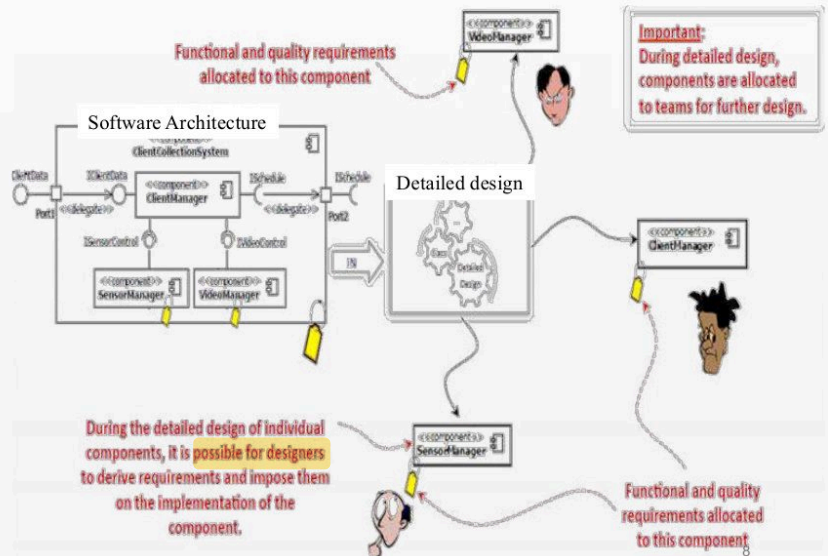
Answer: Understanding the architecture and requirements; creating detailed designs; evaluating detailed designs (technical reviews); documenting the design (SDD); monitoring and controlling implementation (synchronicity).

1. Understanding Architecture and Requirements

- Unlike the software architecture, where the complete set of requirements are evaluated and well understood, designers during detailed design activity focus on requirements allocated to their specific components.

Extra:

During **detailed design**, big components from architecture (e.g., **SensorManager**, **VideoManager**, **ClientManager**) are **assigned to teams**. Each team refines its component, adds detailed requirements (like validation or performance), and prepares it for **implementation (coding)**.



NOTES

- 1 In architecture the **whole** set of requirements is evaluated. In detailed design, each designer focuses on the **requirements allocated to their specific component**.
- 2 Architecture components (SensorManager, VideoManager, ClientManager in the diagram) are **assigned to teams**, each with the functional and quality requirements allocated to it. Each team refines its component and prepares it for coding.
- 3 During detailed design, designers may also **derive new requirements** for their component (validation rules, performance limits) and impose them on its implementation.

2. Creating Detailed Designs

- After the architecture and requirements for assigned components are well understood, the detailed design of software components can begin. (Detailed design consist of both **structural and behavioral designs**).
- When creating detailed designs, focus is placed on the following:
 - ✓ **Interface Design** - Internal & External
 - ✓ **Graphical User Interface (GUI) Design**
 - ✓ **Internal Component Design**
 - ✓ Structural
 - ✓ Behavioral
 - ✓ **Data Design** ~ *Database; data dictionary*

NOTES

- 1 Once the architecture and requirements for the assigned component are understood, detailed design starts. It covers both **structural and behavioural** design.
- 2 **Interface design:** internal interfaces (between components of the system) and external ones (with other systems).
- 3 **GUI design:** the screens users interact with.
- 4 **Internal component design:** the structure (classes, relationships) and behaviour (interactions, states) inside each component. Lectures 2–3 give the principles for this.
- 5 **Data design:** the database and the data dictionary (the definitions of every data element).

3. Evaluating Detailed Designs

- The most popular technique for evaluating detailed designs involves **Technical Reviews**. When conducting technical reviews, keep in mind the following:
 - ✓ Send a **review notice** with enough time for others to have appropriate time to thoroughly review the design.
 - ✓ Include a **technical expert** in the review team, as well as **stakeholders** of your design.
 - ✓ Include a member of the software **quality assurance** or **testing team** in the review.
 - ✓ During the review, **focus on** the important aspects of your designs; those that **show how your design helps meet functional and non-functional requirements**.
 - ✓ Document the review process. (Make sure that any **action items** generated during the review are **captured** and assigned for processing)

Extra:

Technical Reviews (to evaluate detailed design):

1. **Send review notice early (give time to prepare).**
2. **Include experts, stakeholders, and QA/test team.**
3. **Focus on how design meets functional & non-functional requirements.**
4. **Document review and assign action items.**

Example: When reviewing the design of an e-commerce checkout system, focus on how the design ensures secure payments (non-functional) and smooth order processing (functional).

NOTES

- 1 The most popular way to evaluate detailed designs is a **technical review**.
- 2 Send the **review notice** early enough that reviewers can study the design thoroughly.
- 3 Include a **technical expert**, the **stakeholders** of your design, and a member of the **QA or testing** team.
- 4 During the review, focus on what matters: how the design meets the **functional and non-functional** requirements. E-commerce checkout example: secure payments (non-functional) and smooth order processing (functional).
- 5 **Document** the review, and make sure every **action item** is captured and assigned to someone.

EXAM ANGLE

Prediction

Q1

Give three guidelines for conducting a technical review of a detailed design.

Answer: Send the review notice early so reviewers can prepare; include a technical expert, stakeholders and a QA/testing member; focus on how the design meets functional and non-functional requirements; document the review and assign every action item.

4. Documenting Detailed Designs

- Documentation of a project's software design is mostly captured in the **software design document (SDD)**, also known as software design description. The SDD is used widely throughout the development of the software. (Used by programmers, testers, maintainers, systems integrators, etc.)
- Other forms of documentation include:
 - ✓ **Interface Control Document**
 - A document describes system's interfaces and how the components of the system communicate.
 - ✓ **Version Control Document**
 - Contains information about what is included in a software release, including different files, scripts and executable. Different versions of the design depend on specific software release

Extra:

Software Design Documentation

- **Software Design Document (SDD):** Main document describing the design. Used by programmers, testers, maintainers, and integrators.
- **Interface Control Document:** Defines system interfaces and how components communicate.
- **Version Control Document:** Lists what is included in each software release (files, scripts, executables).

NOTES

- 1 The design is documented mainly in the **software design document (SDD)**, also called the software design description. It is used throughout development by programmers, testers, maintainers and system integrators.
- 2 The **interface control document** describes the system's interfaces and how components communicate.
- 3 The **version control document** lists what each software release contains (files, scripts, executables); different versions of the design correspond to specific releases.

EXAM ANGLE

Prediction

Q1

Detailed designs are documented mainly in the:

- A. Software Design Document (SDD)
- B. Software Requirements Specification
- C. Test plan
- D. Project charter

Answer: A. Software Design Document (SDD)

4. Documenting Detailed Designs

➤ The sections of the SDD and sample table of contents:

Section	Description
Date of issue and status	Date of issue is the day on which the SDD has been formally released. Every time the SDD is updated and formally released, there should be a new date of issue.
Scope	Scope provides a high level overview of the intended purpose of the software. It sets a limit as to what the SDD will describe and defines the objectives of the software.
Issuing organization	Issuing organization is the company which produced the SDD.
Authorship	Authorship pertains to who wrote the SDD and certain copyright information.
References	References provide a list of all applicable documents that are referred to within the SDD. If there is a certain technology that is used within the design, it is important to refer to the corresponding documentation on that technology, so it may be referenced. When reading the referenced documents, stakeholders may uncover inconsistencies in how the technology should be used and how it is used in the software design.
Context	Description of the context of the SDD.
Body	Body is the main section of the SDD where the design is documented. This is where stakeholders look to understand the software and how it is to be constructed.
Summary	
Glossary	A glossary provides definitions for all software related terms and acronyms used in the SDD.
Change history	Change history is a brief description of the items added to, deleted from, or changed within the SDD.

1. Introduction
1.1. Date of Issue
1.2. Context
1.3. Scope
1.4. Authorship
1.5. Change history
1.6. Summary
2. Software Architecture
2.1. Overview
2.2. Stakeholders
2.3. System Design Concerns
2.4. Architectural Viewpoint 1
2.4.1. Design View 1
2.5. Architectural Viewpoint 2
2.5.1. Design View 2
2.6. Architectural Viewpoint <i>n</i>
2.6.1. Design View <i>n</i>
3. Detailed Design
3.1. Overview
3.2. Component 1 Design Viewpoint 1
3.2.1. Design View 1
3.3. Component 2 Design Viewpoint 2
3.3.1. Design View 2
3.4. Component <i>n</i> Design Viewpoint <i>n</i>
3.4.1. Design View <i>n</i>
4. Glossary
5. References

NOTES

- 1 The SDD sections and a sample table of contents.
- 2 **Front matter:** date of issue and status (each formal release gets a new date), scope (the intended purpose of the software), issuing organisation, authorship, references (every document or technology referred to), and context.
- 3 **Body:** the main section where the design itself is documented, the part stakeholders read to understand how the software is to be built. In the sample contents it holds the software architecture (stakeholders, concerns, architectural viewpoints and views) and the detailed design (one viewpoint per component).
- 4 **Back matter:** summary, **glossary** (definitions of all terms and acronyms) and **change history** (what was added, removed or changed).

5. Managing Implementation

- Detailed design **synchronicity** is concerned with the degree of how well detailed designs adhere to the software architecture and how well software code adheres to the detailed design.
 - ✓ Low degree of synchronicity points to a flaw in the process and can lead to software project failure.
- Particular attention needs to be paid when projects enter the **maintenance** phase or when **new engineers** are brought into the project.
- Processes must be in place to ensure that overall synchronicity is high

Extra:

Detailed Design Synchronicity

- Ensures detailed design follows architecture and code follows design.
- Low synchronicity → flaws, risk of project failure.
- Critical during maintenance or when new engineers join.
- Processes must keep synchronicity high.
- Example: If the architecture requires **encrypted login**, but the detailed design forgets encryption and the code just stores plain passwords → synchronicity is broken.

NOTES

- 1 **Detailed design synchronicity** measures how well the detailed design adheres to the architecture and how well the code adheres to the detailed design.
- 2 A **low degree** of synchronicity points to a flaw in the process and can lead to project failure.
- 3 It needs particular attention when the project enters **maintenance** or **new engineers** join, the moments when design and code drift apart. Processes must be in place to keep it high.
- 4 Example: the architecture requires **encrypted login**, the detailed design forgets encryption, and the code stores plain passwords. Each level broke synchronicity with the one above.

EXAM ANGLE

Prediction

Q1

What is detailed design synchronicity, and why does low synchronicity matter?

Answer: It is the degree to which the detailed design follows the architecture and the code follows the detailed design. Low synchronicity means the documentation no longer describes the real system; it points to a process flaw and can lead to project failure, especially during maintenance or when new engineers join.

Summary

➤ Overview of Detailed Design

- ✓ What is detailed design?
- ✓ Where does it fit?

➤ Key Tasks in Detailed Design

- ✓ Understanding architecture and requirements – (chapters 3 & 4)
- ✓ Creating detailed designs
- ✓ Evaluating detailed designs
- ✓ Documenting detailed designs
- ✓ Monitoring and controlling implementation

➤ Next ... Design Patterns

Prince Sultan University | Slides prepared from "Software Engineering Design: Theory and Practice" book and slides

15

NOTES

- 1 Recap: detailed design refines the architecture until it can be coded, through five tasks: understand, create, evaluate, document, monitor.
- 2 Next: the principles for judging a detailed design (SOLID, Lectures 2–3), then design patterns.

Original slides from the SE322 lecture deck, with study notes. Check exact wording against the [original deck](#).
Notes and questions are study aids. Made by Shoug Alomran.