

OVERVIEW Major Quality Attributes: Definitions, Scenarios and Tactics

Six quality attributes, each in the same three steps: a definition, a general scenario (the possible values of the six scenario parts) with a concrete example, and a tree of tactics that achieve it. Availability is about failures and their duration, modifiability about the cost of change, performance about timeliness, security about resisting unauthorised access, testability about how easily faults can be revealed, and usability about how easily users accomplish tasks.

Availability

Detect faults (ping/echo, heartbeat, exceptions), prepare and repair (voting, redundancy, spare), reintroduce (shadow, resync, rollback), prevent (removal from service, transactions, process monitor).

Modifiability and performance

Modifiability: localise changes, prevent ripple effects, defer binding time. Performance: resource demand, management, arbitration.

Security and testability

Security: resist, detect, recover from attacks.
Testability: manage input/output, internal monitoring.

Usability

Separate the user interface, support user initiative (cancel, undo, aggregate), support system initiative (user, system, task models).

HOW THE LECTURE IS ORGANIZED

Slides 2–3 Definitions of the major quality attributes

Slides 4–6 Availability: general scenario, example, tactics

Slides 7–10 Modifiability: scenarios and tactics

Slides 11–13 Performance: scenario and tactics

Slides 14–16 Security: scenario and tactics

Slides 17–19 Testability: scenario and tactics

Slides 20–22 Usability: scenario and tactics

Slides 23–24 Summary and a tactics exercise

EXAM FOCUS

Final 252 Q3 (5 marks) asked you to define three quality attributes and to construct a general availability scenario; Midterm 252 Q5 asked for an availability scenario; Midterm 261 Q5 asked whether heartbeat is a performance tactic (it is availability); Quiz 1 261 matched the parts of usability, testability and security scenarios.

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, L02

NOTES

- 1 Second quality lecture. Lecture 1 gave the tools (scenarios and tactics); this one applies them to each major quality attribute in turn.

SLIDE 2 / 24 Lecture Outlines and quality definitions

Lecture Outlines

➤ Major Quality Attributes

- Definition
- Example with scenario
- Tactics

Quality	Description
Usability	The degree of complexity involved when learning or using the system.
Modifiability	The degree of complexity involved when changing the system to fit current or future needs.
Security	The system's ability to protect and defend its information or information system.
Performance	The system's capacity to accomplish useful work under time and resource constraints.
Reliability	The system's failure rate.
Portability	The degree of complexity involved when adapting the system to other software or hardware environments.
Testability	The degree of complexity involved when verifying and validating the system's required functions.
Availability	The system's uptime.
Interoperability	The system's ability to collaborate with other software or hardware systems.

NOTES

- 1 Each quality gets a **definition**, an **example scenario**, and its **tactics**.
- 2 The table repeats Chapter 2's one-line descriptions: usability (complexity of learning or using), modifiability (complexity of changing), security (protecting information), performance (useful work under time and resource limits), reliability (failure rate), portability, testability, availability (uptime), interoperability.

SLIDE 3 / 24 Major Quality Attributes

Major Quality Attributes

- **Availability:** is concerned with system failure and duration of system failures. (System failure means, when the system does not provide the service for which it was intended)
- **Modifiability:** is about the cost of change, both in time and money.
- **Performance:** is about timeliness. Events occur and the system must respond in a timely fashion.
- **Security:** is the ability of the system to prevent or resist unauthorized access while providing access to legitimate users. An attack is an attempt to breach security.
- **Testability:** refers to the ease with which the software can be made to demonstrate its faults. To be testable the system must control inputs and be able to observe outputs.
- **Usability:** is how easy it is for the user to accomplish tasks and what support the system provides for the user to accomplish this.

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

3

NOTES

- 1 **Availability** is concerned with system failure and the **duration** of failures. A failure means the system does not provide the service it was intended for.
- 2 **Modifiability** is about the **cost of change**, in time and money.
- 3 **Performance** is about **timeliness**: events occur and the system must respond to them in time.
- 4 **Security** is the ability to prevent or resist **unauthorised access while still serving legitimate users**. An attack is an attempt to breach security.
- 5 **Testability** is how easily the software can be made to **demonstrate its faults**. To be testable, a system must let testers control its inputs and observe its outputs.
- 6 **Usability** is how easy it is for users to accomplish tasks, and what support the system gives them to do so.

Past exam · Final 252 · Q3-1 · 3 marks

Q1

List and define three of the major quality attributes of software architecture.

Answer: Any three, one mark each: **Availability:** the system is ready to carry out its task when needed; concerned with failures and their duration. **Performance:** timeliness, how quickly the system responds to events (latency, throughput). **Security:** the ability to resist unauthorised access while still serving legitimate users. **Modifiability:** the cost in time and money of making a change. **Usability:** how easy the system is to learn and use. **Testability:** how easily the software can be made to show its faults through testing.

Prediction

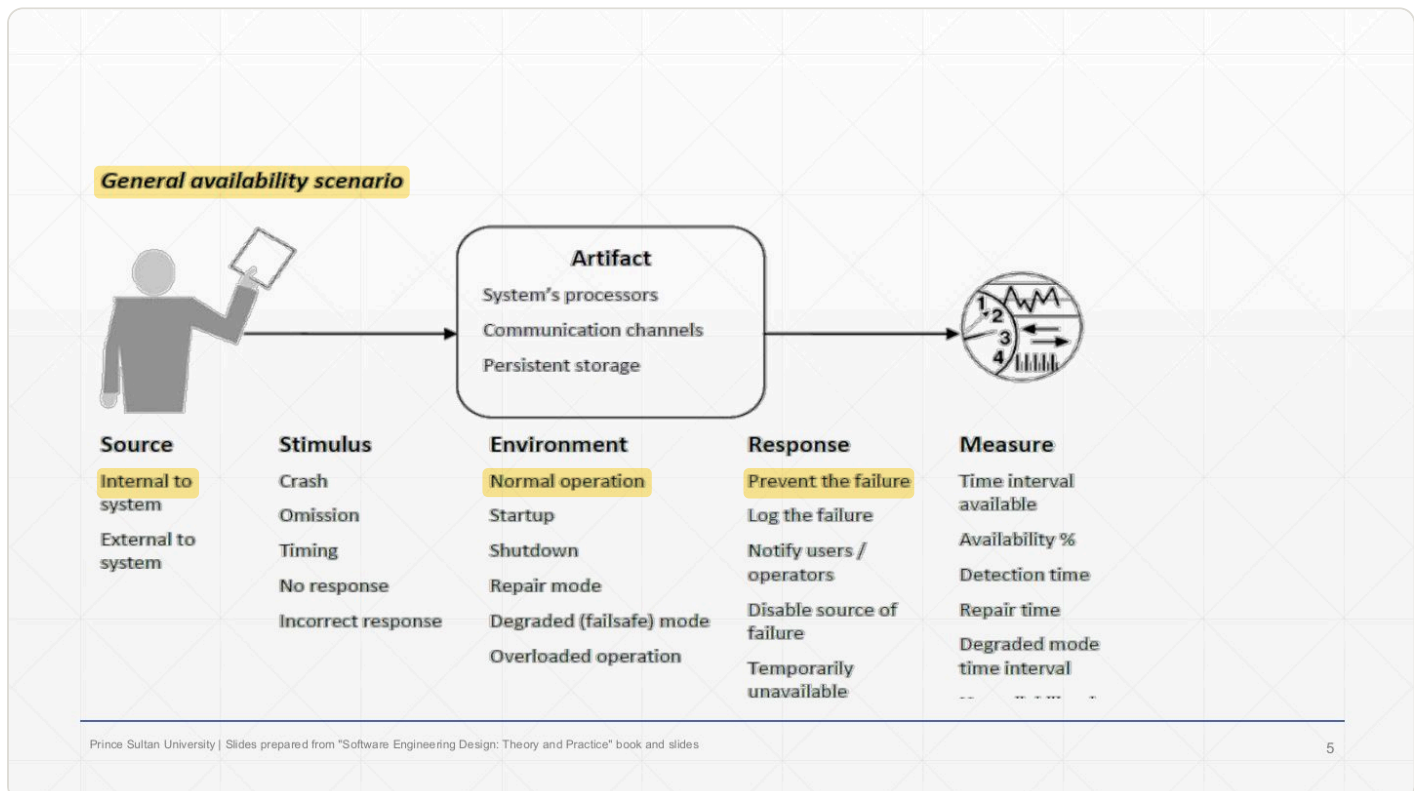
Q2

Which quality attribute is about the cost of change in time and money?

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

Answer: A. Modifiability

SLIDE 4 / 24 Availability: general scenario



NOTES

- 1 The **general availability scenario** lists every possible value of each part. Pick one value per column to write a concrete scenario.

- 2 **Source:** internal or external to the system. **Stimulus** (the fault): crash, omission, timing, no response, incorrect response. **Artifact:** processors, communication channels, persistent storage.
- 3 **Environment:** normal operation, start-up, shutdown, repair mode, degraded (fail-safe) mode, overloaded. **Response:** prevent the failure, log it, notify users or operators, disable the source, or be temporarily unavailable.
- 4 **Measure:** time interval available, availability %, detection time, repair time, degraded-mode time interval, unavailability time.

EXAM ANGLE

Past exam · Midterm 252 · Q5d · 1 mark

Q1

Trip planning system (blackboard design): write an availability quality attribute scenario.

Answer: **Source:** heartbeat monitor (internal to the system). **Stimulus:** the Trip Planning Service (or a worker) becomes unresponsive or crashes. **Artifact:** the Trip Planning Service process. **Environment:** normal operation. **Response:** detect the failure, inform the operator, fail over to a backup instance and continue to operate. **Response measure:** no downtime (or recovers within 5 seconds, 99.9% availability).

SLIDE 5 / 24 Availability Scenario Example

Extra:

Response:

• *Example on the response:* If a server becomes unresponsive, the response might involve:

- Switching to a backup server.
- Reallocating resources.
- Or other corrective actions to ensure continued service.

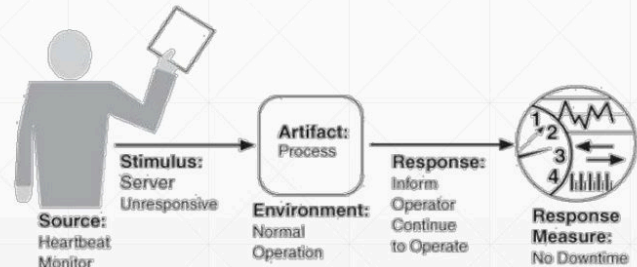
• **Response Measure:** It is the criteria used to evaluate the effectiveness of the response. Also, we need it to determine how well the response addressed the problem, **focusing on the outcome and impact on the system's overall operation.**

• *Example on the response measure:* For the server unresponsiveness issue, a response measure could be:

- The time taken to restore service.
- Or the system's ability to continue operating without downtime.

Availability – Scenario Example

Sample availability scenario



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 as solution (e.g., log error, retry, deny access).
6. **Response Measure** → How success is judged (e.g., response time < 2 sec, 99% uptime).

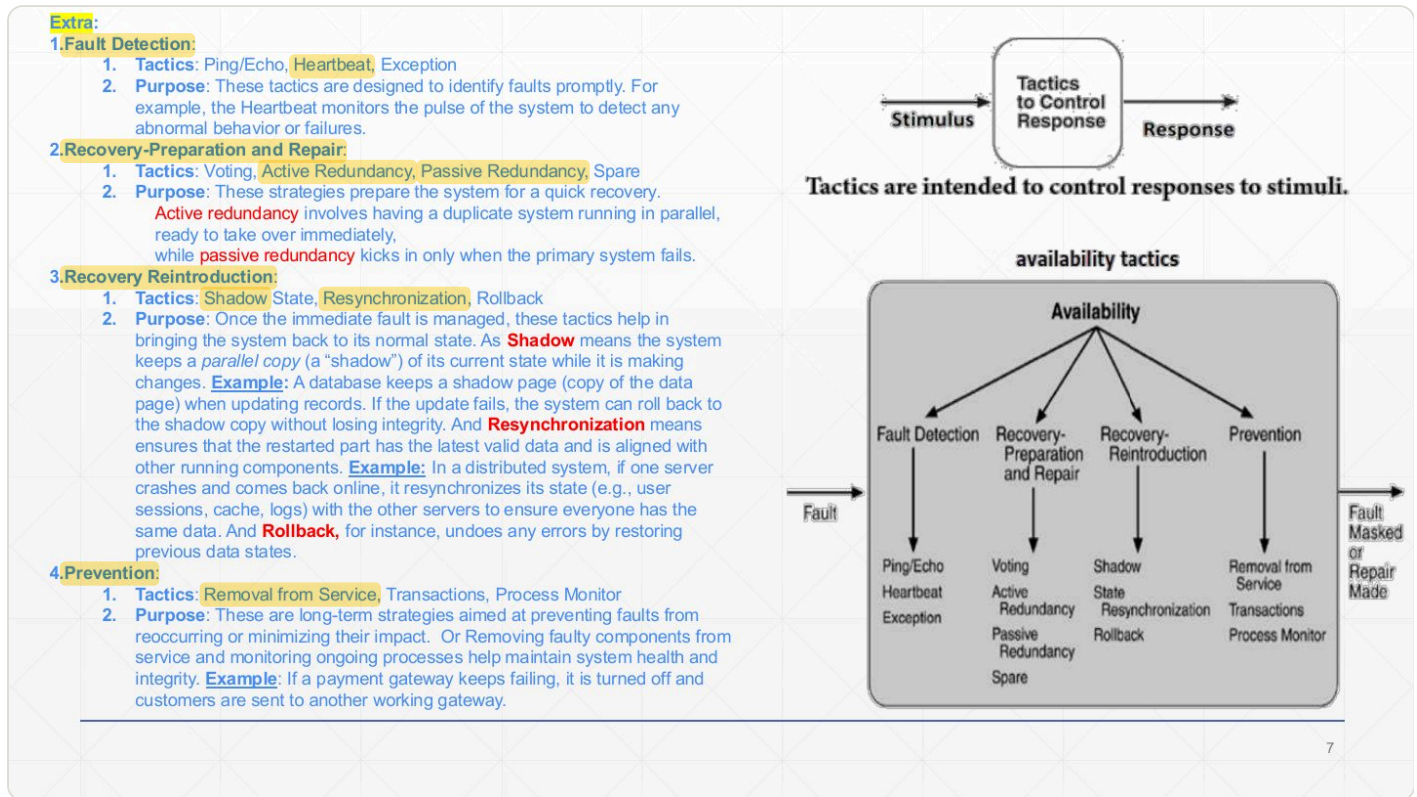
6

NOTES

- 1 A concrete availability **response**: if a server becomes unresponsive, switch to a backup server, re-allocate resources, or take other corrective action to keep the service running.
- 2 The **response measure** judges how well the response worked, focusing on its effect on the whole system: the time taken to restore service, or whether the system kept operating with no downtime.

- 3 The right-hand box repeats the six parts with examples: source (user, attacker, system), stimulus (request, failure, attack), environment (under load, during maintenance), artifact (database, server, UI), response (log, retry, deny access), measure (under 2 s, 99% uptime).

SLIDE 6 / 24 Availability – tactics



NOTES

- 1 Fault detection:** *ping/echo* (send a ping, expect an echo), *heartbeat* (a component sends a regular "I'm alive" signal; a missing beat means a fault), *exceptions* (catch the error when it happens). The goal is to find faults quickly.
- 2 Recovery – preparation and repair:** *voting* (several components compute the result and the majority wins), *active redundancy* (a duplicate runs in parallel, ready to take over instantly), *passive redundancy* (the backup starts only when the primary fails), *spare* (a standby that must be configured before use).
- 3 Recovery – reintroduction** brings a repaired part back: *shadow* (run it alongside the active one, or keep a shadow copy of the state, e.g. a database shadow page to fall back to), *state resynchronization* (bring its data up to date with the others, e.g. sessions and caches after a server restarts), *rollback* (restore a previous good state).
- 4 Prevention** stops faults recurring: *removal from service* (take a failing component out, e.g. switch off a payment gateway that keeps failing and route customers to another), *transactions* (all-or-nothing updates), *process monitor* (watch processes and restart them). The diagram: a fault enters, tactics act, and the fault is masked or repaired.

Past exam · Midterm 261 · Q5 (true/false)

Q1

Heartbeat is a performance tactic used to reduce response time.

Answer: False — Heartbeat is an availability tactic for fault detection; it monitors whether a component is still functioning.

Prediction

Q2

What is the difference between active and passive redundancy?

Answer: In active redundancy the duplicate processes the same inputs in parallel and can take over immediately; in passive redundancy the backup only takes over (after being updated) when the primary fails, so recovery is slower but cheaper.

SLIDE 7 / 24 Extra: Modifiability – Scenario Example

•Extra:

Modifiability – Scenario Example

General modifiability scenario

The diagram illustrates a general modifiability scenario. A person (Source) holds a tablet (Stimulus) and points it towards a box labeled 'Artifact'. The 'Artifact' box contains 'Code', 'Data', 'Interfaces', 'Components', 'Resources', and 'Configurations ...'. An arrow points from the 'Artifact' box to a circular icon containing a waveform and the numbers 1, 2, 3, and 4, representing a measure of change or effort.

Source	Stimulus	Environment	Response	Measure
End-user	Add / delete / modify	Runtime	Make modification	Cost in effort
Developer	functionality,	Compile time	Test modification	Cost in money
System-administrator	quality attribute, capacity or technology	Build time	Deploy modification	Cost in time
		Initiation time		Cost in number, size, complexity of affected artifacts
		Design time		

9

NOTES

- Three modifiability scenarios, differing by **who** wants the change. **End user:** cannot change the code, so they *request* changes ("I want dark mode", "it's too slow with 500 users"). Environment: run time. Artifact: interfaces, resources, configuration. Response: developers or admins implement and redeploy. Measure: cost in money, time, downtime and user satisfaction.
- Developer:** adds, modifies or deletes functionality or qualities (refactoring, improving security). Environment: design, compile or build time. Artifact: code, components, data. Response: implement → test → integrate. Measure: coding and testing effort, complexity of the changed artifacts.

- 3 **System administrator:** changes capacity or technology (scale servers, upgrade the database, adjust configuration). Environment: initialisation or run time. Artifact: configurations, resources, deployment environments. Response: apply → redeploy → test stability. Measure: downtime, effort, risk of errors.

SLIDE 8 / 24 Extra: three modifiability scenarios

Extra Scenarios:		
1. Source: End user	2. Source: Developer	3. Source: system administrator
•Stimulus: An end user requests a new feature or reports a bug. •Artifact: The issue mainly affects user interfaces or functionalities. •Environment: Feedback is considered during updates at runtime. •Response: The software team updates the UI or functionalities based on the feedback. •Measure: Success is measure by user satisfaction and feedback, decreased complaints	•Stimulus: A developer wants to enhance the user interface (UI). •Artifact: This change affects the code that controls the UI. •Environment: The adjustments are made during the design phase. •Response: The developer updates and tests the new UI changes. •Measure: Success is evaluated based on the functionality and speed of implementation, aiming for completion and testing within three hours.	•Stimulus: The system faces increased load or requires enhanced security measures. •Artifact: Focuses on scripts, network setups, and security settings •Environment: Updates are made during live operation or planned downtime to avoid disruption. •Response: Implements hardware upgrades, adjusts network settings, •Measure: Evaluates success through better system reliability, quicker system responses, and improved security outcomes.

NOTES

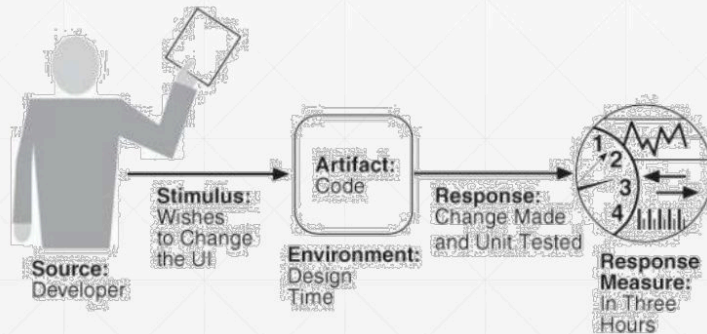
- 1 End user version: a user requests a feature or reports a bug that affects the UI or a function; feedback is handled during updates at run time; the team updates the UI or function; success = user satisfaction and fewer complaints.
- 2 Developer version: a developer wants to enhance the UI; the change affects the UI code; it is made at design time; the developer updates and tests it; success = done and tested within **three hours** (the classic sample scenario on slide 9).
- 3 Administrator version: the system faces more load or needs stronger security; scripts, network set-up and security settings change during live operation or planned downtime; success = better reliability, faster responses, improved security.

Modifiability – Scenario Example

Sample modifiability scenario

More information about general and sample Modifiability scenarios can be found in:

<http://www.ece.ubc.ca/~matei/EECE417/BASS/ch04lev1sec4.html>



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

11

NOTES

- The textbook's **sample modifiability scenario**: source developer; stimulus wishes to change the UI; artifact code; environment design time; response change made and unit tested; response measure in three hours. Memorise this one as the model answer shape for a modifiability scenario.

SLIDE 10 / 24 Modifiability tactics

Modifiability – tactics

- Extra:**
- Localize Changes:** When developers wish to change the UI, employing a strategy like "Generalize Module" or "Semantic Coherence" ensures that the change does not impact other parts of the software, focusing the change only on relevant components.
- Prevention of Ripple Effect:** Example: You are updating the UI of a banking app to make it more user-friendly. any changes to the UI, like adding new visual elements or changing layouts, do not directly impact how account data is processed or retrieved.
- Defer Binding Time** = delaying a decision as late as possible. Usually at runtime, not at design or compile time.

Why we do it

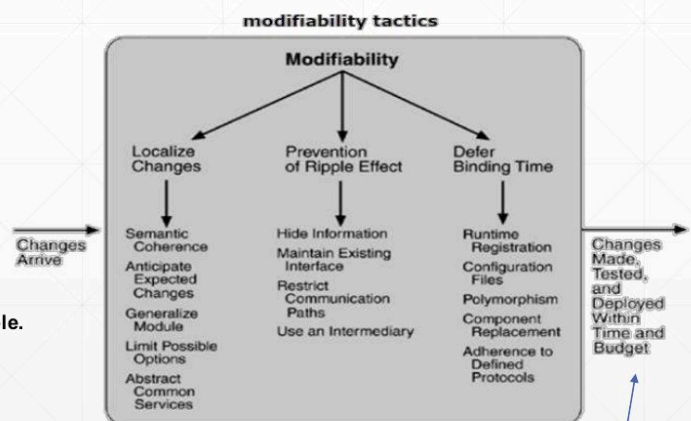
- ☑ Easier to change
- ✗ No need to recompile or rebuild
- ☑ More flexibility

examples

- 🔄 UI Theme
 - ✗ Hard binding: theme written in code
 - ☑ Deferred binding: theme read from a config file at runtime
- 🌐 Language setting
 - ✗ Chosen at compile time
 - ☑ User selects language while app is running

summary

Deferred binding time means postponing decisions to runtime so the system stays flexible and easy to modify.



•Response Measures in Context: Measures of cost in including time, effort, and complexity of implementing changes are important for understanding efficiency.

12

NOTES

- 1 **Localise changes:** assign responsibilities so a likely change touches as few modules as possible. Tactics: semantic coherence (keep related responsibilities together), anticipate expected changes, generalise the module, limit possible options, abstract common services. Changing the UI should only touch UI components.
- 2 **Prevent ripple effects:** stop a change in one module from forcing changes in others. Tactics: hide information, maintain existing interfaces, restrict communication paths, use an intermediary. Updating a banking app's UI must not affect how account data is processed.
- 3 **Defer binding time:** postpone decisions as late as possible, ideally to run time, so changes need no recompile. Tactics: runtime registration, configuration files, polymorphism, component replacement, adherence to defined protocols. Example: a UI theme read from a config file at run time instead of hard-coded; a language the user can switch while the app runs.
- 4 The response measures for modifiability are costs: time, effort and complexity of implementing the change. The diagram's output: changes made, tested and deployed within time and budget.

EXAM ANGLE

Prediction

Q1

Name the three categories of modifiability tactics and give one tactic for each.

Answer: Localise changes (semantic coherence, generalise module); prevent ripple effects (hide information, maintain existing interfaces, use an intermediary); defer binding time (configuration files, runtime registration, polymorphism).

SLIDE 11 / 24 Performance – Scenario Example

Performance – Scenario Example

Extra:

1. **Source:** it is the entity that generates the need for a response or action by the system.

- Internal to the System: Such as a process within the system that requests resources.
- External to the System: Such as a user interaction or an external system's demand.

2. **Stimulus:**

- **Periodic Events:** Regularly occurring events, e.g., daily data backups.
- **Sporadic Events:** Occasional, unpredictable events, e.g., a user starting a large data upload unexpectedly.
- **Bursty Events:** Sudden changes in activity, e.g., many users logging in simultaneously.
- **Stochastic Events:** Random events that have probabilistic distributions, e.g., incoming network traffic.

3. **Artifact:**

- System Component: Any part of the system where the performance is measured or impacted, such as a server, database, or network component.

4. **Environment:**

- **Normal Mode:** Standard operating conditions.
- **Overload Mode:** When the system is operating beyond its usual capacity.
- **Reduced Capacity Mode:** When the system is operating under resource constraints, possibly due to maintenance or partial failure.
- **Emergency Mode:** High-priority operations taking precedence during a crisis.
- **Peak Mode:** Highest expected operational demand.

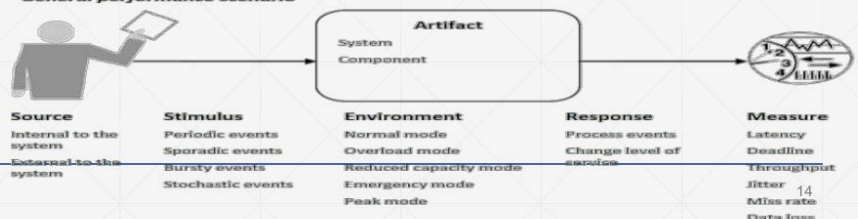
5. **Response:**

- **Process Events:** Handling the incoming tasks or requests.
- **Change Level of Service:** Adjusting the performance levels, possibly by degrading non-essential services to maintain core functionality.

6. **Measure:**

- **Latency:** Delay before data starts moving.
 - o Example: Waiting 2 seconds before a video starts playing.
- **Deadline:** Latest time a task must finish.
 - o Example: A payment transaction must confirm within 5 seconds.
- **Throughput:** How much work/data is done per second.
 - o Example: A server handles 1,000 requests per minute.
- **Jitter:** Inconsistency in response times.
 - o Example: Sometimes a message arrives in 1s, other times in 4s.
- **Miss Rate:** How often deadlines or tasks are missed.
 - o Example: out of 100 frames, 2 frames did not meet the speed requirement → Miss Rate = 2%.
- **Data Loss:** Data that never arrives.
 - o Example: 5 out of 100 packets lost in a video call.

General performance scenario



NOTES

- 1 **Environment** for performance: normal mode, overload (beyond usual capacity), reduced capacity (maintenance or partial failure), emergency (high-priority work first), peak (highest expected demand). The same system may meet its target in one mode and fail in another.
- 2 **Stimulus** = how events arrive: *periodic* (regular, e.g. daily backups), *sporadic* (occasional and unpredictable, e.g. a sudden large upload), *bursty* (sudden spikes, e.g. everyone logging in at once), *stochastic* (random with a probability distribution, e.g. network traffic). Source: internal (a process) or external (users, other systems). Artifact: the component whose performance matters (server, database, network). Response: process the events, or change the level of service (degrade non-essential services to protect core ones).
- 3 **Measures**: *latency* (delay before work starts, e.g. 2 s before a video plays), *deadline* (latest finish time, e.g. payment confirmed within 5 s), *throughput* (work per unit time, e.g. 1,000 requests/minute), *jitter* (variation in response time), *miss rate* (share of deadlines missed, e.g. 2 of 100 frames = 2%), *data loss* (data that never arrives).

EXAM ANGLE

Prediction

Q1

Define latency, throughput and jitter as performance response measures.

Answer: Latency: the delay before a response (or before data starts moving). Throughput: the amount of work done per unit time. Jitter: the variation (inconsistency) in response times.

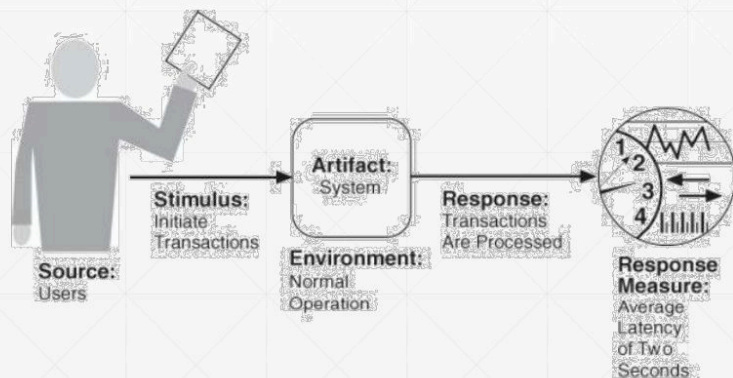
SLIDE 12 / 24 Performance Scenario Example

Performance – Scenario Example

More information about general and sample Performance scenarios can be found in:

<http://www.ece.ubc.ca/~matei/EECE417/BASS/ch04lev1sec4.html>

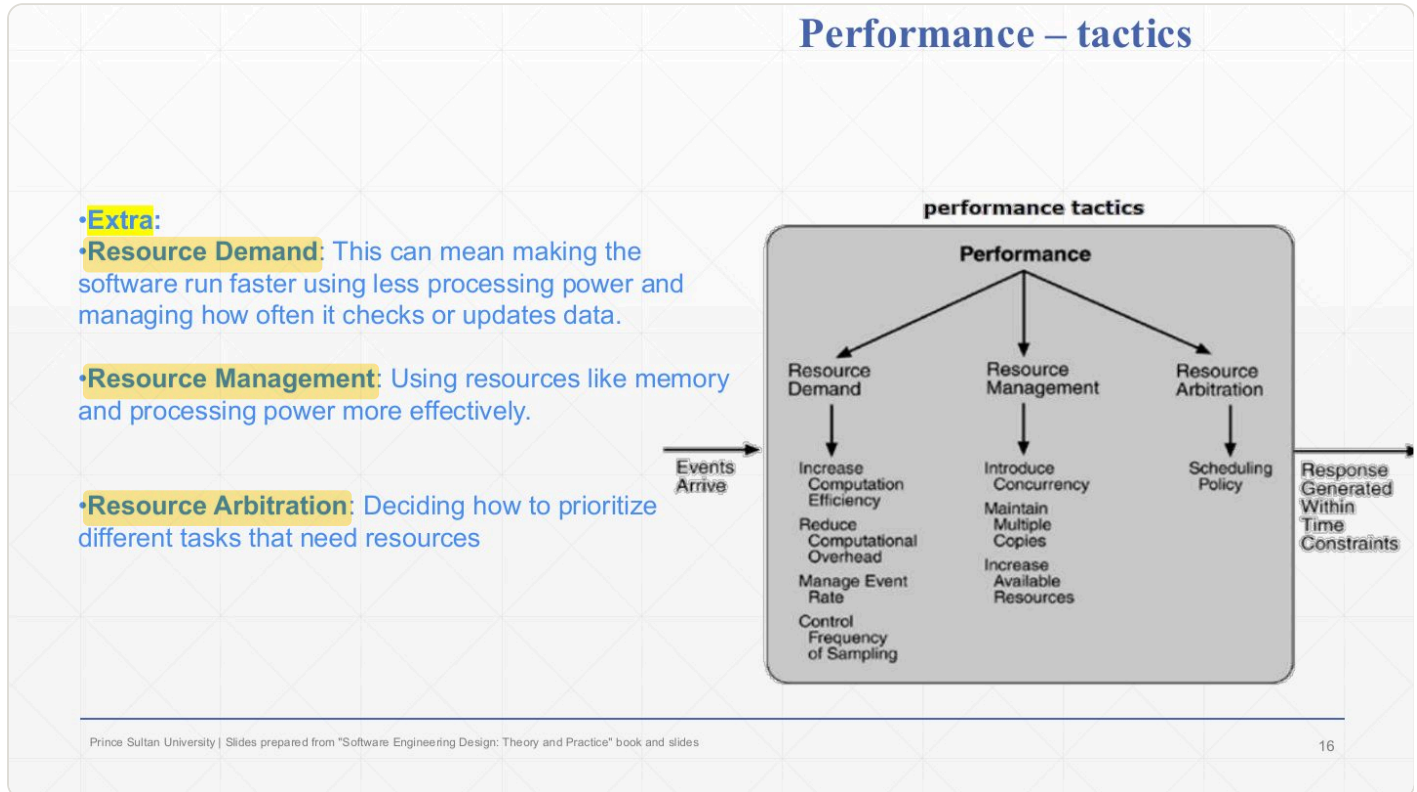
Sample performance scenario



NOTES

- 1 The **sample performance scenario**: source users; stimulus initiate transactions; artifact system; environment normal operation; response transactions are processed; response measure average latency of two seconds. The same scenario as Lecture 1, slide 10.

SLIDE 13 / 24 Performance tactics



NOTES

- 1 **Resource demand**: reduce the work that reaches the system. Tactics: increase computational efficiency (better algorithms), reduce computational overhead, manage the event rate, control the frequency of sampling.
- 2 **Resource management**: make better use of the resources you have. Tactics: introduce concurrency (process requests in parallel), maintain multiple copies (of data or computations, e.g. caches and replicas), increase available resources (faster CPU, more memory).
- 3 **Resource arbitration**: decide which task gets a contended resource first, through a **scheduling policy** (FIFO, priority, deadline-first). Events arrive, the tactics act, and the response is generated within time constraints.

Prediction

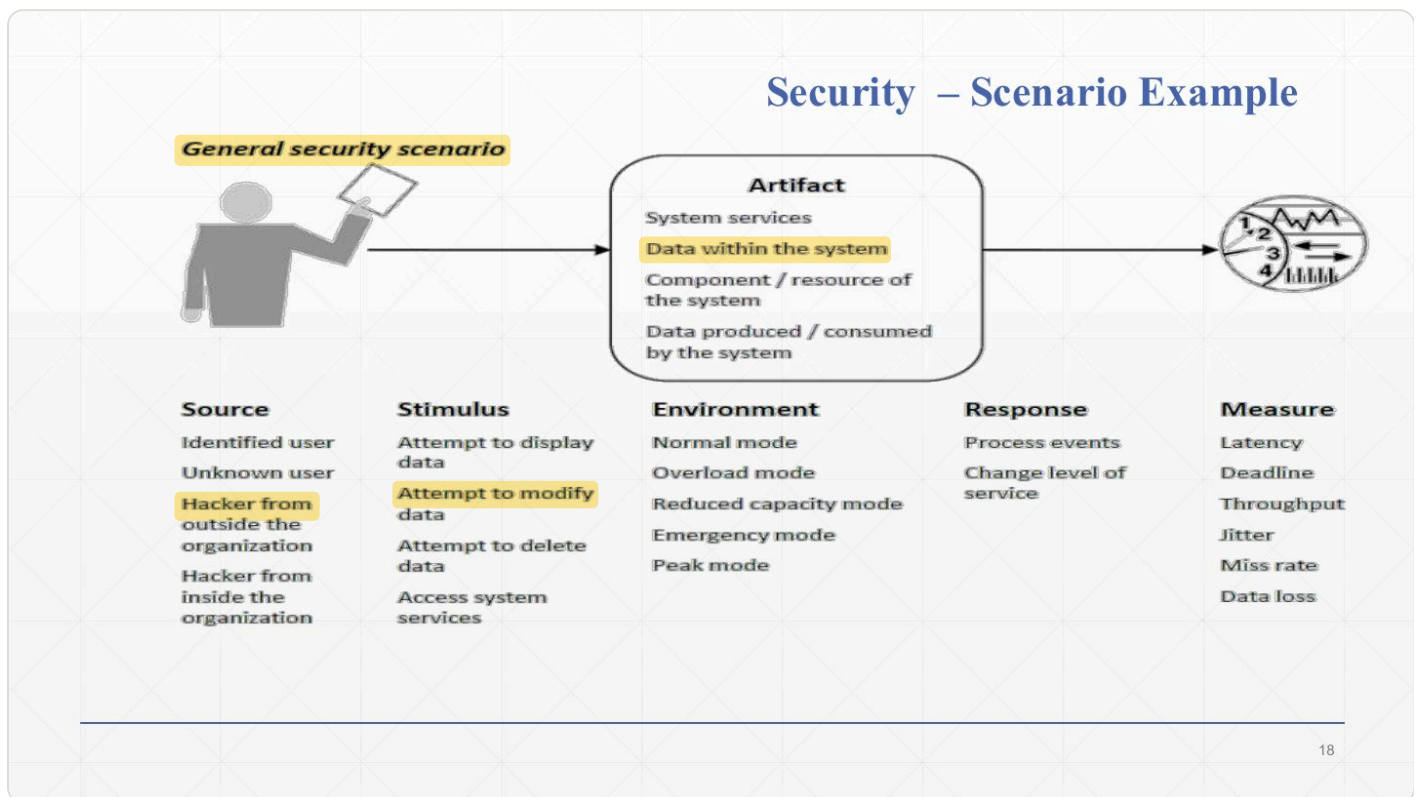
Q1

Choosing which waiting request gets the CPU first is which category of performance tactic?

- A. Resource arbitration
- B. Resource demand
- C. Resource management
- D. Fault detection

Answer: A. Resource arbitration

SLIDE 14 / 24 Security Scenario Example



NOTES

- 1 The **general security scenario**. Read the first three columns: **source** an identified user, an unknown user, or a hacker from outside or inside the organisation.
- 2 **Stimulus**: an attempt to display, modify or delete data, or to access system services. **Artifact**: system services, data within the system, a component or resource, data produced or consumed by the system.
- 3 **Careful**: the Environment, Response and Measure columns on this slide (normal/overload/peak mode; process events; latency, throughput, jitter) are the **performance** scenario's values, copied by mistake. For security, the textbook uses: environment online/offline, connected or not, behind a fire-wall or not; response authenticate, block access, encrypt, log and notify; measure time or effort to break in, probability of detecting the attack, how much data was exposed, time to restore. Use those in an exam.

Prediction

Q1

Give a correct environment, response and response measure for a security scenario (the slide's columns belong to performance).

Answer: Environment: online, connected to the public Internet (or behind a firewall). Response: authenticate the user, block the access, log the attempt and notify an administrator. Response measure: attack detected within 1 minute; account locked after 5 failed attempts; no data exposed; data restored within 1 day.

SLIDE 15 / 24 Extra: Security – Scenario Example

Extra:

Source: A (Dissatisfied person) attempting to modify payroll data from a remote location.

Stimulus: The attempt to modify the pay rate data.

Artifact: The specific data targeted by the attack within the system.

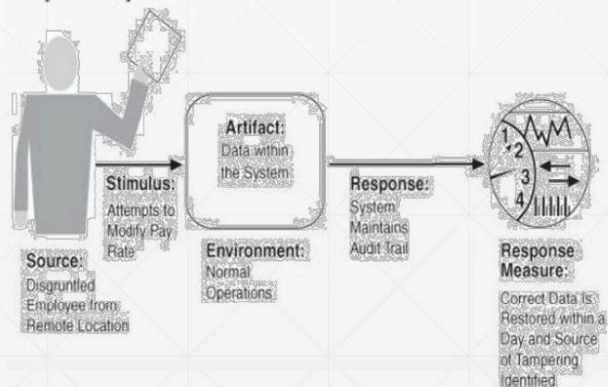
Environment: The system operates under normal conditions when the attack occurs.

Response: The system's response includes maintaining an audit trail, which helps in tracking and recording the sequence of activities.

Response Measure: The effectiveness of the response is measured by how quickly the correct data is restored and the identification of the source of tampering.

Security – Scenario Example

Sample security scenario



19

NOTES

- 1 A concrete security scenario. **Source:** a dissatisfied person (e.g. an employee) working from a remote location. **Stimulus:** the attempt to modify pay-rate data.
- 2 **Artifact:** the targeted payroll data. **Environment:** normal operation.
- 3 **Response:** the system maintains an **audit trail**, a record of who did what and when, so the change can be traced.
- 4 **Response measure:** how quickly the correct data is restored and whether the source of tampering is identified (the textbook version: correct data restored within a day, source identified).

Past exam · Quiz 1 · 261 · Q27 (matching)

Q1

Security scenario: "While the system is connected to the public Internet, an unauthorized external user repeatedly attempts to log in using incorrect credentials to an administrative account. The authentication service rejects the attempts, records the activity, and temporarily locks the account after repeated failures. The system must block the account after 5 failed attempts within 10 minutes and generate an audit record for every attempt." Match the six elements.

Answer: Source: unauthorized external user. Stimulus: repeatedly attempts to log in with incorrect credentials. Artifact: authentication service / administrative account. Environment: system connected to the public Internet. Response: rejects the attempts, records the activity, temporarily locks the account. Response measure: account blocked after 5 failed attempts within 10 minutes; an audit record for every attempt.

SLIDE 16 / 24 Security – tactics

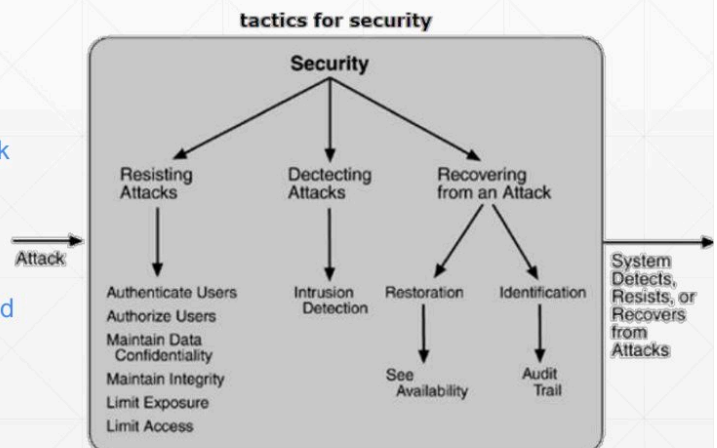
Extra

Resisting Attacks: Methods to prevent unauthorized access or modifications. This includes authenticating and authorizing users, maintaining data confidentiality and integrity, and limiting access and exposure.

Detecting Attacks: Techniques to identify when an attack is occurring, by intrusion detection systems.

Recovering from an Attack: Steps to restore system functionality and data integrity after an attack. This involves restoration, identification of the threat source, and maintaining an audit trail.

Security – tactics



NOTES

- 1 **Resisting attacks** (keep them out): authenticate users (who are you?), authorise users (what may you do?), maintain data confidentiality (encryption), maintain integrity (checksums, hashes), limit exposure (fewer entry points), limit access (firewalls, need-to-know).
- 2 **Detecting attacks:** intrusion detection, comparing traffic and behaviour against known attack patterns.
- 3 **Recovering from an attack:** *restoration* (uses the availability tactics, "see availability" in the tree) and *identification* of the attacker through an **audit trail**. The output: the system detects, resists or recovers from attacks.

Prediction

Q1

Authenticating and authorising users belong to which category of security tactics?

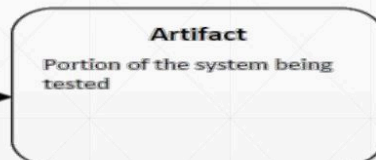
- A. Resisting attacks
- B. Detecting attacks
- C. Recovering from attacks
- D. Resource arbitration

Answer: A. Resisting attacks

SLIDE 17 / 24 Testability Scenario Example

Testability – Scenario Example

General testability scenario



Source

Unit tester
Integration tester
System tester
Acceptance tester
End user
Automated testing tools

Stimulus

Execution of tests
due to completion
of code increment

Environment

Design time
Development time
Compile time
Integration time
Deployment time
Run time

Response

Execute test suite &
capture results
Capture cause of
fault
Control & monitor
state of the system

Measure

Effort to find fault
Effort to achieve
coverage %
Probability of fault
being revealed by
next test
Time to perform
tests
Effort to detect
faults

21

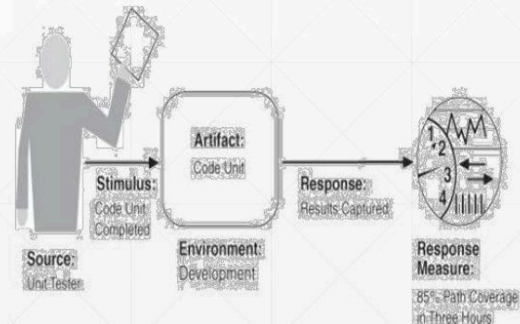
NOTES

- 1 The **general testability scenario**. **Source**: unit, integration, system or acceptance tester, an end user, or automated testing tools.
- 2 **Stimulus**: tests are executed because a code increment is complete. **Artifact**: the portion of the system being tested. **Environment**: design, development, compile, integration, deployment or run time.
- 3 **Response**: execute the test suite and capture results, capture the cause of a fault, control and monitor the system's state. **Measures**: effort to find a fault, effort to reach a coverage %, probability that the next test reveals a fault, time to perform tests, effort to detect faults.

Testability – Scenario Example

- Extra:**
- Source:** The **Unit Tester** is the individual responsible for testing.
- Stimulus:** The **completion of a code unit**, prompting the test.
- Artifact:** The specific code unit that has been completed and is being tested.
- Environment:** **Development environment** where testing typically occurs.
- Response:** Results of the testing are captured.
- Response Measure:** Evaluates the effectiveness of the testing with metrics such as **85% path coverage within three hours**.

Sample testability scenario



NOTES

- 1 A concrete testability scenario. **Source:** the unit tester. **Stimulus:** a code unit is completed. **Artifact:** that code unit.
- 2 **Environment:** the development environment. **Response:** the test results are captured.
- 3 **Response measure:** **85% path coverage within three hours**. Coverage plus a time limit makes the testability goal checkable.

EXAM ANGLE

Past exam · Quiz 1 · 261 · Q26 (matching)

Q1

Testability scenario: "During pre-release system testing, a QA engineer initiates an automated test suite to verify the order-processing component. The system exposes the required interfaces, records internal test results, and reports failed test cases. The complete test suite must finish within 15 minutes and identify the failing component in at least 95% of injected faults." Match the six elements.

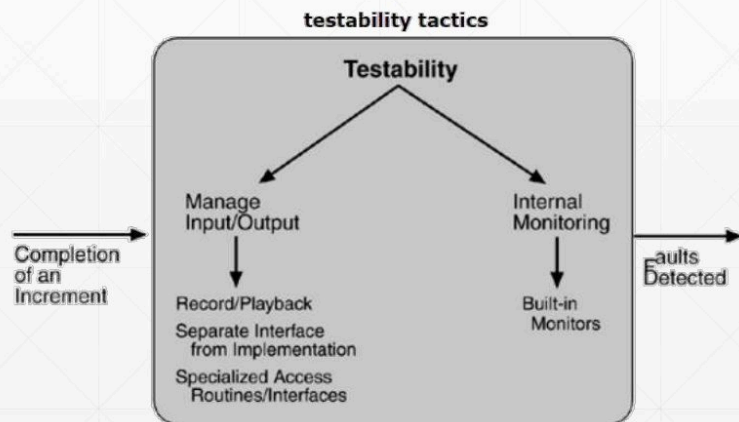
Answer: Source: QA engineer. Stimulus: initiates an automated test suite. Artifact: order-processing component. Environment: pre-release system testing. Response: exposes test interfaces, records results and reports failed test cases. Response measure: suite finishes within 15 minutes and identifies the failing component in at least 95% of injected faults.

Testability – tactics

Extra

This diagram outlines general strategies that enhance the testability of a system:

- **Manage Input/Output:** Includes techniques like record/playback and separating interface from implementation, which simplify testing by allowing testers to manage how data enters and exits the system.
- **Internal Monitoring:** Involves built-in monitors that help in observing the system's internal state, providing valuable data during testing.



NOTES

- 1 **Manage input/output:** control what goes in and observe what comes out. Tactics: *record/playback* (capture real inputs and replay them to reproduce a failure), *separate interface from implementation* (swap in test doubles behind the interface), *specialised access routines/interfaces* (test-only hooks into a component).
- 2 **Internal monitoring:** *built-in monitors* that expose the component's internal state (counters, logs, performance data) during testing. The diagram: an increment is completed, the tactics act, and faults are detected.

EXAM ANGLE

Prediction

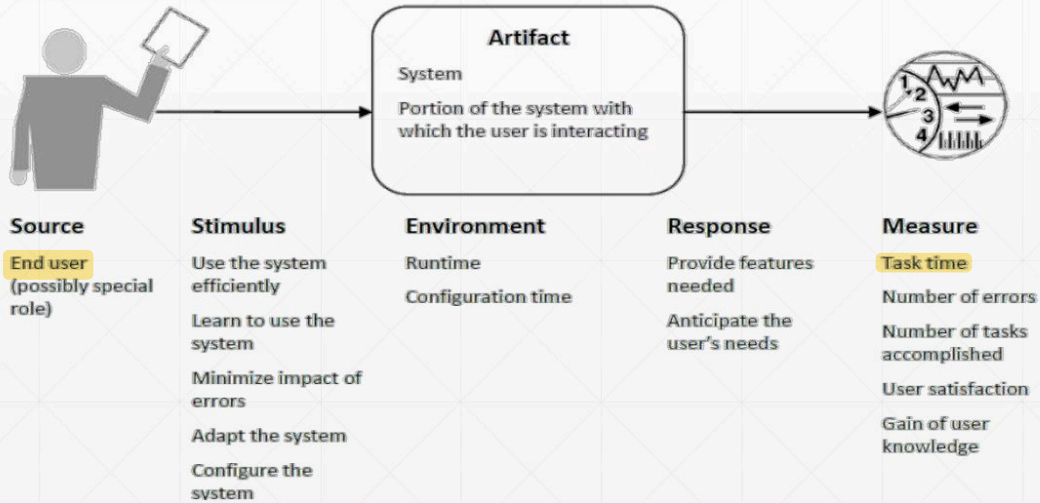
Q1

Give two testability tactics.

Answer: Record/playback and separating interface from implementation (manage input/output); built-in monitors (internal monitoring).

Usability – Scenario Example

General usability scenario



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

24

NOTES

- 1 The **general usability scenario**. **Source**: an end user, possibly in a special role. **Stimulus**: what the user wants: use the system efficiently, learn it, minimise the impact of errors, adapt it, configure it.
- 2 **Artifact**: the system, or the part the user is interacting with. **Environment**: run time or configuration time. **Response**: provide the features needed and anticipate the user's needs.
- 3 **Measures**: task time, number of errors, number of tasks accomplished, user satisfaction, gain of user knowledge.

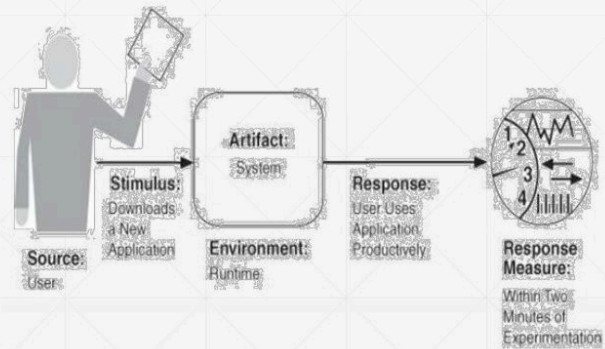
Usability – Scenario Example

Extra:

This diagram represents a usability scenario where a user downloads and begins using a new application. The key elements are:

1. **Source:** Users who initiate the test by downloading the app.
2. **Stimulus:** The action of downloading and starting the app.
3. **Artifact:** The overall system that supports and runs the app.
4. **Environment:** Occurs in real-time during application use.
5. **Response:** The user is able to use the app effectively almost immediately.
6. **Response Measure:** Success is measured by the user achieving productivity within two minutes of starting the app, indicating high usability.

Sample usability scenario



NOTES

- 1 A concrete usability scenario. **Source:** users. **Stimulus:** downloading and starting a new app. **Artifact:** the system that runs the app.
- 2 **Environment:** at run time, while the app is used. **Response:** the user can use the app effectively almost immediately.
- 3 **Response measure:** the user is productive **within two minutes** of starting the app. That number is what makes "easy to learn" testable.

EXAM ANGLE

Past exam · Quiz 1 · 261 · Q23 (matching)

Q1

Usability scenario: "A first-time student user uses the university Learning Management System during normal operation and attempts to submit an assignment for the first time. The system provides clear instructions, guides the student through selecting and uploading the file, and displays a confirmation message after successful submission. The student should complete the task within 2 minutes, without external help, and with no more than one input error." Match the six elements.

Answer: Source: first-time student user. Stimulus: attempts to submit an assignment for the first time. Artifact: assignment submission interface / LMS. Environment: normal operation. Response: provides instructions, guides the file upload and confirms success. Response measure: task completed within 2 minutes, without help, with at most one input error.

Extra:**1. Separate User Interface**

Explanation: Keeping the user interface distinct from system operations.

Example: A blog platform lets you customize your blog's design through a visual editor without needing to touch the underlying code.

2. Support User Initiative

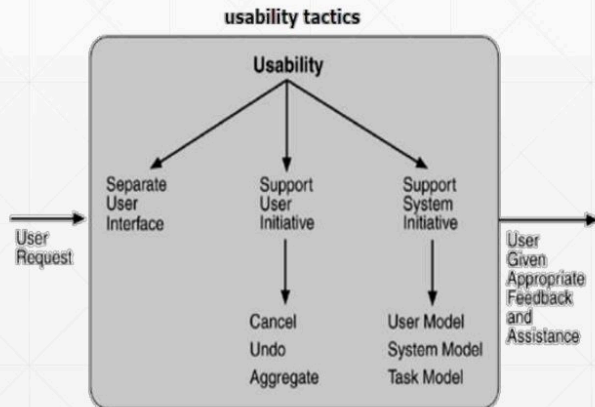
Explanation: Tools that help users manage their actions.

- **Cancel:** In an online shopping cart, users can remove items before checkout.
- **Undo:** In a document editor, users can revert changes by clicking the "undo" button.
- **Aggregate:** A fitness app compiles weekly activity into a summary report for easy review.

3. Support System Initiative

Explanation: The system anticipates and reacts to user needs based on user behavior and tasks.

- **User Model:** A streaming service recommends movies based on your viewing history.
- **System Model:** An email client automatically syncs your most important emails first.
- **Task Model:** A task management app sends reminders for deadlines based on project timelines.

Usability – tactics

26

NOTES

- 1 Separate the user interface** from the system's operations, so the UI can change on its own. Example: a blog platform's visual editor changes the design without touching the code (MVC in Chapter 4 does exactly this).
- 2 Support user initiative:** give users tools to manage their own actions: *cancel* (remove items before checkout), *undo* (revert an edit), *aggregate* (combine many items into one, e.g. a weekly activity summary).
- 3 Support system initiative:** the system anticipates the user using models: a *user model* (recommendations from viewing history), a *system model* (sync the most important emails first), a *task model* (deadline reminders from the project timeline).

EXAM ANGLE**Prediction**

Q1

An "Undo" button supports which usability tactic?

- Support user initiative
- Support system initiative
- Separate the user interface
- Resource arbitration

Answer: A. Support user initiative

Summary

➤ Major Quality Attributes

- Definition
- General and sample scenario examples
- Tactics

➤ Next ... Architecture Design Patterns

➤ https://www.youtube.com/watch?time_continue=85&v=rKwSIih_U74&embeds_referring_euri=https%3A%2F%2Fchatgpt.com%2F&source_ve_path=MjM4NTE

NOTES

- 1 For each major quality you should be able to give the definition, write a general and a sample scenario, and name its tactics.
- 2 Next chapter: architecture patterns, ready-made structures that bundle several of these tactics (e.g. MVC separates the UI, a broker hides locations).

SLIDE 24 / 24 **Extra: Question – one tactic per quality**

Extra: Question List one design tactic that you can use to achieve each of the following quality attributes:

Quality Attribute	Tactics
Security	Resisting Attacks, Detecting Attacks, Authenticate users
Testability	Separate interface from implementation
Modifiability	Localize changes, Encapsulation, Hide information
Performance	Concurrency, Increase available resources.

NOTES

- 1 The exercise: give one tactic per quality attribute. Name the tactic, not the quality again.
- 2 Model answers from the slide: **security** resisting or detecting attacks, authenticate users; **testability** separate interface from implementation; **modifiability** localise changes, encapsulation, hide information; **performance** concurrency, increase available resources. Add **availability**: heartbeat or redundancy, and **usability**: undo or cancel.

EXAM ANGLE

Prediction

Q1

List one design tactic for each of: security, testability, modifiability, performance, availability, usability.

Answer: Security: authenticate users. Testability: record/playback or separate interface from implementation. Modifiability: localise changes (semantic coherence) or hide information. Performance: introduce concurrency or increase resources. Availability: heartbeat or active redundancy. Usability: undo/cancel.

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.