universidade de aveiro

Critical software

Software Quality Attributes

Nuno Silva, PhD, Critical Software SA
E.: npsilva@ua.pt; M: 932574030

Mestrado em Cibersegurança – Robust Software

Agenda

- Motivation
- Objectives
- Software Quality Assurance
- Software Quality Standards
- Software Quality Attributes
- References
- Exercise

Motivation

- Software Quality is the “sum” of the Software Quality attributes. If we focus on only a few of these attributes we will suffer the consequences…
- Generally, **system requirements** shall define what is expected regarding the quality attributes applicable to the system, however, this is not always completely done.
- Being aware of these attributes and the expectations will drive the design and the implementation of the system…
- and avoid bad news later!

Objectives

- Identify and quantify software quality attributes.
- Acknowledge the importance of quality attributes.
- Be aware of how they can be verified / validated.
- Be able to determine if the set of quality attributes is complete.
- Be prepared to design and implement taking into account the defined software attributes.

Software Quality Assurance

- Software Quality Assurance is a set of rules for ensuring the quality of the software that will result in the quality of software product. Software quality includes the following activities:
  - Process definition and implementation
  - Auditing
  - Training

- But what is Quality?

Software Quality Assurance

- Degree of excellence – Oxford dictionary
- Fitness for purpose – Edward Deming
- Best for the customer's use and selling price – Feigenbaum
- The totality of characteristics of an entity that bear on its ability to satisfy stated or implied needs – ISO
- Capability of a software product to satisfy stated and implied needs under specified conditions. Quality represents the degree to which software products meet their stated **requirements**.

Software Quality Standards

- IEEE 1061 Technique to establish quality and validate the software with the quality metrics
- IEEE 1059 Guidance to software verification and validation
- IEEE 1008 Supports unit testing
- IEEE 1012 Supports Verification and Validation
- IEEE 1028 Guides software inspections
- IEEE 1044 Categorizes anomalies in software
- IEEE 830 Standard for development of a system with accurate requirements specifications
- IEEE 730 Standard for the product's quality assurance
- IEEE 1061 Standard for the product's quality metrics
- IEEE 12207 Standard for life cycle processes of both data and software
- ISO/IEC 29119: Software Testing Standard
- ISO/IEC 250xx: Systems and software Quality Requirements and Evaluation (SQuaRE)

Software Quality Standards

- ISO/IEC 25010:2011 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models
  - A **quality in use model** composed of five characteristics (some of which are further subdivided into subcharacteristics) that relate to the outcome of interaction when a product is used in a particular context of use. This system model is applicable to the complete human-computer system, including both computer systems in use and software products in use.
  - A **product quality model** composed of eight characteristics (which are further subdivided into subcharacteristics) that relate to static properties of software and dynamic properties of the computer system. The model is applicable to both computer systems and software products.

Software Quality Standards

- ISO/IEC 25010:2011 **quality in use model**
  - Product dev activities that can use the quality models include:
    - identifying software and system requirements;
    - validating the comprehensiveness of a requirements definition;
    - identifying software and system design objectives;
    - identifying software and system testing objectives;
    - identifying quality control criteria as part of quality assurance;
    - identifying acceptance criteria for a software product and/or software-intensive computer system;
    - establishing measures of quality characteristics in support of these activities.

Software Quality Attributes

Source: ISO/IEC CD 25010 Software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Quality model and guide, 2011.

Software Quality Attributes

We can have a very extensive list of attributes:

- Safety
- Security
- Reliability
- Resilience
- Robustness
- Understandability
- Testability
- Adaptability
- Modularity
- Complexity
- Portability
- Usability
- Reusability
- Efficiency
- Learnability
- And many other “ilities”

Software Quality Attributes

- Static
  - System structure and organization – architecture and design related, source code.
  - Not visible to the operator but affect system's development and maintenance costs.
- Dynamic
  - System behavior – architecture, design and source code, configuration and deployment parameters, system environment and running platform.
  - They are perceived at runtime, visible to the operator.

Software Quality Attributes

universidade de aveiro

Critical software

- Examples of Static Quality Attributes
  - Testability
  - Maintainability
  - Reusability
  - Modularity
  - Extensibility
- How can they be verified/tested:
  - Reviews (design, documentation)
  - Inspections (design, documentation, code)
  - Static Code Analysis (coding style, complexity, coupling …)
  - Pair Programming

Software Quality Attributes

universidade de aveiro

Critical software

- Examples of Dynamic Quality Attributes
  - Robustness
  - Scalability
  - Fault Tolerance
  - Throughput
  - Latency
- How can they be verified/tested:
  - Testing (memory usage, execution times)
  - Non-Functional tests (performance, load/stress, robustness/fault injection, simulators, log players, …)

Software Quality Attributes

universidade de aveiro

Critical software

- For Security we are focusing now on:
  - Confidentiality
  - Integrity
  - Non-repudiation
  - Accountability
  - Authenticity
  - Compliance
- Quality attributes will drive architectural tradeoffs (e.g. stateless vs stateful solution).

Software Quality Attributes::Confidentiality

- "is the property, that information is not made available or disclosed to unauthorized individuals, entities, or processes." (*Beckers, K. (2015). Pattern and Security Requirements: Engineering-Based Establishment of Security Standards. Springer. p. 100. ISBN 9783319166643.*)
- Similar to "privacy" the two concepts aren't interchangeable. Confidentiality is a component of privacy that is implemented to protect data from unauthorized viewers.
- Examples of confidentiality of electronic data being compromised include laptop theft, password theft, or sensitive emails being sent to the incorrect individuals.

Software Quality Attributes::Integrity

- Maintaining and assuring the accuracy and completeness of data over its entire lifecycle.(*Boritz, J. Efrim (2005). "IS Practitioners' Views on Core Concepts of Information Integrity". International Journal of Accounting Information Systems. Elsevier.* ***6*** *(4): 260–279. doi:10.1016/j.accinf.2005.07.001*)
- This means that data cannot be modified in an unauthorized or undetected manner.
- It can be viewed as a special case of consistency as understood in the classic ACID model of transaction processing.
- Information security systems typically provide message integrity alongside confidentiality.
- Nota: ACID = Atomicity, Consistency, Isolation and Durability

Software Quality Attributes::Non-repudiation

- It implies that one party of a transaction cannot deny having received a transaction, nor can the other party deny having sent a transaction.(*McCarthy, C. (2006). "Digital Libraries: Security and Preservation Considerations". In Bidgoli, H. (ed.). Handbook of Information Security, Threats, Vulnerabilities, Prevention, Detection, and Management.* ***3****. John Wiley & Sons. pp. 49–76. ISBN 9780470051214*)
- However, it is not, for instance, sufficient to show that the message matches a digital signature signed with the sender's private key, and thus only the sender could have sent the message, and nobody else could have altered it in transit (data integrity). The alleged sender could in return demonstrate that the digital signature algorithm is vulnerable or flawed, or allege or prove that his signing key has been compromised.
- The sender may repudiate the message (because authenticity and integrity are pre-requisites for non-repudiation).

Software Quality Attributes::Accountability

- People will be held responsible for their actions and for how they perform their duties.
- Accountability involves having control and verification systems in place, and, if necessary, the ability to arrest, prosecute and convict offenders for illegal, or corrupt behaviour. All personnel must be held accountable under the law regardless of rank, status or office. (CIDS (2015), Integrity Action Plan: a handbook for practitioners in defence establishments. p 8.)

Software Quality Attributes::Authenticity

- The property that data originated from its purported source. In the context of a key-wrap algorithm, the source of authentic data is an entity with access to an implementation of the authenticated-encryption function with the Key-Encryption-Key (KEK). (NIST SP 800-38F)
- The property of being genuine and being able to be verified and trusted; confidence in the validity of a transmission, a message, or message originator. (NIST SP 800-37 Rev. 2)

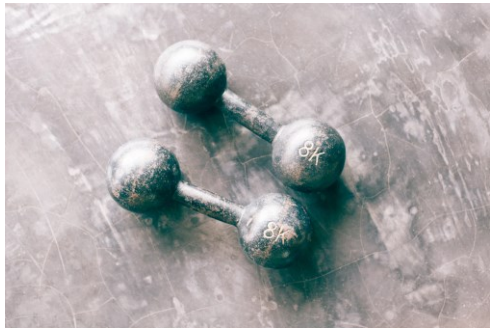
Software Quality Attributes::Compliance

- An effective system for IT security compliance ensures that only individuals with the appropriate credentials can access the secure systems and databases that contain sensitive customer data. IT organizations that implement security monitoring systems must ensure that access to those systems is monitored at an organization level, and that actions within the system are logged such that they can be traced to their origin.
- **GDPR** - The European General Data Protection Act (GDPR) is applied to all companies that process the personal data of people who live in the European Union, even companies that are physically based outside of Europe. Compliance to GDPR is now mandatory.

References

- Cyber Security Engineering: A Practical Approach for Systems and Software Assurance, Nancy Mead and Carol Woody, 2016 (https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=483667).
- ISO/IEC 250xx: Systems and software Quality Requirements and Evaluation (SQuaRE) (https://webstore.ansi.org/industry/software/software-engineering/square-software-product-quality-requirements-and-evaluation)

Exercise



Exercise

- #1: Use the IEEE Manuscript Templates for Conference Proceedings for your Report (A4 DOC or LaTeX), **present it in pdf (in English)** (https://www.ieee.org/conferences/publishing/templates.html);
  - Suggested Sections:
    - **Abstract (10%) / 1. Introduction (30%) / 2. Secure Life Cycle Description (50%)** / 3. Solution Description (-) / 4. Relevant Requirements (-) / 5. Discussion of Results (-) / 6. Conclusions (-) / References (-)
  - Note1: 10% is for the presentation, English, completeness and consistency of the work.
  - Note2: You can add one or 2 more sections, if necessary
  - Note3: Work as a group of 3 (eventually 4) and start the work in class – **present the main idea before the end of the class for approval**.
  - **Deadline: Class + 2 weeks**

Exercise

- #2: Produce a report (<= 4 pages):
  - Title: For now its not relevant
  - Fill in the name and affiliation part (include all of the group)
  - Abstract (10%)
  - 1. Introduction (30%)
  - 2. Secure Life Cycle Description (50%)
- See details in the next slides.

- Note: **You will use this same work for future assignments – do it well.**

Exercise

- Abstract (10%)
  - Write a first version of an abstract (max 200 words) where your state that you are reporting about a secure solution developed to solve **Reliability, Robustness** and **Security** issues and why that solution is important, relevant and different from the existing applications.

  - Note1: Reliability and Robustness are defined in the “Extra slides”.
  - Note2: For the part of “different from the existing applications” try to be imaginative.
  - Note3: Define a sample project, sample solution with your team

Exercise

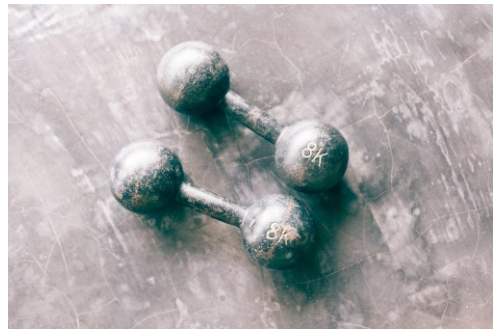
- 1. Introduction (30%)
  - Write an introduction (About 1 page)
  - Where you present a problem (real or ficticious) related to Reliability, Robustness and Security, these are only examples:
    - E.g. 1. A nuclear power plant control system that has shown to be unreliable and unsecure – rebooting randomly, erroneous outputs, freezing, external actors entering the facilities, etc.;
    - E.g. 2. A web app that can be brought down with a DoS attack, or is vulnerable to cross site scripting, accepts invalid inputs and processes them, can return unreliable results when attacked...;
    - E.g. 3. Supermarket unpaid products detection system that shuts down or doesn't detect randomly stollen products;
    - E.g. 4. A car braking system that has performance issues (e.g. takes 250 ms to react to a brake commands), stops braking at midnight, instead of braking slowly activated the full brake power when commanded, sometimes no brake is actuated, can be hacked from the wifi network, etc.
    - Etc.

Exercise

- 1. Introduction (30%) – cont'd
  - Present the problem (you can also refer to real similar cases) and the reason why it is important to be solved (security, commercial, life threats, publicity/image, etc.).
  - Present the foreseen solution by focusing on:
    - **Secure development**
    - **Quality of the solution**
    - **Confidentiality, Integrity, Non-repudiation, Accountability, Authenticity or Compliance of the solution whenever applicable.**
  - And that will be the rest of the report...

Exercise

- 2. Secure Life Cycle Description (50%)
  - Write down a short plan (About 2 pages) by picking up at least one process per phase (there are 6 main phases) and developing it (**what is planned to be done in each phase: who is the responsible for each phase, what tasks are foreseen, what tools and technologies are needed, what training …**).
  - **Phases:**
    - See next slide
    - Present it in structured text (subsections, bullet points, table or diagrams, or a combination of those)



Exercise

- Don’t forget to use/refer to the phases of the SSDL:

Phase	Microsoft SDL	McGraw Touchpoints	SAFECode
Education and awareness	Provide training		Planning the implementation and deployment of secure development
Project inception	Define metrics and compliance reporting Define and use cryptography standards Use approved tools		Planning the implementation and deployment of secure development
Analysis and requirements	Define security requirements Perform threat modelling	Abuse cases Security requirements	Application security control definition
Architectural and detailed design	Establish design requirements	Architectural risk analysis	Design
Implementation and testing	Perform static analysis security testing (SAST) Perform dynamic analysis security testing (DAST) Perform penetration testing Define and use cryptography standards Manage the risk of using third-party components	Code review (tools) Penetration testing Risk-based security testing	Secure coding practices Manage security risk inherent in the use of third-party components Testing and validation
Release, deployment, and support	Establish a standard incident response process	Security operations	Vulnerability response and disclosure

Extra slides

- Extra SW Quality Assurance Properties

Extra SW Quality Assurance Properties

- **Correctness:** The correctness of a software system refers to:
  - Agreement of program code with specifications
  - Independence of the actual application of the software system.
- The **correctness** of a program becomes especially critical when it is embedded in a complex software system.
- **Reliability:** Reliability of a software system derives from
  - Correctness
  - Availability
- The behavior over time for the fulfillment of a given specification depends on the reliability of the software system.
- **Reliability** of a software system is defined as the probability that this system fulfills a function (determined by the specifications) for a specified number of input trials under specified input conditions in a specified time interval (assuming that hardware and input are free of errors).
- A software system can be seen as reliable if this test produces a low error rate (i.e., the probability that an error will occur in a specified time interval.)
- The error rate depends on the frequency of inputs and on the probability that an individual input will lead to an error.

Extra SW Quality Assurance Properties

- **Adequacy: Factors for the requirement of Adequacy:**
  - The input required of the user should be limited to only what is necessary. The software system should expect information only if it is necessary for the functions that the user wishes to carry out. The software system should enable flexible data input on the part of the user and should carry out plausibility checks on the input. In dialog-driven software systems, we vest particular importance in the uniformity, clarity and simplicity of the dialogs.
  - The performance offered by the software system should be adapted to the wishes of the user with the consideration given to extensibility; i.e., the functions should be limited to these in the specification.
  - The results produced by the software system: The results that a software system delivers should be output in a clear and wellstructured form and be easy to interpret. The software system should afford the user flexibility with respect to the scope, the degree of detail, and the form of presentation of the results. Error messages must be provided in a form that is comprehensible for the user.
- **Learnability:** Learnability of a software system depends on:
  - The design of user interfaces
  - The clarity and the simplicity of the user instructions (tutorial or user manual).
- The user interface should present information as close to reality as possible and permit efficient utilization of the software's failures.
- The user manual should be structured clearly and simply and be free of all dead weight. It should explain to the user what the software system should do, how the individual functions are activated, what relationships exist between functions, and which exceptions might arise and how they can be corrected. In addition, the user manual should serve as a reference that supports the user in quickly and comfortably finding the correct answers to questions.

Extra SW Quality Assurance Properties

- **Robustness:** Robustness reduces the impact of operational mistakes, erroneous input data, and hardware errors.
- A software system is robust if the consequences of an error in its operation, in the input, or in the hardware, in relation to a given application, are inversely proportional to the probability of the occurrence of this error in the given application.
  - Frequent errors (e.g. erroneous commands, typing errors) must be handled with particular care.
  - Less frequent errors (e.g. power failure) can be handled more laxly, but still must not lead to irreversible consequences.
- **Maintainability:** Maintainability = suitability for debugging (localization and correction of errors) and for modification and extension of functionality.
- The **maintainability** of a software system depends on its:
  - Readability
  - Extensibility
  - Testability

Extra SW Quality Assurance Properties

- **Readability:** Readability of a software system depends on its:
  - Form of representation
  - Programming style
  - Consistency
  - Readability of the implementation programming languages
  - Structuredness of the system
  - Quality of the documentation
  - Tools available for inspection
- **Extensibility:** Extensibility allows required modifications at the appropriate locations to be made without undesirable side effects. Extensibility of a software system depends on its:
  - Structuredness (modularity) of the software system
  - Possibilities that the implementation language provides for this purpose
  - Readability (to find the appropriate location) of the code
  - Availability of comprehensible program documentation

Extra SW Quality Assurance Properties

- **Testability**: suitability for allowing the programmer to follow program execution (runtime behavior under given conditions) and for debugging. The testability of a software system depends on its:
  - Modularity
  - Structuredness
- Modular, well-structured programs prove more suitable for systematic, stepwise testing than monolithic, unstructured programs.
- Testing tools and the possibility of formulating consistency conditions (assertions) in the source code reduce the testing effort and provide important prerequisites for the extensive, systematic testing of all system components.
- **Efficiency:** ability of a software system to fulfill its purpose with the best possible utilization of all necessary resources (time, storage, transmission channels, and peripherals).

Extra SW Quality Assurance Properties

- **Portability**: the ease with which a software system can be adapted to run on computers other than the one for which it was designed.
- The portability of a software system depends on:
  - Degree of hardware independence
  - Implementation language
  - Extent of exploitation of specialized system functions
  - Hardware properties
  - Structuredness: System-dependent elements are collected in easily interchangeable program components.
- A software system can be said to be portable if the effort required for porting it proves significantly less than the effort necessary for a new implementation.