CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Lecture 3: Quality - Non-Functional Attributes

Gregory Gay
DIT635 - January 29, 2020

Today's Goals

- Examine non-functional quality properties.
  - **Performance:** Ability to meet timing requirements.
  - **Scalability:** Ability to "grow" the system to process an increasing number of requests.
  - **Availability:** Ability of a system to mask or repair faults such that the cumulative service outage does not exceed a required value over a time interval
  - **Security:** Ability of the software to protect data and information from unauthorized access
- How to assess each using scenarios.

Assessing Performance and Scalability

Performance

- The ability of the software to meet timing requirements.
- Can we characterize the pattern of events arriving and the pattern of responses?
  - Requests served per minute.
  - Variation in output time.
- Driving factor in software design.
  - Often at expense of other quality attributes.
  - **All** systems have performance requirements.

Scalability

- The ability to “grow” the system to process an increasing number of requests.
  - While still meeting performance requirements.
- Horizontal scalability (“scaling out”)
  - Adding more resources to logical units.
    - Adding a cluster of servers.
    - “Elasticity” - can customers to add or remove VMs from a pool?
- Vertical scalability (“scaling up”)
  - Adding more resources to a physical unit.
    - Adding memory to a single computer.

Scalability

- How can we effectively utilize additional resources?
- Effective scalability requires that:
  - Additional resources result in performance improvement.
  - Did not require undue effort to add.
  - Did not disrupt operations.
- Can be thought of as a form of modifiability.
  - The system must be designed to scale (i.e., designed for concurrency).

Performance and Scalability Scenarios

Performance Quality Scenarios

- Measure system performance (not user).
- Begins with an event 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 Quality Scenarios

- Response measurements include:
  - **Latency:** The time between the arrival of the stimulus and the system's response to it.
  - **Response Jitter:** The allowable variation in latency.
  - **Throughput:** Usually number of transactions the system can process in a unit of time.
  - **Deadlines in processing:** Points where processing must have reached a particular stage.
  - **Number of events not processed** because the system was too busy to respond.

Measurements - Latency

- Time it takes to complete an interaction.
- **Responsiveness** measures how quickly the system responds to routine tasks.
  - Key consideration: user productivity.
  - How responsive is the user's device? The system?
  - Measured probabilistically (... 95% of the time)
  - Under a load of 350 update transactions per minute, 90% of "open account" requests should return a reply to the calling program within 10 seconds.

Measurements - Latency

- **Turnaround time** = time to complete larger tasks.
  - Can task be completed in available time?
  - Impact on system while running?
  - Can partial results be produced?
  - Assuming a daily throughput of 850,000 requests, the process should take no longer than 4 hours, including writing results to a database.
  - It must be possible to resynchronize with all monitoring stations and reset database to reflect the current state within 5 minutes.

Measurements - Response Jitter

- Response time is non-deterministic.
  - If this non-determinism can be controlled, this is OK.
  - 10s +- 1s, great! 10s +- 10 minutes, not great.
- **Response jitter** defines how much variation in the latency is allowed.
  - Places boundaries on when a task can be completed.
  - If boundaries are violated, quality is compromised.
  - “All writes to the database must be completed within 120 to 150 ms.”

Measurements - Throughput

- **Throughput** - The workload a system can handle in a defined time period.
  - Shorter the processing time, higher the throughput.
  - As load increases (and throughput rises), response time for individual transactions tends to increase.
    - With 10 concurrent users, request takes 2s.
    - With 100 users, request takes 4s.

Measurements - Throughput

- Possible to end up in situation where throughput goals conflict with response time goals.
  - With 10 users, each can perform 20 request per minute (throughput: 200/m).
  - With 100 users, each can perform 12 per minute (throughput: 1200/m - but at cost to response time).

Measurements - Deadlines

- Some tasks must take place at scheduled times. If these times are missed, the system will fail.
  - In a car, the fuel must ignite when the cylinder is in the correct position.
  - This places a deadline on when the fuel must ignite.
- Deadlines can be used to place boundaries on when events must complete.

Measurements - Missed Events

- If the system is busy, input may be ignored.
  - Or, queued until too late to matter.
- We can track how many input events are ignored because the system is too slow to respond.
  - Set upper bound on how many events can be missed in a defined timeframe.

Performance Quality Scenarios

- For real-time systems (i.e., embedded devices), measurements should be absolute.
  - Look at worst-case scenario.
- For non-real-time systems, measurements should be probabilistic.
  - 95% of the time, the response should be N.
  - 99% of the time, the response should be M.

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?
  - Requirements must specify context and a clearly-defined response time goal.
  - Also define when a transaction starts and ends.
- Why probabilistic definitions are important.
  - Not all requests take the same amount of time, even with constant load.

Generic Performance Scenario

- **Overview:** Description of the scenario.
- **System/environment state:** System can be in various operational modes, such as normal, emergency, peak load, or overload.
- **External Stimulus:** Stimuli arrive from external or internal sources. The stimuli are event arrivals. The arrival pattern can be periodic, stochastic, or sporadic, characterized by numeric parameters.

Generic Performance Scenario

- **Required system behavior:** The system must process arriving events. This may cause a change in the system environment (e.g., from normal to overload mode).
- **Response measure:** The response measures are the time it takes to process the arriving events (latency or a deadline), the variation in this time (jitter), the number of events that can be processed within a particular time interval (throughput), or a characterization of the events that cannot be processed (miss rate).

Bad Performance Scenario

- **Overview:** How the server handles concurrent requests with graceful response times.
- **System/environment state:** Application is packaged and deployed on the server. The server process is up and ready to serve requests.
- **External Stimulus:** Concurrent requests arrive in high volume.
- **Required system behavior:** Server spawns new threads and handle each request concurrently based on resources configured (like available memory, CPU speed etc.)
- **Response measure:** Server successfully handles all requests.

Good Performance Scenario

- **Overview:** Check system responsiveness for adding items to shopping cart under normal operating conditions.
- **System/environment state:** Normal load is defined as deployment environment with no failures and less than 20 customer requests per second. System is communicating over good internet connection to client.
- **External Stimulus:** Customer adds product to shopping cart.
- **Required system behavior:** Web page refreshes. Icon on right side of web page displays last item added to cart. If item is out of stock, cart icon has exclamation point overlay on top of cart icon.
- **Response measure:** In 95% of requests, web page is loaded and displayed to user within 1 second. In 99.9% of requests, web page is loaded and displayed to user within 5 seconds.

Scalability Scenarios

- Ability to address more requests is often part of **performance** scenarios.
- Scenarios assessing scalability directly (*the ability to adjust available resources to the system*) deal with the impact of adding or removing resources.
- Response measures reflect:
  - Changes to performance.
  - Changes to availability.
  - Load assigned to existing and new resources.

Example Scalability Scenario

- **Overview:** Addition of new hardware improves credit card transaction speed.
- **System/environment state:** Before addition of new hardware, 95% of credit card transactions were completed within 10 seconds, 99.9% within 15s. Additional server has doubled threads available for processing requests. System is under normal load, with normal connectivity.
- **External Stimulus:** Customer completes a purchase.

Example Scalability Scenario

- **Required system behavior:** Order confirmation is displayed, with a list of items purchased, expected arrival date, and total cost of items.
- **Response measure:** In 95% of requests, web page is loaded and displayed to user within 5 second. In 99.9% of requests, web page is loaded and displayed to user within 7.5 seconds.

Key Points

- Performance is about management of resources in the face of demand to achieve acceptable timing.
  - Performance usually measured in terms of throughput and latency.
- Performance can be improved by reducing demand or by managing resources.
  - Reducing demand will have the side effect of reducing fidelity or missing some requests.
  - Managing resources can be done through scheduling, replication, or just increasing resources.

Key Points

- Scalability is the ability to “grow” the system to process an increasing number of requests.
  - While still meeting performance requirements.
  - Assessed as part of performance.
- How can we effectively utilize additional resources?
- Effective scalability requires:
  - New resources result in a performance improvement.
  - New resources did not require undue effort to add.
  - New resources did not disrupt operations.

Let’s take a break.

Assessing Availability

Availability

- Is the software there and ready to carry out its task when you need it to be?
  - Encompasses **reliability** and **repair**.
    - Does the system tend to show correct behavior?
    - Can the system recover from an error?
- **Availability** refers to the ability of a system to mask or repair faults such that the cumulative service outage does not exceed a required value over a time interval.

Availability

- Closely related to security and performance.
  - Security: DDOS attack can make system unavailable.
  - Performance: Has the system failed, or is it recovering or limiting the damage from a hazard?
- Availability is about minimizing outage time by mitigating the effect of faults.
  - A **failure** is a visible deviation from expected behavior (crash, incorrect output).
  - Failures caused by **faults** - mistakes in the source code.

Availability

- Achieving availability requires understanding the nature of the failures that can arise.
- Faults/failures can be prevented, tolerated, removed, or forecasted.
  - How are faults detected?
  - How frequently do failures occur?
  - What happens when a failure occurs?
  - How long can the system be out of operation?
  - When can faults or failures occur safely?
  - Can faults or failures be prevented?
  - What notifications are required when failure occurs?

Measuring Availability

- Time to repair is the time until the failure is no longer observable.
- “Observability” can be hard to define.
  - Stuxnet caused problems for months before being noticed. How does that impact availability?
- Software can remain partially available more easily than hardware.
- If code containing a fault is executed, but the system is able to recover, there was no failure.

Measuring Availability

- The availability of a system reflects its ability to deliver services when available (uptime/total time).
  - Takes repair and restart time into account.
  - Scheduled downtime often does not count.
- Availability of 0.9999 means the system is available 99.99% of the time.
  - 0.9 = down for 144 minutes a day, 0.99 = down for 14.4 minutes, 0.999 = down for 84 seconds, 0.9999 = down for 8.4 seconds.

Probabilistic Availability

- (alternate definition)
- Availability is the probability that the system will provide a service within required bounds over a specified time interval.
  - **Availability = MTBF / (MTBF + MTTR)**
    - MTBF: Mean time between failures.
    - MTTR: Mean time to repair

Availability Scenarios

Availability Quality Scenarios

- The ability of the system to mask or repair faults such that the outage period does not exceed a required value over a time period.
- Measure how the system responds to failure.
  - When the system breaks, how long does it take to resume normal operation?
- Stimuli should always be a failure.

Availability Quality Scenarios

- Response measures should always include a measure of availability:
  - availability percentage (must be at least 0.9999)
  - time to detect or repair fault
  - time system in degraded mode
    - (95% of the time, must be back online within five minutes)
  - Can be either explicit (per-execution, repeat multiple times) or probabilistic (averaged over all repeats).
    - Often done probabilistically.

Availability Quality Scenarios

- Scenarios must distinguish physical failures in the system and the software's perception of the failure.
  - Do not assume software is omniscient.
- Scenarios tend to deal with:
  - Failure of a physical component or external system.
  - Reconfiguration of the physical system.
  - Maintenance or reconfiguration of the software.

Generic Availability Scenario

- **Overview:** Description of the scenario.
- **System/environment state:** The state of the system when the fault or failure occurs may also affect the desired system response.
  - If the system has already failed and is not in normal mode, it may be desirable to shut it down.
  - If this is the first failure, degradation of response time or functions may be preferred.

Generic Availability Scenario

- **External Stimulus:** Differentiate between internal and external origins of failure because desired system response may be different.
- Stimuli is:
  - An *omission* (a component fails to respond to an input),
  - A *crash* (component repeatedly suffers omission faults)
  - *timing* (component responds but response is early/late)
  - *response* (a component responds with incorrect value).

Generic Availability Scenario

- **Required system behavior:** There are a number of possible reactions to a failure. Fault must be detected and isolated before any other response is possible. After the fault is detected, the system must recover. Actions include:
  - Logging the failure
  - Notifying selected users or other systems
  - Taking actions to limit the damage caused by the fault
  - Switching to a degraded mode with less capacity or less function
  - Shutting down external systems, or becoming unavailable during repair.

Generic Availability Scenario

- **Response measure:**
  - Can specify an availability percentage
  - Can specify a time to detect the fault, time to repair the fault, times or time intervals where system must be available, or duration for which the system must be available.

Bad Availability Scenario

- **Overview:** How the server manages multiple applications with desired isolation.
- **System/environment state:** Multiple applications (unrelated) are deployed on the server. The server is up and running.
- **External Stimulus:** User(s) establishes session with each of these applications.
- **Required system behavior:** Server deploys each application in its own context which can be configured to share or not share any application specific data between them.
- **Response measure:** Applications are isolated.

Good Availability Scenario

- **Overview:** How the system handles additional beer taps being added to the dispensing system.
- **System/environment state:** The system is operating normally.
- **External Stimulus:** A user powers up a new Kegboard on the network with six additional taps.
- **Required system behavior:** The kegboards send init messages to the central Kegbot server. The server interrogates the kegboards and adds the additional taps to the inventory of taps. The system continues to service the existing taps without interruption.
- **Response measure:** There is no interruption of service to existing taps. Within 1 second, the new kegboard is added to the administrative interface on the KegBot web server for administraton configuration.

Availability Scenario 2

- **Overview:** One of the client-facing web servers fails during transmission of client page update.
- **System/environment state:** System is working correctly under normal load. Customer has generated a “add item to shopping cart” post, which was routed to web server <X> in transaction pool.
- **External Stimulus:** Web server <X> crashes during response generation.

Availability Scenario 2

- **Required system behavior:** Response page may be corrupted on client browser. Load balancer component no longer receives heartbeat message from web server and so removes it from the pool of available servers after 2s of missed messages, or upon next request sent to the server. Load balancer will remove the server from the pool of available servers. From client's perspective, a page reload will be automatically routed to alternate server by load balancer and page will be correctly displayed.
- **Response measure:** On client-side page refresh, client state and display contains state after last transaction. Time for re-routed refresh is equivalent to "standard" refresh (<1 second 95% of the time).

Key Points

- Availability is the ability of the system to be available for use, especially after a failure.
- Failures must be recognized or prevented.
  - System response can range from “ignore it” to “keep on going as if it didn’t occur”.

Let’s take a break.

Assessing Security

“Your personal identity isn’t worth quite as much as it used to be - at least to thieves willing to swipe it. According to experts who monitor such markets, the value of stolen credit card data may range from $3 to as little as 40 cents.

That’s down tenfold from a decade ago - even though the cost to an individual who has a credit card stolen can soar into the hundreds of dollars.”

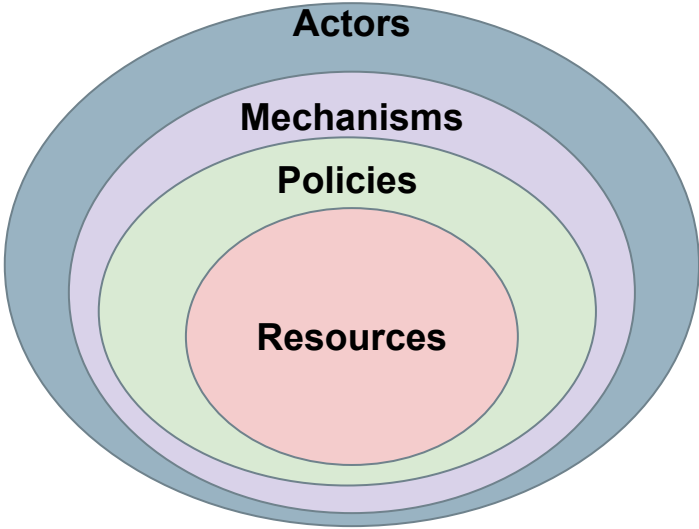
Taylor Buley, Forbes.com

Security

- The ability of the software to protect data and information from unauthorized access.
  - **While still providing access to people and systems that are authorized.**
- Can we protect software from attacks?
  - Any action taken against a computer system with the intent of causing harm.
  - Unauthorized access attempts.
  - Attempts to deny service to legitimate users.

Security

- Processes that allow owners of resources to control access.
  - Actors (systems or users).
  - Resources are sensitive elements, operations, and data of the system.
  - Policies define legitimate access to resourced.
  - Enforced by security mechanisms used by actors to gain access to resources.



Security Characterization (CIA)

- Confidentiality
  - Data and services protected from unauthorized access.
    - A hacker cannot access your tax returns on an IRS server.
- Integrity
  - Data/services not subject to unauthorized manipulation.
    - Your grade has not changed since assigned.
- Availability
  - The system will be available for legitimate use.
    - A DDOS attack will not prevent your purchase.

Supporting CIA

- Authentication - Verifies identities of all parties.
  - Did the e-mail really come from the bank?
- Nonrepudiation - Guarantees that the sender of a message cannot deny sending the message, and the recipient cannot deny receiving the message.
  - You cannot deny ordering the book, and Amazon cannot claim you never ordered.
- Authorization - Grants privilege of performing a task.
  - The bank authorizes you to check balances.

Security Approaches

- Achieving security relies on:
  - Detecting attacks.
  - Resisting attacks.
  - Reacting to attacks.
  - Recovering from attacks.
- Objects being protected are:
  - Data at rest.
  - Data in transit.
  - Computational processes.

Security is Risk Management

- **Not simply secure/not secure.**
  - All systems will be compromised.
  - Try to avoid attack, prevent damage, and quickly recover.
  - Balance risks against cost of guarding against them.
  - Set realistic expectations!

Security Scenarios

Security Quality Scenarios

- Measure of the system's ability to protect data from unauthorized access while still providing service to authorized users.
- Scenarios measure response to attack.
  - Stimuli are attacks from external systems/users or demonstrations of policies (log-in, authorization).
- Responses: auditing, logging, reporting, analyzing.

Generic Security Scenario

- **Overview:** Description of the scenario.
- **System/environment state:**
  - The attack can come when the system is online or offline
  - Connected to or disconnected from a network
  - Behind a firewall or open to a network
  - Fully operational, partially operational, or not operational.

Generic Security Scenario

- **External Stimulus:**
  - The source of the attack may be either a human or another system. It may have been previously identified or may be currently unknown.
  - A human attacker may be from outside the organization or from inside the organization.
  - The stimulus is an **attack** (unauthorized attempt to display data, change or delete data, access services, change the system's behavior, or reduce availability).

Generic Security Scenario

- **Required system behavior:**
  - Should ensure that transactions are such that:
    - Data/services protected from unauthorized access
    - Data/services not manipulated without authorization
    - Parties to a transaction are identified and cannot repudiate their involvement
    - Data, resources, and system services will be available for legitimate use.

Generic Security Scenario

- **Required system behavior:**
  - The system should also track activities by:
    - Recording access or modification and attempts to access data, resources, or services
    - Notifying appropriate entities (people or systems) when an apparent attack is occurring.

Generic Security Scenario

- **Response Measure:** Measures of a system's response include:
  - How much of a system is compromised when a particular component or data value is compromised.
  - How much time passed before an attack was detected
  - How many attacks were resisted
  - How long it took to recover from a successful attack
  - How much data was vulnerable to a particular attack.

Bad Security Scenario

- **Overview:** How the server restricts damage when someone maliciously gains control over it.
- **System/environment state:** Multiple applications deployed on the server. Servers running and serving requests
- **External Stimulus:** One application deployed is breached.
- **Required system behavior:** Server can be configured with different privileges, providing more granular control over their access to system resources and potentially preventing one breached application from allowing access to others.
- **Response measure:** Remaining applications are not breached.

Example Security Scenario

- **Overview:** A disgruntled employee at a remote location attempts to change their pay rate.
- **System/environment state:** The system is operating normally, without problems. 100 active users are logged into the system.
- **External Stimulus:** An employee has discovered the location of a configuration file storing all employee pay rates. They log in (using their credentials) and use a stolen passkey to open the locked file. They modify the file with a new rate and save changes.

Example Security Scenario

- **Required system behavior:** The system maintains an audit trail. The user is able to modify the file, as they have the passkey. However, the log records the date, time, identify of user, and modification made. System administrators are informed of the modification.
- **Response measure:** The correct data is restored within a day and the source of tampering has been identified and reported.

Example Security Scenario 2

- **Overview:** A user attempts to authenticate with the beer dispensing system (to purchase beer) but the authentication fails due to unrecognized auth token or due to system unavailability.
- **System/environment state:** There is a valve installed on the tap. There is a flow meter installed on the tap. There is a buzzer installed on the Kegboard. Authentication hardware (RFID or one-wire) is installed on the Kegboard. There is no pour in progress. The system is operating normally, without problems.
- **External Stimulus:** A user presents an auth token to the authentication hardware on the Kegboard.

Example Security Scenario 2

- **Required system behavior:** The auth token is unrecognized, and the valve is not opened. An audible sound is played from the buzzer, indicating authentication failure.
- **Response measure:** No beer is dispensed.

Key Points

- Attacks against a system are attacks against the confidentiality, integrity, or availability of a system or its data.
  - Confidentiality: Keeping data away from those who shouldn’t have it.
  - Integrity: No unauthorized modifications or deletion of data.
  - Availability: System is accessible to authorized users.

Key Points

- Identifying, authenticating, and authorizing actors are how we determine who is entitled to access the system.
- No tactic is foolproof. Systems will be compromised.
  - Important to come up with as many scenarios as you can

Next Time

- Testing Fundamentals
  - Phases, basic types of testing, terminology
  - Optional reading: Pezze & Young, Ch 1-4

- **Form your teams!**
  - Deadline: Thursday, January 30
  - E-mail me (ggay@chalmers.se) with list of team members, e-mail addresses, and a team name.
    - Or e-mail if you want to be assigned to a team

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY