SE 2AA4, CS 2ME3 (Introduction to Software Development)

Winter 2018

04 Software Quality Continued (Ch. 2)

Dr. Spencer Smith

Faculty of Engineering, McMaster University

January 16, 2018

04 Software Quality Continued (Ch. 2)

- Administrative details
- Example qualities
  - Correctness, Reliability, Robustness
  - Performance
  - Usability
  - Verifiability
  - Maintainability
  - Reusability
  - Portability
  - Understandability
  - Interoperability
  - Productivity
  - Timeliness
- Software systems requiring special qualities
- Measurement of quality

Administrative Details

- Avenue posts?
- Avenue notifications
- Assignment 1
  - Part 1: January 22, 2018
  - Partner Files: January 28, 2018
  - Part 2: January 31, 2018
- Questions on assignment?
  - Constructor, accessor, mutator?
- Suggest bringing your laptop to tutorials
- Suggest reading Chapter 2 of Ghezzi et al

Question on Correctness, Reliability and Robustness

Reliable programs are a superset of correct programs AND robust programs are a superset of reliable programs. Is this statement True or False?

A. True

B. False

Performance

What are some aspects of performance?

What are some ways you could measure software performance?

What are some ways you could specify performance requirements to make them unambiguous and verifiable?

Performance

- The performance of a computer product is the efficiency with which the product uses its resources (memory, time, communication)
- Performance can be evaluated in three ways
  - Empirical measurement
  - Analysis of an analytic model
  - Analysis of a simulation model
- Poor performance often adversely affects the usability and scalability of the product
- Performance requirements can often be stated as relative comparisons to existing products

Usability

What are some examples of excellent usability?

When you go to a friend's house, you can likely operate their microwave without reading the manual. What did human factors engineers do to make this possible?

What are some examples of standardization? What are some examples of a missed opportunity for standardization?

Usability

- The usability of a software product is the ease with which a typical human user can use the product
- Usability depends strongly on the capabilities and preferences of the user
- The user interface of a software product is usually the principle factor affecting the product's usability
- Human computer interaction (HCI) is a major interdisciplinary subject concerned with understanding and improving interaction between humans and computers

Verifiability

- The verifiability of a software product is the ease with which the product's properties (such as correctness and performance) can be verified
- Verifiability is often an internal quality, but it can be an external quality

Maintainability

- The maintainability of a software product is the ease with which the product can be modified after its initial release
- Maintenance costs can exceed 60% of the total cost of the software product
- There are three main categories of software maintenance
  1. Corrective: Modifications to fix residual and introduced errors
  2. Adaptive: Modifications to handle changes in the environment in which the product is used
  3. Perfective: Modifications to improve the qualities of the software
- Software maintenance can be divided into two separate qualities
  1. Repairability: The ability to correct defects
  2. Evolvability: The ability to improve the software and to keep it current

Maintainability

What do software developers do to promote maintainability?

Reusability

What are the advantages of reusing code?

Why doesn't it happen more often?

Reusability

- A software product or component is reusable if it can be used to create a new product
- Reuse comes in two forms
  1. Standardized, interchangeable parts
  2. Generic, instantiable components
- Reusability is a bigger challenge in software engineering than in other areas of engineering

Portability

- A software product is portable if it can run in different environments
- The environment for a software product includes the hardware platform, the operating system, the supporting software and the user base
- Since environments are constantly changing, portability is often crucial to the success of a software product
- Some software such as operating systems and compilers, is inherently machine specific

Understandability

- The understandability of a software product is the ease with which the requirements, design, implementation, documentation, etc. can be understood
- Understandability is an internal quality that has an impact on other qualities such as verifiability, maintainability, and reusability
- There is often a tension between understandability and the performance of a software product
- Some useful software products completely lack understandability (e.g. those for which the source code is lost)

Interoperability

- A software system is interoperable if it can work with other systems
- A software product is an open system if parts of the system - such as interface specifications, protocols, and source code - are available to the public
- Open systems tend to be more interoperable than nonopen systems

Productivity

- The productivity of a software development process is the measure of how efficiently the process produces software
- Productivity highly depends on the skills and organization of the development team
- Productivity is very hard to measure
- The number of lines of code per unit time is a terrible metric for measuring software productivity
- Productivity can be greatly increased by the use of development tools, environments, and methods
- Software reuse decreases productivity in the short term, but increases productivity in the long term

Timeliness

- The timeliness of a software development process is the ability to deliver a product on time
- Timeliness is difficult to achieve in software development
- Important trade-off: Should a software product with flaws be delivered on time or should it be delivered late without flaws?
- Standard project management techniques are difficult to apply to software engineering because
  - It is difficult to define the requirements for software
  - It is difficult to quantify software
  - Requirements for software tend to continuously change as the project progresses
- Incremental delivery is one technique for achieving timeliness

Visibility

- A software development process is visible if the steps of the process and the product itself are documented
- The documentation should be accessible to the whole development team as well as to management
- Benefits of visibility
  - Promotes communication
  - Facilitates planning
  - Protects against personnel changes

Relationship between Qualities

Draw a diagram showing the relationships between the various software qualities

Software Systems requiring Special Qualities

- Information systems store and retrieve data
  - Information security (privacy, integrity, and availability) is a key quality
- Real-time systems respond to external events within a strict time-frame
  - Safety is often an important quality
- Distributed systems consist of independent subsystems connected by communication networks
  - Important qualities include concurrency control, parallelizability, fault tolerance, code mobility

Software Systems requiring Special Qualities Continued

- Embedded systems control physical devices
  - usability and other human-oriented qualities are not as important as for other software systems
- Scientific computing
  - Accuracy important – how close the output is to the true solution (not explicitly in Ghezzi et al.)
  - Reliability is very important, but verifiability is difficult to achieve
  - Sometimes sacrifice maintainability and portability for performance

Measurement of Quality

- A software quality is only important if it can be measured - without measurement there is no basis for claiming improvement
- A software quality must be precisely defined before it can be measured
- Most software qualities do not have universally accepted
- Can you directly measure maintainability?
- How might you measure maintainability?

High Quality Documentation

- To achieve external qualities for documentation, there are some generally agreed on internal qualities
- Internal qualities can more likely be directly measured
- Following list of qualities based on IEEE guidelines for requirements (IEEE Std 830-1998)
  - Complete
  - Consistent
  - Modifiable
  - Traceable
  - Unambiguous
  - Correct
  - Verifiable
  - Abstract