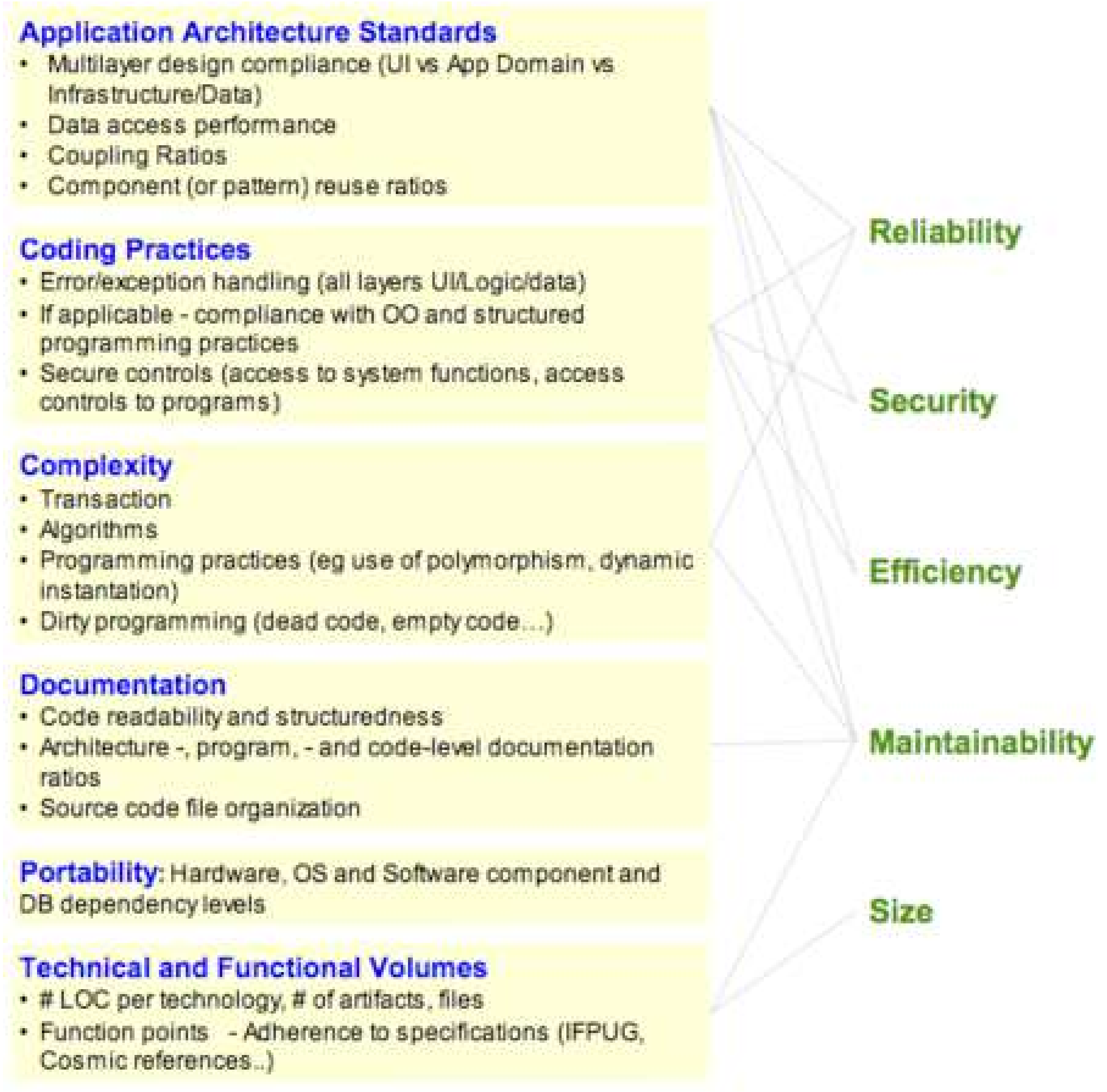
Measuring Software Quality

Impact on Quality Characteristics



Application Architecture Standards
- Multilayer design compliance (UI vs App Domain vs Infrastructure/Data)
- Data access performance
- Coupling Ratios
- Component (or pattern) reuse ratios

Coding Practices
- Error/exception handling (all layers UI/Logic/data)
- If applicable - compliance with OO and structured programming practices
- Secure controls (access to system functions, access controls to programs)

Complexity
- Transaction
- Algorithms
- Programming practices (eg use of polymorphism, dynamic instantation)
- Dirty programming (dead code, empty code...)

Documentation
- Code readability and structuredness
- Architecture -, program, - and code-level documentation ratios
- Source code file organization

Portability: Hardware, OS and Software component and DB dependency levels

Technical and Functional Volumes
- # LOC per technology, # of artifacts, files
- Function points - Adherence to specifications (IFPUG, Cosmic references..)

Importance of Quality for

If you work for an organization that produces software, the quality of the software you produce could determine a few important things:

- The revenue of the company you work for
- The priority of the project or product you are working on within the organization
- The likelihood you’ll be promoted to a senior role, demoted or even fired
- Your salary.

Quality Aspect 1: Reliability @ CISQ

Reliability refers to the level of risk inherent in a software product, and the likelihood it will fail. It also addresses "stability," as termed by ISO: how likely are there to be regressions in the software when changes are made.

A related term coined in recent years is "**resilience**." This views the problem from a different direction, asking what is the software's ability to deal with failure, which will inevitably happen.

For example, modern applications based on containerized micro-services can be easily and automatically redeployed in case of failure, making them highly resilient.

Why measure reliability?

To reduce and prevent severe malfunctions or outages, and errors that can affect users and decrease user satisfaction.

Software is better if it fails less often, and easily recovers from failure when it happens.

How can you measure reliability?

- **Production incidents** – A good measure of a system's reliability is the number of high priority bugs identified in production.
- **Reliability testing** – Common types of reliability testing are load testing, which checks how the software functions under high loads, and regression testing, which checks how many new defects are introduced when software undergoes changes.

The aggregate results of these tests over time can be a measure of software resilience.

Consortium for IT Software Quality

How can you measure reliability?

- **Reliability evaluation** – An in-depth test conducted by experts who construct an operational environment <u>simulating</u> <u>the real environment</u> in which the software will be run. In this simulated environment, they test how the software works in a steady state, and with certain expected growth (e.g. more users or higher throughput).
- **Average failure rate** – Measures the average number of failures per period per deployed unit or user of the software.

CISQ
Consortium for IT Software Quality

Quality Aspect 2: **Performance @** CISQ

Consortium for IT Software Quality

In the CISQ software quality model, this aspect is known as “**Efficiency**.” Typically, the most important elements that contribute to an application’s performance are how its source code is written, its software architecture and the components within that architecture: databases, web servers, etc. **Scalability** is also key to performance: systems which are able to scale up and down can adapt to different levels of required performance.

Quality Aspect 2: **Performance**

Performance is especially important in fields like algorithmic or transactional processing, where massive amounts of data need to be processed very quickly, and even small latency can cause significant problems.

But today performance is becoming universally important as users of web and mobile applications demand high performance and become quickly frustrated if a system does not respond quickly.

How can you measure performance?

- **Load testing** – Conducted to understand the behavior of the system under a certain load, for example, with 1,000 concurrent users.
- **Stress testing** – Understanding the upper limit of capacity of the system.
- **Soak testing** – Checking if the system can handle a certain load for a prolonged period of time, and when performance starts to degrade.
- **Application performance monitoring** (APM) – Providing detailed metrics of performance from the user’s perspective.

Quality Aspect 3: **Security @** CISQ Consortium for IT Software Quality

Security, in the context of software quality, reflects how likely it is that attackers might breach the software, interrupt its activity or gain access to sensitive information, due to poor coding practices and architecture. A central concept in security is “**vulnerabilities**” – known issues that can result in a security issue or breach.

The number and severity of vulnerabilities discovered in a system is an important indication of its level of security.

How can you measure security?

- **Number of vulnerabilities** – It is possible to scan software applications to identify known vulnerabilities. The number of vulnerabilities found is a good (negative) measure of security..
- **Time to resolution** – How long does it take from the time a vulnerability was introduced in the software until a fix or patch was released?

CISQ
Consortium for IT Software Quality

How can you measure security?

- **Deployment of security updates** – For software deployed on users equipment, how many users have actually installed a patch or security update?
- **Actual security incidents, severity and total time of attacks** – How many times was a system actually breached, how badly did the breach affect users, and for how long?

CISQ
Consortium for IT Software Quality

Quality Aspect 4: **Maintainability @** CISQ Consortium for IT Software Quality

Software maintainability is the ease with which software can be adapted to other purposes, how portable it is between environments, and whether it is transferable from one development team or from one product to another.

Maintainability is closely related to code quality. If code is of high quality, the software is likely to be more easily maintainable.

How can you measure maintainability?

- **Lines of code** – A very simple metric that has an impact on the maintainability of a system. Software with more lines of code tends to be more difficult to maintain and more prone to code quality issues.
- **Software complexity metrics** – There are several ways to measure how complex software is, such as cyclomatic complexity and N-node complexity. Code that is more complex is likely to be less maintainable.

How can you measure maintainability?

CISQ
Consortium for IT Software Quality

- **Static code analysis** – Automatic examination of code to identify problems and ensure the code adheres to industry standards. Static analysis is done directly on the code without actually executing the software.

Quality Aspect 5: **Rate of Delivery @** CISQ Consortium for IT Software Quality

In agile development environments, new iterations of software are delivered to users quickly.

Many organizations today ship new versions of their software every week, everyday, or even several times a day. This is known as **Continuous Delivery**, or in its extreme form, **Continuous Deployment**, in which every change to the software is immediately shipped to production.

Quality Aspect 5: **Rate of Delivery @** CISQ Consortium for IT Software Quality

Rate of software delivery is related to quality, because a new version of a software system will typically contain improvements that can impact the user.

A higher frequency of releases that are delivered to the user should, in theory, mean that the user gets better software faster.

How can you measure rate of delivery?

CISQ
Consortium for IT Software Quality

- **Number of software releases** – This is the basic measurement of how frequently new software is delivered to users.
- **Agile stories which are "done" in a certain time period** – Counting the number of "stories," or user requirements, which are actually shipped to the user, provides a more granular measure of the rate of delivery.
- **User consumption of releases** – For example, measuring the number of users who download or install a new patch or software update.