

## OVERVIEW **Quality Attributes: Scenarios, Tactics and Design Decisions**

Functional requirements say what the system does; quality attributes say how well it does it, and the plan for meeting them lives in the architecture. Because quality words are vague and overlap (is a denial-of-service attack an availability, performance or security problem?), qualities are pinned down with quality attribute scenarios: six parts from source to response measure. Tactics are the design techniques that achieve a scenario's response, and the lecture ends with the seven categories of design decision through which an architect shapes quality.

### **QA vs. functional**

Functional = what the system shall do (planned in detailed design). Quality = criteria for judging the system as a whole (planned in the architecture).

### **Six-part scenario**

Source, stimulus, environment, artifact, response, response measure. The measure is what makes it testable.

### **Tactics**

Primitive, proven design techniques that control a quality attribute response (redundancy, caching...).

### **Seven decision types**

Responsibilities, coordination, data model, resources, binding time, technology, mapping.

## HOW THE LECTURE IS ORGANIZED

**Slides 1–5** Architecture and requirements: functional vs. quality, who cares about which quality

**Slides 6–12** Quality attribute considerations and scenarios, with three worked examples

**Slides 13–14** Tactics

**Slides 15–18** The seven categories of quality design decisions

## EXAM FOCUS

Scenarios are the most examined idea in SE322. Quiz 1 261 had six matching questions on scenario parts (one on the definitions, five on usability, performance, availability, testability and security examples), Midterm 261 Q10 (3 marks) asked for the six parts of an online-voting scenario, Midterm 252 Q5 asked for an availability scenario, and Final 252 Q3-2 asked you to construct a general availability scenario.

## 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

**[Quality Attributes]** – Chapter 03, L01

---

## NOTES

- 1 Chapter 3 is about **quality attributes**, the non-functional requirements that architecture exists to satisfy. Lecture 1 gives the tools (scenarios, tactics, design decisions); Lecture 2 applies them to six major qualities.

---

## SLIDE 2 / 18 Lecture Outlines

### Lecture Outlines

- Architecture and Requirements
- Quality Attribute Considerations
- Quality Design Decisions
- Major Quality Attributes

## NOTES

- 1 First, how functional and quality requirements relate to architecture.
- 2 Then how qualities are specified precisely (scenarios) and achieved (tactics).
- 3 Then the categories of design decisions that affect quality.
- 4 The six major quality attributes themselves come in Lecture 2.

## SLIDE 3 / 18 Architecture and Requirements

### Architecture and Requirements

➤ System requirements can be categorized as:

✓ **Functional requirements:**

- State what the system must do, usually in the form of “system shall do <requirements>”.
- Functional requirements define specific behavior or function
- The plan for implementing functional requirements is produced in the system detailed design.

✓ **Non-functional requirements or Quality attributes (QA):**

- Specify criteria that can be used to judge the operation of the system as a whole rather than specific behavior
- The plan for implementing non-functional requirements is detailed in the system architecture

## NOTES

- 1 **Functional requirements** state what the system must do, usually written as “the system shall ...”. Each defines a specific behaviour or function.
- 2 The plan for implementing functional requirements is produced in the **detailed design**.
- 3 **Non-functional requirements**, also called **quality attributes (QA)**, are criteria for judging how the system operates **as a whole**, not a specific behaviour: speed, reliability, security...
- 4 The plan for implementing them is detailed in the **architecture**. That split (functions → detailed design, qualities → architecture) is the reason architecture spends so much effort on quality.

Past exam · Quiz 1 · 261 · Q14 (true/false)

Q1

Software architecture is concerned only with functional structure and does not need to consider non-functional requirements.

**Answer: False** — Architecture must address important quality attributes such as performance, security and availability; their plan lives in the architecture.

SLIDE 4 / 18 **Architecture and Requirements**

## Architecture and Requirements

- Achieving quality attributes must be considered throughout design, implementation, and deployment.
- Architecture, by itself, is unable to achieve qualities. It provides the foundation for achieving quality, but this foundation will be to no use if attention is not paid to the details in later phases of software development.

## NOTES

- 1 Quality has to be considered in **every phase**: design, implementation and deployment. A fast architecture can still be ruined by slow code or a badly configured server.
- 2 Architecture **cannot achieve qualities on its own**. It provides the **foundation**, but the foundation is useless if the details in later phases are ignored (the same point as Chapter 2: architecture alone cannot guarantee quality).

## Architecture and Requirements

- Architectural documents are used by a variety of system stakeholders:
  - ✓ E.g. Developers, managers, sales, testers, support, maintenance, and customers
- Each stakeholder will have their own opinion about what non-functional requirement matters the most:
  - ✓ e.g. The development team will care about maintainability more than the customer
  - ✓ e.g. Quality assurance team will be more interested in the testability of the application
  - ✓ What will the end user be interested in???

### NOTES

- 1 Architecture documents are read by many stakeholders: developers, managers, sales, testers, support, maintenance staff and customers.
- 2 Each has **their own opinion** about which quality matters most, so the architect has to weigh them against each other.
- 3 The development team cares about **maintainability** (easy to change) more than the customer does; the QA team cares about **testability**.
- 4 The **end user** (the slide's question) mostly cares about qualities they can feel: usability, performance and availability.

### EXAM ANGLE

#### Prediction

Q1

Which quality attribute is the QA (testing) team most likely to prioritise, and which the development team?

**Answer:** QA: testability. Development team: maintainability (modifiability).

## Quality Attribute Considerations

- Quality attributes are **constraints on the manner** in which the system implements and delivers its functionality (**how the system achieves its functionality**).
- Extra: Quality attributes are used as **criteria for evaluating** the performance and effectiveness of the system during and after development.
  - ✓ i.e. Efficiency, availability, scalability, adaptability, security, dependability, etc. ...
- **Example:** assume the following functional requirement “when the user presses the **green** button the Options dialog appears”:
- ✓ A **performance QA** might describe how quickly the dialog will appear
- ✓ An **availability QA** might describe how often this function will fail
- ✓ A **usability QA** might describe how easy it is to learn this function.

### NOTES

- 1 Quality attributes are **constraints on the manner** in which the system implements and delivers its functionality: they say *how* a function must be achieved, not *what* it is.
- 2 They are used as **criteria** to evaluate the system during and after development: efficiency, availability, scalability, adaptability, security, dependability...
- 3 Take one functional requirement: “when the user presses the **green** button, the Options dialog appears”. The function is the same in every case below; only the quality constraint changes.
- 4 A **performance QA** says how quickly the dialog appears; an **availability QA** says how often this function may fail; a **usability QA** says how easy the function is to learn.

### EXAM ANGLE

#### Prediction

Q1

For the requirement “when the user presses the green button the Options dialog appears”, write one performance and one usability quality requirement.

**Answer:** Performance: the dialog appears within 0.5 seconds of the press. Usability: a first-time user can find and open the dialog without help in under 10 seconds.

## Quality Attribute Considerations – Scenarios

- Quality attributes can have **unstable definition** or an **overlapping concerns**. **Example:**  
Is a system failure due to a **denial of service attack** an aspect of availability, performance, security, or usability?
- **Extra:** A Denial-of-Service (DoS) attack:
  - ✓ Availability → system is down.
  - ✓ Performance → system becomes very slow.
  - ✓ Security → the attack is a security breach.
  - ✓ Usability → users can't access the service.
- Here, the same problem “overlaps” multiple attributes.
- A solution for this (unstable definitions and overlapping concerns) is to use **quality attribute scenarios** as a means of characterizing quality attributes.
  - Extra:**
  - Solution: Quality Attribute Scenarios**
    - Instead of **vague** terms, define concrete scenarios that describe:
      - Stimulus (what happens),
      - Response (what the system should do),
      - Measure (how success is judged).
    - ◆ **Real Example**
      - Attribute: **Availability**
        - Scenario: If the **payment server crashes** during a transaction (**stimulus**),
        - The system should automatically switch to a backup server within 2 seconds (**response**),
        - With 99.99% uptime guaranteed (**measure**).
    - This way, the attribute is **clear, testable, and not overlapping**.

7

### NOTES

- 1 Quality attribute words have **unstable definitions** and **overlapping concerns**: people mean different things by “reliable” or “secure”.
- 2 The DoS example shows the overlap: a denial-of-service attack makes the system down (availability), very slow (performance), is an attack (security) and stops users reaching the service (usability). Which quality does it “belong” to?
- 3 The solution is to characterise qualities with **quality attribute scenarios**: concrete stories with a stimulus, a response and a measure, instead of vague words.
- 4 Real example: if the payment server **crashes during a transaction** (stimulus), the system switches to a backup within 2 seconds (response), keeping 99.99% uptime (measure). That is clear, testable and no longer overlaps with other qualities.

## Quality Attribute Considerations – Scenarios

➤ A quality attribute scenario has the following parts:

1. **Stimulus** – a condition that needs to be considered
2. **Source of stimulus** - (e.g., human, computer system, etc.)
3. **Environment** - what are the conditions when the stimulus occurs?
4. **Artifact** – what elements of the system are stimulated.
5. **Response** – the activity undertaken after arrival of the stimulus
6. **Response measure** – when the response occurs it should be measurable so that the requirement can be tested.

## Extra: 6 Steps for Using Quality Attribute Scenarios

1. **Source** → Who triggers the action (e.g., user, attacker, system).
2. **Stimulus** → The action or event that happens (e.g., request, failure, attack).
3. **Environment** → The situation in which it happens (e.g., under load, during maintenance).
4. **Artifact** → The part of the system affected (e.g., database, server, UI).
5. **Response** → How the system reacts (e.g., log error, retry, deny access).
6. **Response Measure** → How success is judged (e.g., response time < 2 sec, 99% uptime).

## NOTES

- 1 **Stimulus:** a condition that needs to be considered when it arrives at the system: a request, a failure, an attack, a change request.
- 2 **Source of stimulus:** the entity that generates it: a human, another computer system, an internal component.
- 3 **Environment:** the conditions the system is in when the stimulus occurs: normal operation, overload, start-up, maintenance.
- 4 **Artifact:** the element of the system that is stimulated: the whole system, a database, a server, the UI, the code.
- 5 **Response:** the activity the system undertakes after the stimulus arrives: process it, log it, fail over, deny access.
- 6 **Response measure:** the response must be **measurable**, so the requirement can be tested: response time under 2 seconds, 99% uptime, repair within 30 minutes. A scenario without a measure is not testable.

Match each quality attribute scenario element with its meaning: source, stimulus, artifact, environment, response, response measure.

**Answer:** Source → **entity that generates the stimulus**. Stimulus → **event or condition that affects the system**. Artifact → **system element affected by the stimulus**. Environment → **condition under which the stimulus occurs**. Response → **what the system does after the stimulus**. Response measure → **measurable criterion used to evaluate the response**.

## SLIDE 9 / 18 A Quality Attribute Scenario Template

### A Quality Attribute Scenario Template

- A template for writing a quality attribute scenario which is commonly used is:

*Some **(source)** generates some events **(stimulus)** that arrives at some **artefact** under some conditions **(environment)** and must be dealt with **(response)** in a satisfactory way **(response measure)***

## NOTES

- 1 The one-sentence template: some **source** generates some event (**stimulus**) that arrives at some **artifact** under some conditions (**environment**) and must be dealt with (**response**) in a satisfactory way (**response measure**). Write any scenario by filling these six slots in this order. **Source**: who or what generates the stimulus. **Stimulus**: the event or condition that arrives. **Environment**: the conditions the system is in when it arrives. **Artifact**: the part of the system that is stimulated. **Response**: what the system does. **Response measure**: how the response is measured, so the requirement can be tested.

## Past exam · Midterm 261 · Q10 · 3 marks

Q1

Online Voting System: "During an election, thousands of eligible voters access the Online Voting System to submit their votes. While the system is operating under heavy election-day traffic, a voter submits a vote for a selected candidate. The system must validate the voter, securely record the vote, prevent duplicate voting, and display a confirmation message. The complete voting transaction must be processed within 4 seconds for at least 95% of voting requests." Identify the six elements of the quality attribute scenario.

**Answer:** **Source:** eligible voters. **Stimulus:** submitting a vote for a selected candidate. **Environment:** heavy election-day traffic. **Artifact:** the Online Voting System. **Response:** validate the voter, securely record the vote, prevent duplicate voting and display a confirmation. **Response measure:** processed within 4 seconds for at least 95% of voting requests.

## Past exam · Quiz 1 · 261 · Q24 (matching)

Q2

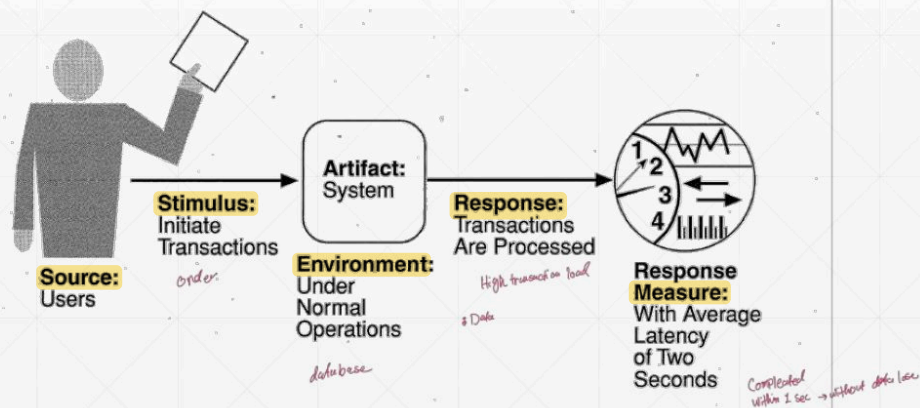
Performance scenario: "During peak registration time, an enrolled student submits a course search request to the university registration system. The system processes the request and displays matching course sections. The search results must be displayed within 2 seconds for at least 95% of requests." Match the six elements.

**Answer:** **Source:** enrolled student. **Stimulus:** submits a course search request. **Artifact:** university registration system. **Environment:** peak registration time. **Response:** processes the request and displays matching course sections. **Response measure:** results displayed within 2 seconds for at least 95% of requests.

## SLIDE 10 / 18 Quality Attribute Scenarios: a performance example

## Quality Attribute Considerations – Scenarios

## A Performance Scenario Example



## NOTES

- 1 A **performance** scenario drawn as a picture. Read it left to right, like the template on slide 9.
- 2 **Source:** users. **Stimulus:** they initiate transactions (the handwritten note gives "orders" as an example). **Artifact:** the system. **Environment:** under normal operations.
- 3 **Response:** the transactions are processed. **Response measure:** with an average latency of two seconds. The measure is what turns "the system should be fast" into a requirement you can test.

## EXAM ANGLE

### Prediction

Q1

In the performance scenario on this slide, which part is "with average latency of two seconds"?

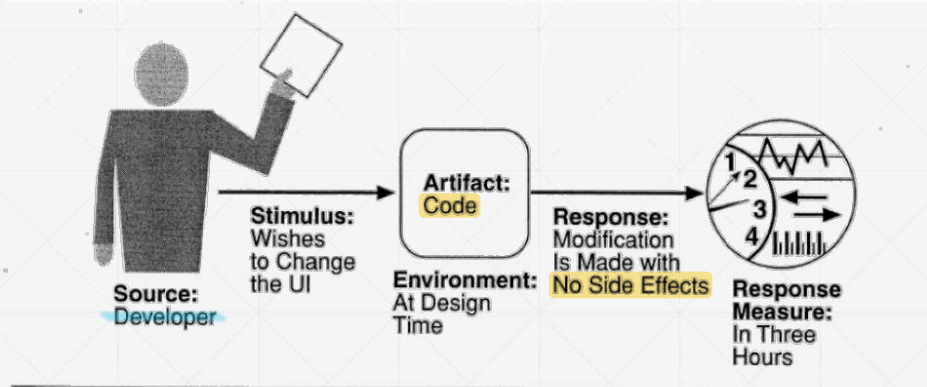
- A. Response measure
- B. Response
- C. Environment
- D. Stimulus

**Answer: A. Response measure**

## SLIDE 11 / 18 Quality Attribute Scenarios: a maintainability example

### Quality Attribute Considerations – Scenarios

#### A maintainability Scenario Example



## NOTES

- 1 A **maintainability** (modifiability) scenario. Here the "stimulus" is not a failure or a request but a wish to change the system.
- 2 **Source:** a developer. **Stimulus:** wishes to change the UI. **Artifact:** the code. **Environment:** at design time (not while the system runs).

- 3 **Response:** the modification is made with no side effects. **Response measure:** in three hours. If the UI is well separated from the rest of the code, the change stays small and the measure is met.

## EXAM ANGLE

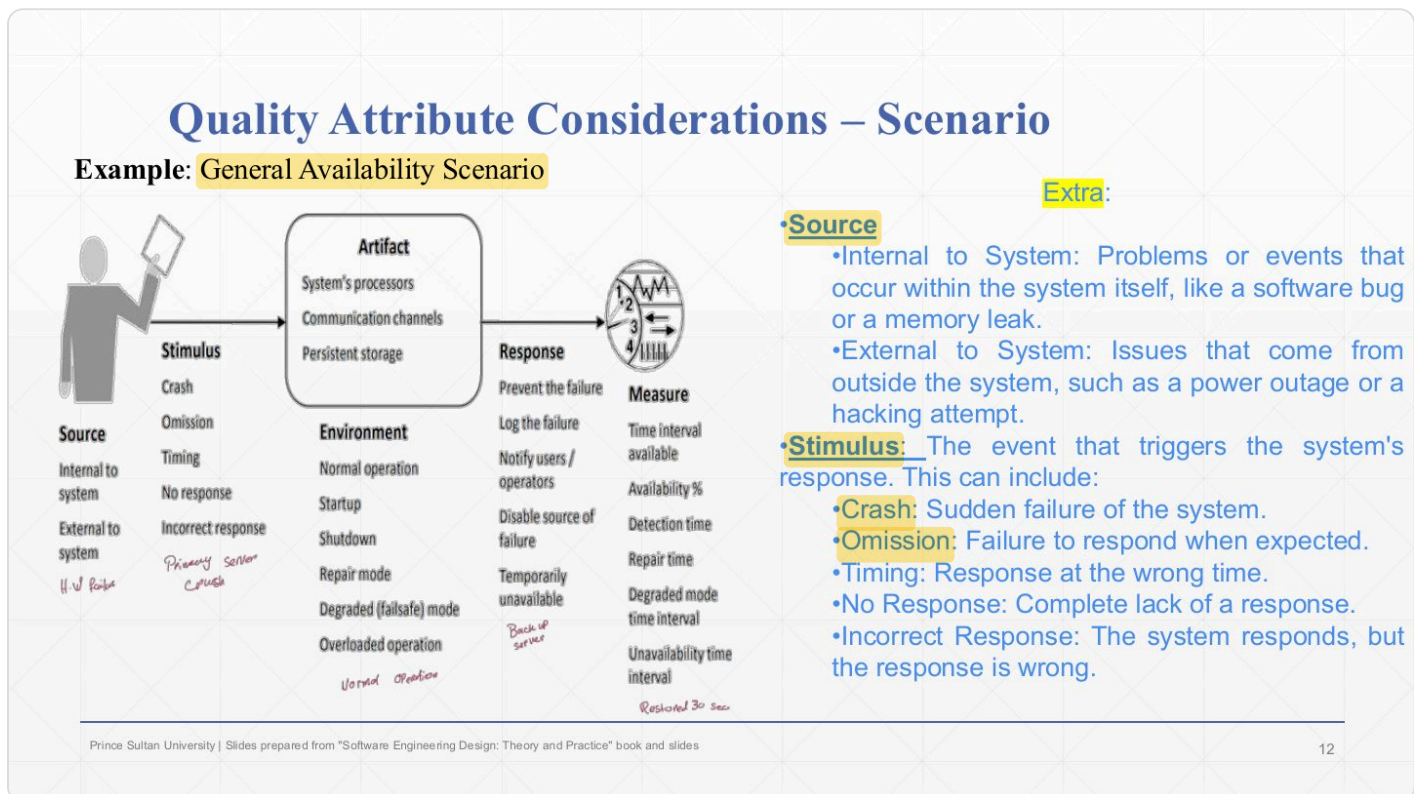
### Prediction

Q1

Write a modifiability scenario in which a developer must add a new payment method.

**Answer:** Source: developer. Stimulus: needs to add a new payment method. Artifact: the payment module code. Environment: design/development time. Response: the method is added and tested without changing other modules (no side effects). Measure: within 2 person-days, touching no more than one module.

## SLIDE 12 / 18 Quality Attribute Considerations Scenario



## NOTES

- 1 A **general** scenario lists the *possible* values of each part for one quality (here availability). A concrete scenario then picks one value per part.
- 2 **Source:** internal to the system (a software bug, a memory leak) or external (a power outage, a hacking attempt).
- 3 **Stimulus** (the fault): **crash** (sudden failure), **omission** (no response when one was expected), **timing** (response at the wrong time), no response, or an incorrect response.

4 The rest of the picture:

**Artifact**

The system's processors, communication channels or persistent storage.

**Environment**

Normal operation, start-up, shutdown, repair mode, degraded (fail-safe) mode, overloaded operation.

**Response**

Prevent the failure, log it, notify users or operators, disable the source of the failure, or be temporarily unavailable.

**Measure**

Time interval available, availability %, detection time, repair time, degraded-mode time interval, unavailability time.

5 The handwritten notes show one concrete pick: hardware failure → primary server crash → normal operation → switch to a backup server → restored in 30 seconds.

---

**EXAM ANGLE**

Past exam · Final 252 · Q3-2 · 2 marks

Q1

Construct one quality attribute general scenario for availability.

**Answer:** **Source:** heartbeat monitor (internal to the system). **Stimulus:** a server becomes unresponsive (crash/omission). **Environment:** normal operation. **Artifact:** the server process. **Response:** detect the fault, notify the operator, disable the failed source and switch to the backup, then repair it. **Response measure:** failover within 3 seconds; repair time under 30 minutes; 99.9% availability. All six parts with a measurable response measure earn the marks.

Past exam · Quiz 1 · 261 · Q25 (matching)

Q2

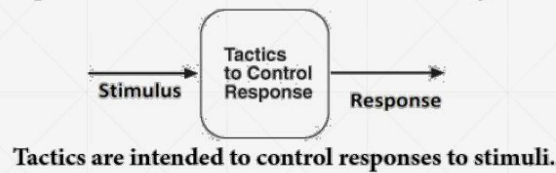
Availability scenario: "A monitoring service detects that the primary application server has stopped responding during normal operation. The system redirects requests to a standby server and restores service within 30 seconds with no more than 0.1% failed requests." Match the six elements.

**Answer:** **Source:** monitoring service. **Stimulus:** detects the primary application server has stopped responding. **Artifact:** application service/server infrastructure. **Environment:** normal operation. **Response:** redirects requests to a standby server and restores service. **Response measure:** restored within 30 seconds with no more than 0.1% failed requests.

---

## Quality Attribute Considerations – Tactics

- There are a collection of **primitive design techniques** that an architect can use to achieve a quality attribute **response**.
- We call these architectural design primitives **tactics**.
- Tactics, **like design patterns**, are techniques that architects have been used for years by experienced architects that we can simply capture and apply in our designs.
- We will show examples of tactics later in the following lecture.



### NOTES

- 1 Architects have a collection of **primitive design techniques** for achieving a quality attribute response.
- 2 These architectural design primitives are called **tactics**. A tactic targets one quality response (Chapter 2: a tactic is a design decision that controls a quality attribute response).
- 3 Like design patterns, tactics are proven techniques experienced architects have used for years; we capture them and apply them. Lecture 2 lists tactics for each major quality.

## Extra Examples of Tactics

### ➤ 1. Redundancy

- **Purpose:** To prevent system failure in case of a component failure.
- **Tactic:** Implement multiple instances of critical components so that if one fails, others can take over.
- **Stimulus:** Hardware failure, network outage.
- **Response:** Automatic failover to backup systems.

### ➤ 2. Caching

- **Purpose:** To improve response time and reduce load on the system.
- **Tactic:** Store frequently accessed data in a temporary storage area to make it quickly accessible.
- **Stimulus:** High number of read requests.
- **Response:** Serve data from cache rather than re-fetching it from the primary storage.

## NOTES

- 1 Redundancy** (availability): run multiple instances of critical components so that if one fails another takes over. Stimulus: hardware failure or network outage. Response: automatic failover to the backup.
- 2 Caching** (performance): keep frequently accessed data in fast temporary storage. Stimulus: many read requests. Response: serve from the cache instead of fetching from primary storage, which cuts response time and load.

## EXAM ANGLE

### Prediction

Q1

Keeping several instances of a critical component so that another takes over when one fails is a tactic for:

- A. Availability
- B. Usability
- C. Testability
- D. Modifiability

**Answer: A. Availability**

## Quality Design Decisions

➤ Architecture design is a **systematic approach to making design decisions**.

These decisions can be categorized into:

1. **Allocation of system responsibilities** (responsibilities of system functions)
2. **Coordination model** (coordination between system elements)
3. **Data model** (identify data abstractions, their operations and properties)
4. **Management of resources**
5. **Binding time decisions** (Determine how and when architectural elements are bounded)
6. **Choice of technology**
7. **Mapping among architectural elements** (mapping of modules and runtime elements )

15

### NOTES

- 1 Architecture design is a **systematic** way of making design decisions, and every quality is achieved through them. They fall into seven categories.
- 2 The first three are about the system's parts: **allocation of responsibilities** (which element does which function), the **coordination model** (how elements interact) and the **data model** (data abstractions, their operations and properties).
- 3 The last four: **management of resources**, **binding time** (how and when elements are bound), **choice of technology**, and **mapping** among elements (modules to run-time elements). Slides 16–17 give an example of each.

### EXAM ANGLE

#### Prediction

Q1

List the seven categories of quality design decisions.

**Answer:** Allocation of responsibilities, coordination model, data model, management of resources, binding time decisions, choice of technology, mapping among architectural elements.

### Extra: Quality Design Decisions Explanation

#### 1. Allocation of System Responsibilities:

- ✓ This is about deciding which parts of the system will handle specific tasks.
- ✓ For example, in a shopping app, one part handles user logins, another manages the shopping cart, and other processes payments.

#### 2. Coordination Model:

- ✓ This involves figuring out how different parts of the system will communicate and work together.
- ✓ For example, how the payment system informs the inventory system when an item has been purchased.

#### 3. Data Model:

- ✓ This decision revolves around the types of data the system will use, how it's organized, and how it's accessed.
- ✓ For example, deciding to use a customer object that includes name, address, and order history.

16

### NOTES

- 1 Allocation of responsibilities:** deciding which part handles which task. In a shopping app, one part handles logins, another the cart, another payments.
- 2 Coordination model:** how the parts communicate and work together, e.g. how the payment system tells the inventory system an item was bought (direct call? event? message queue?).
- 3 Data model:** which data the system uses, how it is organised and accessed, e.g. a Customer object with name, address and order history.

**4. Management of Resources:**

- ✓ This is about deciding how the system's resources like memory, processor time, and network bandwidth are used and allocated.
- ✓ For example, ensuring there are enough server resources during a high traffic period on a website.

**5. Mapping among Architectural Elements**

- ✓ This involves planning out how different parts of the system (like code modules) correspond to runtime elements (like processes or tasks).
- ✓ For example, mapping a user interface module to specific services that run when the application is used.

**6. Binding Time Decisions:**

- ✓ These decisions determine when parts of the system are set or chosen, such as when certain modules are loaded or when specific configurations are applied.
- ✓ For instance, choosing whether a user's language preference is set at installation or at runtime.

**7. Choice of Technology:**

- ✓ This is about selecting the technologies the system will use, like programming languages, databases, or hardware platforms.
- ✓ For example, deciding to use Python for server-side logic, PostgreSQL for the database, and AWS for cloud hosting.

**NOTES**

- 1 Management of resources:** how memory, CPU time and network bandwidth are used and allocated, e.g. making sure there is enough server capacity during peak traffic (a performance decision).
- 2 Mapping:** how code modules correspond to run-time elements (processes, tasks), e.g. which services run when the UI module is used.
- 3 Binding time:** *when* something is fixed: at compile time, installation, start-up or run time. Example: is the user's language chosen at installation or switchable at run time? Later binding gives flexibility (modifiability) at some cost (performance, complexity).
- 4 Choice of technology:** languages, databases, platforms, e.g. Python on the server, PostgreSQL, AWS hosting.

**EXAM ANGLE****Prediction****Q1**

Deciding whether a user's language is set at installation or can change at run time is which kind of design decision?

- A. Binding time decision
- B. Coordination model
- C. Data model
- D. Management of resources

**Answer: A. Binding time decision**

## Summary

- Architecture and Requirements
- Quality Attribute Considerations
- Quality Design Decisions
- Next ... Major Quality Attributes

### NOTES

- 1 Next lecture: the major quality attributes one by one (availability, performance, security, modifiability, testability, usability...), each with its general scenario and tactics. Practise writing one six-part scenario for each.

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.