CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Lecture 3: Quality Scenarios

Gregory Gay
DIT636/DAT560 - January 26, 2026

When is Software Ready for Release?

When you can argue that it shows *sufficient quality*.

- Requires choosing **quality attributes**.
  - … specifying **measurements** and **thresholds**.
  - … different measurements and thresholds for **different functionality and execution scenarios**.
- Assessed through **Verification and Validation**.

Quality Attributes

- **Dependability**
  - Ability to offer correct functionality, even under unforeseen or unsafe conditions. Encompasses correctness, **reliability**, safety, robustness
- **Availability**
  - Ability to avoid, ignore, and to recover from failures.
- **Performance**
  - Ability to meet timing requirements.
- **Scalability**
  - Ability to maintain reliability and performance as the available resources grow or shrink.

Today's Goals

- We have discussed what "quality" can mean (attributes).
- We have discussed how to measure **reliability**, **availability**, **performance**, and **scalability**.
- Today: How to assess each using **scenarios**.

Assessing Quality

- Quality goals differ between parts of the system.
  - (e.g., user-facing functionality, code components).
- For those parts, required quality may differ:
  - Between situations.
    - (e.g., system under normal use, heavy load, partially down)
  - Between different groups of users.
    - (e.g., new versus experienced users)
- **Scenarios** describe different execution conditions.

Assessing Quality

- For each scenario, set quality **thresholds** (targets).
- Execute the scenario, measure quality, compare to the threshold.
  - If the threshold is met, good!
  - If not, there is a fault in the system.
- Is that enough?

Assessing Quality

- Quality is often non-deterministic.
  - Performance differs every time code is run.
- Quality evolves with the code.
- Scenarios must be executed many times.
  - Thresholds usually set based on averages or probabilities (“95% of the time…”).
  - Absolute thresholds when there must never be a violation.

Writing Scenarios

Scenarios

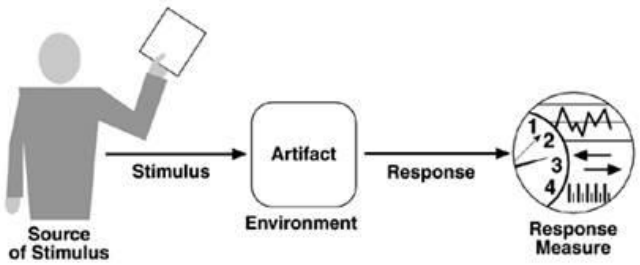
- Description of interaction between external entity and system. Defines:
  - Event that triggers the scenario.
  - Interaction initiated by the external entity.
  - Response required (in terms of quality attribute).
- Similar to use cases or user stories, but examines both **quality and functionality**.

Scenarios

Capture a range of requirements:

- A set of interactions with users to which a system must respond.
- Processing in response to timed events.
- Peak load situations that could occur.
- Regulator demands.
- Failure response.
- A change that a maintainer might make.
- Any situation that the design must handle.

Scenario Format



Overview:

Brief description of the scenario.

System State:

Aspects of internal state that affect quality (e,g., important information stored in the system, current load)

Environment State:

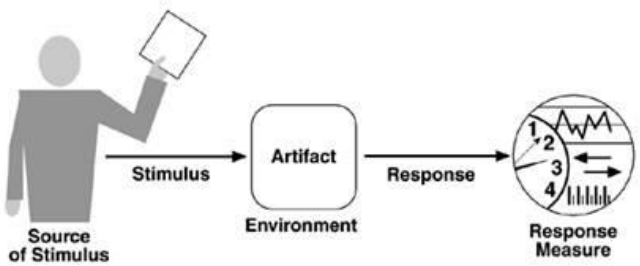
Observations about the environment (e.g., network connection, external system availability).

Scenario Format

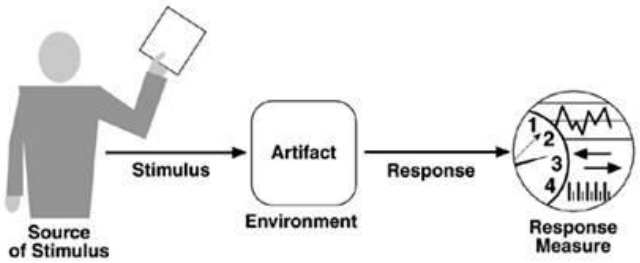
External Stimulus:

Event (input or environmental factor) that triggers a response from the system.

(e.g., user request, infrastructure changes or failures, attacks, expiration of time limit)



Scenario Format



Required System Response:

How does the system respond and meet the quality goal?

(e.g., how should it handle a defined increase in requests?)

Response Measure:

How we judge whether the system meets the quality goal.

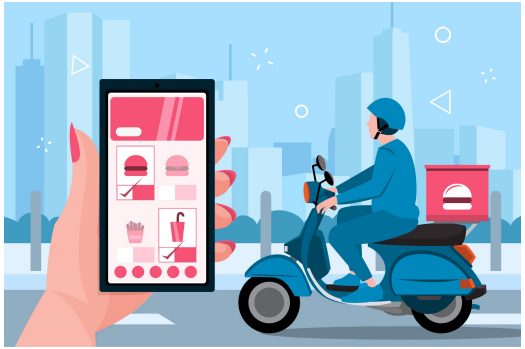
(e.g., throughput, timing, availability)

Response Measures

- Many quality measurements are non-deterministic.
  - May be OK if quality goals are violated *occasionally*.
  - **Most time measures should be probabilistic.**
    - 95% of the time, the response should be N. **(common case)**
    - 99% of the time, the response should be M. **(worst case)**
- If not non-deterministic, give an **absolute** threshold.
  - When safety is affected, measurements should also give a **worst-case limit**.

Running Example: Food Delivery

- We have an online service for ordering food.
- Users can:
  - Search for restaurants.
  - View menus of restaurants.
  - Add items to their cart.
  - Order items for delivery.
- May be used by thousands of concurrent users.



Reliability Example

- **Overview:** A basic scenario where we check that a user can add multiple items to their cart from a restaurant.
- **System State:** A valid user is logged in. Their cart is currently empty. The system is under a normal load (5000 concurrent users).
- **Environment State:** The deployment environment is working correctly. The network connection is adequate.
- **External Stimulus:** The user selects a restaurant. They then add three different items to their cart (quantity of one for each item).
- **Required System Behavior:** When each item is added, it should appear in the cart. The total cost of the order should reflect the new item. The number icon in the top-right of the screen should be incremented by one each time.
- **Response Measure:** The POFOD when executing this scenario should be < 1 failure /1000 requests.

Availability Example

- **Overview:** How the system should respond when BankID fails to respond.
- **System State:** A valid user is logged in. They have added a single item to their cart. The system is under a normal load (5000 concurrent users).
- **Environment State:** The deployment environment is working correctly. The network connection is adequate.
- **External Stimulus:** The user attempts to submit their order. Internally, the logic reaches the step where the user must authenticate in BankID. The call to the BankID API fails to yield a response.
- **Required System Behavior:** After waiting for 30 seconds, the system should stop trying to communicate with the BankID API. The order submission should be aborted. An error message should be sent to the user.
- **Response Measure:** The error message should be displayed within 35 seconds in 95% of executions, and within 45 seconds in 99% of executions.

Is a Scenario a Test Case?

- Scenarios prescribe general execution conditions and goals under those conditions.
  - Describe the “type” of event and response that occurs, not the specific input or output.
- Test cases require concrete, specific input.
- **One scenario** can be used to create **many concrete test cases**.

“Good” Scenarios

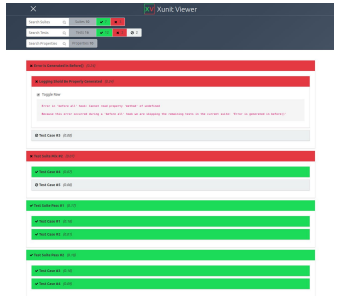
- Credible
  - Describes a realistic scenario.
- Valuable
  - Can be directly used to design and verify system.
- Specific
  - Addresses a single, concrete situation.
- Precise
  - User should be clear about the described situation and response.
- Comprehensible
  - Writing should be unambiguous and free of jargon.

Effective Scenario Use

- Identify a focused scenario set
  - Too many scenarios can be distracting.
  - Prioritize no more than 15-20.
- Use distinct scenarios
  - Avoid redundant scenarios.
  - Consider demands placed on the system.
- Use scenarios early
  - Can focus design activities.

What do we do with Scenarios?

- System Design
- Stakeholder Negotiation
- Exploratory Testing
  - Human experiments with app.
- Formal Test Cases
  - Assign specific input and check response.



Reliability Scenarios

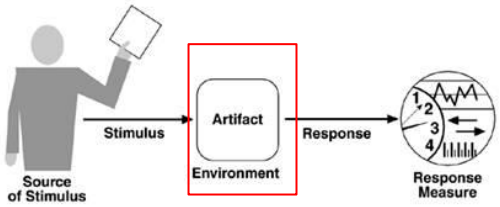
Reliability Scenarios

- **Ability to operate correctly over an observation period** (time or executions).
  - A statistical approximation of correctness.
- Scenarios revolve around one function/code component (or closely-related components).
  - Demonstrates that the function operates correctly.
  - Give context on type of user or system load if it impacts system execution or perceived reliability.

Reliability Metrics

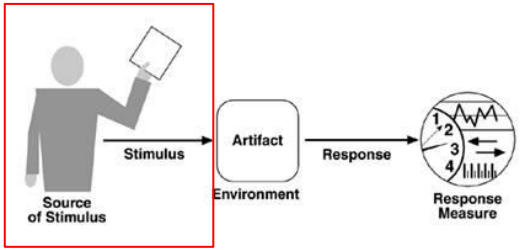
- Time Available: **(uptime) / (total time observed)**
- POFOD: **(# failures) / (# requests)**
- ROCOF: **(# failures) / (period of time)**
- MTBF: **Average time between failures**

Reliability Scenarios



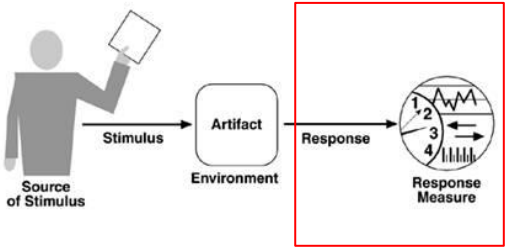
- **Overview:** Highlight the function(s) being used and the context used in. Explain the type of user/load, if relevant.
- **System State:** Data stored in system, past events (multiple failures may leave system vulnerable), load may impact reliability.
- **Environment State:** Available resources may impact reliability (resources, networking).

Reliability Scenarios



- **External Stimulus**:
  - User or external software performs one or more input interactions.
  - State specific interaction(s) performed.
  - If relevant, explain the type of user and reason they would perceive reliability differently.

Reliability Scenarios



- **Required Response**:
  - The functional response of the system.
  - Generally, the normal operation of that function.
- **Reliability Measure**:
  - ROCOF, POFOD, Time Available, MTBF
  - Measures establish general thresholds (i.e., not a single execution, but all executions of this scenario)

Example Reliability Scenario

Requirement: The system shall allow the user to view the menu of a chosen restaurant. This function must be accessible at least 99.99% of the time.

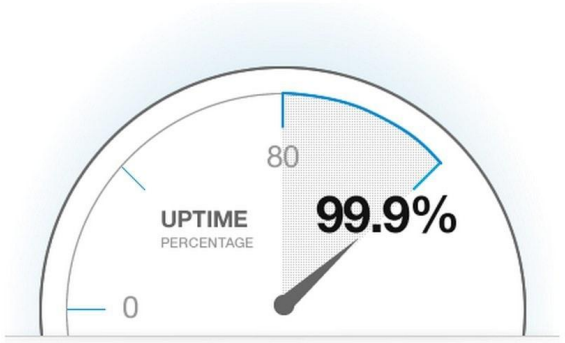
Example Reliability Scenario

- **Overview**: Demonstrates that the user can view the menu of a chosen restaurant under “typical” conditions.
- **System State**: A valid user is logged in. The system is under normal load (5000 concurrent users).
- **Environment State**: The network connection is adequate. The deployment environment is operating under normal conditions.
- **External Stimulus**: The user selects a restaurant, then clicks on “menu”.
- **Required Response**: The menu is displayed to the user.
- **Response Measure**: The time available is 99.999%.
  - (higher than requirement)

Example Reliability Scenario (2)

- **Overview**: Demonstrates that the user can view the menu of a chosen restaurant, even under heavy load.
- **System State**: A valid user is logged in. The system is under heavy load (25000 concurrent users).
- **Environment State**: The network connection is adequate. The load balancer is operating correctly, but available resources are very low.
- **External Stimulus**: The user selects a restaurant, then clicks on “menu”.
- **Required Response**: The menu is displayed to the user.
- **Response Measure**: The time available is 99.99%.
  - (matches the requirement)

Let’s take a break.



Availability Scenarios

Availability Scenarios

- Ability to avoid, ignore, or recover from failure.
- Measure how the system **responds to failure**.
  - How is the failure detected or prevented?
  - What does the system do to return to normal or work around the failure?
  - How long does all of this take?
- **Stimuli: Conditions that trigger the failure**.

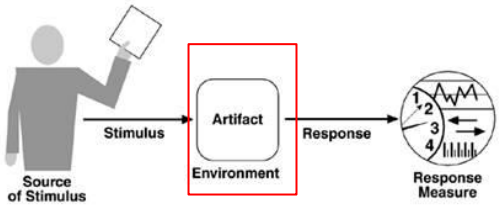
Availability vs Reliability

- Reliability scenarios:
  - Functionality operates without failing.
  - Measures **how often it is allowed to fail**.
- Availability scenarios:
  - How system avoids, handles, or recovers from a failure.
  - Response measures:
    - **Avoiding or ignoring:** reliability measurements
    - **Recovery**: time to recover or resulting “Time Available”

Availability Scenarios

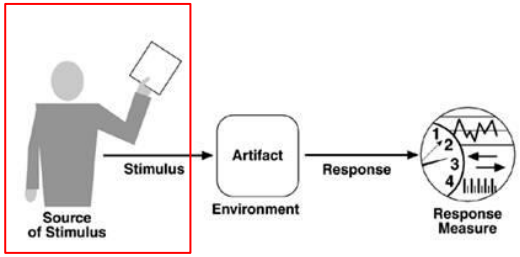
- Distinguish between failures and software's perception of the failure.
  - Do not assume software is omniscient.
- Scenarios tend to deal with:
  - Failure of internal component or external system.
  - Reconfiguration of hardware.
  - Maintenance or reconfiguration of software.

Availability Scenarios



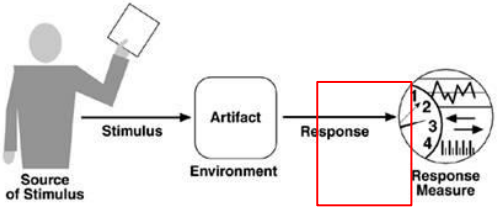
- **Overview:** Be clear about what failed and what needs to be available.
- **System/Environment State:** The state *before failure occurs* may affect the response.
  - If this is the first failure, may choose degradation of response time or functionality.
  - If the system has already failed and is not in normal mode, may choose to shut it down.

Availability Scenarios



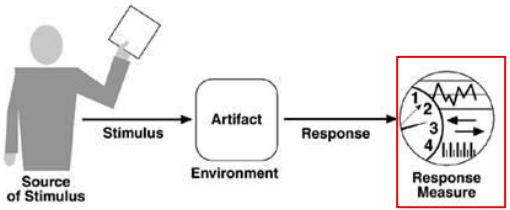
- **External Stimulus:** Differentiate internal/external failure - desired response may differ.
- Stimuli is:
  - An ***omission*** (component fails to respond)
  - A ***crash***
  - ***timing*** (component responds but is early/late)
  - ***incorrect response***

Availability Scenarios



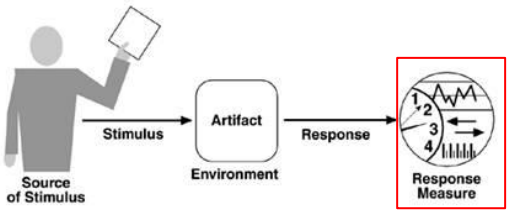
- **Required Response:**
  Failure must be detected and isolated before recovery. Can:
  - Logging the failure
  - Notifying users or systems
  - Taking actions to limit the damage
  - Switching to a degraded mode
  - Shutting down external systems
  - Becoming unavailable during repair.

Availability Scenarios



- **Response Measure:**
  - **Avoid Failure:**
    - Show that reliability is high under this stimulus.
    - POFOD, ROCOF
  - **Ignore Failure:**
    - Show that reliability remains high after the failure.
    - Reliability measurements when failure occurs, compared to “normal” values.
    - May include aspects of “recovery”.

Availability Scenarios



- **Response Measure:**
  - **Recover from Failure:**
    - Show that system can return to normal quickly.
    - Time Available
      - (“Time available must be at least 0.9999”)
    - Time to detect failure
      - (“95% of the time, the failure is detected within 5ms. 99% of the time, within 10ms.”)
    - Time to return to normal
      - (“95% of the time, system is back online within 10 minutes. 99% of the time, within 15 minutes”)

Availability Example

Requirement:

All orders shall be saved in the order database. If an attempt to write an order to the database fails, the failure shall be detected within 30 seconds and reattempted until successful.

Example Availability Scenario

- **Overview**: Demonstrates that the system can recover from a failed attempt to write to the order database (when the database is full).
- **System State**: A valid user is logged in. The system is under normal load (5000 concurrent users).
- **Environment State**: The network connection is adequate. The deployment environment is operating correctly. However, the order database is full.

Example Availability Scenario

- **External Stimulus**: The user places an order under conditions that should normally be successful (i.e., valid payment, restaurant, items). Due to the full database, the request to write the order to the database should fail.
- **Required Response**: After any “write” attempt, the system shall check whether the database contains the order. If not, then the system shall store the order in a temporary queue until it has been successfully written. The order fulfillment shall proceed, using the information stored in this queue. Admins shall be alerted to the failure. The system shall continue to attempt to write to the database until the attempt is successful.
- **Response Measure**: The failure is detected and operation is resumed using the temporary queue within 15 seconds in 95% of executions, and within 30 seconds in 99% of executions.

Performance and Scalability Scenarios

Performance Scenarios

- Begin with events arriving at the system.
  - Responding requires resources to be consumed.
- Arrival pattern for events can be:
  - Periodic (at regular time intervals)
  - Stochastic (events arrive according to a distribution)
  - Sporadic (unknown timing, but known properties)
    - “No more than 600 per minute”
    - “At least 200 ms between arrival of two events”

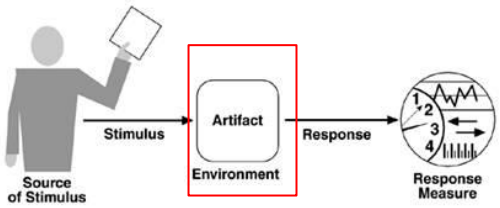
Performance Measurements

- **Latency:** The time between the arrival of the stimulus and the system's response to it.
- **Response Jitter:** Allowed variation in latency.
- **Throughput:** Number of transactions processed in a unit of time.
- **Deadlines in processing:** Points where processing must have reached a particular stage.

Specifying Response Time

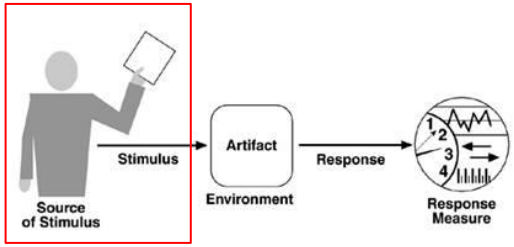
- Response time targets require a defined load.
  - One transaction in 3s is easy if that is the only request.
    - Can you still hit 3s if there are 500 transactions per second?
  - Specify a clearly-defined response time goal.
    - Define when a transaction starts and ends.
- Not all requests take the same amount of time, even with constant load.

Performance Scenarios



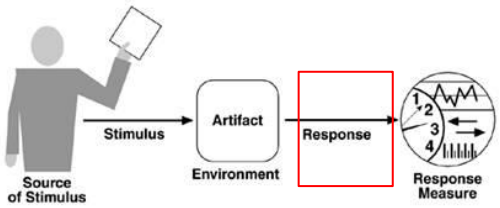
- **Overview:** Description of the scenario.
- **System State:** System can be in various levels of load (normal, emergency, peak load, or overload).
- **Environment State:** Be clear on conditions that can impact performance.
  - Limited resources (disc, memory, CPU)
  - Networking conditions

Performance Scenarios



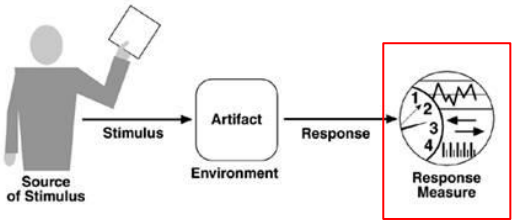
- **External Stimulus:** Stimuli arrive from external or internal sources.
  - The stimuli are event arrivals.
  - Arrival pattern can be periodic, stochastic, or sporadic, characterized by numeric parameters.
  - Be clear on number, duration, concurrency of stimuli.

Performance Scenarios



- **Required System Behavior:**
  - System must process arriving events.
  - May cause change in system
    - (e.g., shift from normal to overload mode).
  - May cause change in environment
    - (e.g., reduction of available memory)

Performance Scenarios



- **Response Measure**:
  - Time to process arriving events (**latency or a deadline**)
  - Variation in latency time (**jitter**)
  - Number of events that can be processed within a time interval (**throughput**)

Performance Example

Requirement:

The system shall display the results of a search query to the user's screen within 5 seconds.

Example Performance Scenario

- **Overview:** Show that search queries can be completed completely under normal conditions.
- **System State**: A valid user is logged in. The system is under normal load (5000 concurrent users).
- **Environment State**: The network connection is adequate. The deployment environment is under normal conditions.
- **External Stimulus**: The user searches for all pizza restaurants within their delivery radius.

Example Performance Scenario

- **Required Response**: The search results are retrieved and displayed to the user. The results for this query should be temporary cached (until the restaurant list or the menu of any restaurant retrieved by the query changes) to reduce the calculation time for other users who issue the same query in the delivery area.
- **Response Measure**: The search results are displayed within 2 seconds in 95% of executions, and within 5 seconds in 99% of executions.

Example Performance Scenario 2

- **Overview:** Show that search queries can be completed completely under heavy load.
- **System State**: The system is under heavy load (25000 concurrent users).
- **Environment State**: The network connection is adequate. The deployment environment is operating correctly, but each server is at or near-capacity.
- **External Stimulus**: 1000 users submit search requests in a 10 second window.

Example Performance Scenario 2

- **Required System Behavior:** The search results are retrieved and displayed to all users. Results are cached for reuse for applicable users. Each server maintains a queue of requests and processes requests as resources become available. The load balancer distributes requests to servers, favoring servers with shorter queues.
- **Response Measure:** All 1000 requests are completed within 10 seconds, in 85% of executions, and within 20 seconds in 99% of executions.

Scalability Scenarios

- **The ability to effectively use available resources**
- Scenarios directly deal with impact of **adding or removing resources**.
- Response measures reflect:
  - Changes to performance.
  - Changes to reliability.
  - Load assigned to existing and new resources.

Example Scalability Scenario

- **Overview:** Addition of new hardware improves order transaction speed.
- **System State:** Before addition of new hardware, 90% of order submissions were completed within 10 seconds, 99% within 15s. System is under normal load (5000 users).
- **Environment State:** An additional server has doubled threads available for processing requests. Environment is otherwise operating normally, with adequate connectivity.

Example Scalability Scenario

- **External Stimulus:** Customer submits a valid order.
- **Required System Behavior:** Order confirmation is displayed, with a list of items purchased, expected delivery time, and total cost of items.
- **Response Measure:** In 90% of requests, submission result is loaded and displayed to user within 5 seconds. In 99% of requests, the result is loaded and displayed to user within 7.5 seconds.

Key Points

- Defining and applying scenarios ensures that desired quality attributes are shown.
- Scenarios define how the system responds to factors that affect quality properties.
- Should include the initial system state and environment state, external stimulus, required system response, and how to assess response.

Next Time

- Testing Fundamentals
- Exercise session: Scenarios

- **Assignment 1 due Feb 8.**
  - **Will be posted after team formation complete.**

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY