SE 2AA4 Winter 2007

02 Software Qualities

William M. Farmer

Department of Computing and Software
McMaster University

15 January 2007

How Software Differs from other Engineering Products

- Intangible.
  - Not physical.
  - Hard to visualize.
  - Hard to separate what is key from what is incidental.
- Malleable.
  - Easy to modify.
  - But modification requires care.
- Human intensive.
  - Software production is 99.9% engineering, 0.1% manufacturing.
  - Software is essentially just documentation.

Software Qualities

- The goal of software engineering is to produce quality software. But what are the desirable qualities that software should possess?
- External vs. internal software qualities.
  - External qualities are visible to the user.
  - Internal qualities are visible to the developer.
  - Internal qualities help external qualities to be achieved.
- Product vs. process qualities.
  - Product qualities concern the product itself.
  - Process qualities concern how the product is developed.
  - Process qualities help product qualities to be achieved.
- The importance of a particular software quality varies across software products.
  - For example, external qualities are not as important for embedded systems as for desktop software.

Correctness

- A software product is correct if it satisfies its requirements specification.
- Correctness is extremely difficult to achieve because:
  - The requirements specification my be imprecise, ambiguous, inconsistent, based on incorrect knowledge, or nonexistent.
  - Requirements often compete with each other.
  - It is virtually impossible to produce "bug-free" software.
  - It is very difficult to verify or measure correctness.
- If the requirements specification is formal, correctness can in theory and possibly in practice be:
  - Mathematically defined.
  - Proven by mathematical proof.
  - Disproven by counterexample.

Reliability

- A software product is reliable if it usually does what it is intended to do.
- Correctness is an absolute quality, while reliability is a relative quality.
  - A software product can be both reliable and incorrect.
- Reliability can be statistically measured.
- Software products are usually much less reliable than other engineering products.

Robustness

- A software product is robust if it behaves reasonably even in unanticipated or exceptional situations.
- A correct software product need not be robust.
  - Correctness is accomplished by satisfying requirements.
  - Robustness is accomplished by satisfying unstated requirements.

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:
  1. Empirical measurement.
  2. Analysis of a analytic model.
  3. Analysis of a simulation model.
- Poor performance often adversely affects the usability and scalability of the product.

Usability

- The usability of a software product is the ease with which an expected 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 principal 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 can be both an internal and 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.

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.
- Increasing reusability decreases production cost.
- 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 the 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.

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 an important quality for many real-time systems.
- Distributed systems consist of independent subsystems connected by communication networks.
  - Qualities important to distributed systems: concurrency control, parallelizability, fault tolerance, code mobility.
- Embedded systems control physical devices.
  - Usability and other human-oriented qualities are not as important for embedded systems as for other software systems.

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 metrics for measuring them.