SENG 421: Software Metrics

Measuring External Product Attributes: Software Quality
(Chapter 8)

Department of Electrical & Computer Engineering, University of Calgary

B.H. Far (far@ucalgary.ca)

http://www.enel.ucalgary.ca/People/far/Lectures/SENG421/08/
Contents

- Software quality
- Software quality models: Boehm’s model, McCall’s model, ISO 9126 model, etc.
- Software quality standards and metrics
- Measuring customer satisfaction
- Software quality assurance (SQA)
- Software quality management
What is Quality?

- **Quality popular views:**
  - Something “good” but not quantifiable
  - Something luxury and classy

- **Quality professional views:**
  - Conformance to requirement (Crosby, 1979)
  - Fitness for use (Juran, 1970)
Software Quality: Difficulties

- Need to account for “creativity” in the “design” of the product and the “requirements” rather than the product itself.

- Kind of art …

- Which one has a higher quality?
What is Software Quality?  

- **Conformance to requirement**
  - The requirements are clearly stated and the product must conform to it
  - Any deviation from the requirements is regarded as a defect
  - A good quality product contains fewer defects

- **Fitness for use**
  - Fit to user expectations: meet user’s needs
  - A good quality product provides better user satisfaction

Both $\rightarrow$ Dependable computing system
What is Software Quality?  /2

Various viewpoints/perspectives:

- Conformance to customers’ requirements
- Requirement, design and development quality
- Process quality vs. end-product quality
  - Process quality: higher usability and dependability
  - End-product quality: less failure
- Internal vs. external characteristics
  - Relativity: advantage over similar products
  - Conformance: conformance to standards
**Definition: Software Quality**

- ISO 8402 definition of quality:
  - The totality of features and characteristics of a product or a service that bear on its ability to satisfy stated or implied needs

- ISO 9126 definition of quality:
  - The totality of features and characteristics of a software product that bear on its ability to satisfy stated or implied needs
Software Quality: Classification

- By observation:
  - Internal quality (quality while the product is being produced, including process and checks)
  - External quality (final product quality)

- By process:
  - Design quality
  - Implementation quality
  - Test quality
  - Maintenance quality
Software Quality Models
Modeling Software Quality

- **Purpose:** To make the concept “quality” operational via refinement and measurement
- Selecting some attributes (or factors)
- Plotting relationship among attributes (many-to-many relationships)

or

- Selecting some criteria (or intermediate and primitive constructs) to represent the attributes
- Mapping criteria (or primitive constructs) to metrics
Quality Model – Structure

User oriented

Software oriented

SW Quality

Quality Factor 1

Quality Factor 2

... Quality Factor n

Quality Criterion 1

Quality Criterion 2

Quality Criterion 3

... Quality Criterion m

Measures

Kapitel 3.1.2

far@ucalgary.ca
Example: Attribute Expansion

- **Design by measurable objectives:**
  Incremental design is evaluated to check whether the goal for each increment was achieved.

### Quality objective

#### Availability

- % of planned System uptime
  - Worst: 95%
  - Best: 99%

#### User friendliness

- Days on job to learn task supplied by new system
  - Worst: 7 days
  - Best: 1 day
Example: CUPRIMDA Model

- Quality parameters (parameters for fitness):
  - Capability
  - Usability
  - Performance
  - Reliability
  - Installability
  - Maintainability
  - Documentation
  - Availability

<table>
<thead>
<tr>
<th>CAPABILITY</th>
<th>USABILITY</th>
<th>PERFORMANCE</th>
<th>RELIABILITY</th>
<th>INSTALLABILITY</th>
<th>MAINTAINABILITY</th>
<th>DOCUMENTATION</th>
<th>AVAILABILITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>●</td>
</tr>
</tbody>
</table>

Example: McCall’s Model

Use

Factor (11)

Product operation
- Usability
- Integrity
- Efficiency
- Correctness
- Reliability
- Maintainability

Product revision
- Testability
- Flexibility
- Reusability
- Portability
- Interoperability

Product transition

Criteria (25)
- Operability
- Training
- Communicativeness
- I/O volume
- I/O rate
- Access control
- Access audit
- Storage efficiency
- Execution efficiency
- Traceability
- Completeness
- Accuracy
- Error tolerance
- Consistency
- Simplicity
- Conciseness
- Instrumentation
- Expandability
- Generality
- Self-descriptiveness
- Modularity
- Machine independence
- S/w system independence
- Comms commonality
- Data commonality

Figure from Fenton’s Book
McCall Model Elements

- McCall quality model is organized around three types of Elements:
  - **Quality Factors (To specify):** They describe the external view of the software, as viewed by the users.
  - **Quality Criteria (To build):** They describe the internal view of the software, as seen by the developer.
  - **Quality Metrics (To control):** They are defined and used to provide a scale and method for measurement.
**SATC* Software Quality Model**

- *Software Assurance Technology Center (1996)*
- NASA Goddard Space Flight Center

<table>
<thead>
<tr>
<th>GOALS</th>
<th>ATTRIBUTES</th>
<th>METRICS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements Quality</td>
<td>Ambiguity</td>
<td>Number of Weak Phrases</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Number of Optional Phrases</td>
</tr>
<tr>
<td></td>
<td>Completeness</td>
<td>Number of TBDs/TEAs</td>
</tr>
<tr>
<td></td>
<td>Understandability</td>
<td>Document Structure</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Readability Index</td>
</tr>
<tr>
<td></td>
<td>Volatility</td>
<td>Count of changes / Count of requirements</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Life cycle stage when change is made</td>
</tr>
<tr>
<td></td>
<td>Traceability</td>
<td>Number of software requirements not traced to system requirements</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Number of software requirements not traced to code and tests</td>
</tr>
<tr>
<td>Product/Code Quality</td>
<td>Structure/Architecture</td>
<td>Logic complexity</td>
</tr>
<tr>
<td></td>
<td>Strength/Architecture</td>
<td>GOTO usage</td>
</tr>
<tr>
<td></td>
<td>Size</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Maintainability</td>
<td>Correlation of complexity/size</td>
</tr>
<tr>
<td></td>
<td>Reusability</td>
<td>Correlation of complexity/size</td>
</tr>
<tr>
<td></td>
<td>Internal Documentation</td>
<td>Comment Percentage</td>
</tr>
<tr>
<td></td>
<td>External Documentation</td>
<td>Readability Index</td>
</tr>
<tr>
<td>Implementation Effectiveness</td>
<td>Resource Usage</td>
<td>Staff hours spent on life cycle activities</td>
</tr>
<tr>
<td></td>
<td>Completion Rate</td>
<td>Task completions</td>
</tr>
<tr>
<td></td>
<td>Planned task completions</td>
<td></td>
</tr>
<tr>
<td>Testing Effectiveness</td>
<td>Correctness</td>
<td>Errors and criticality</td>
</tr>
<tr>
<td></td>
<td>Time of finding of errors</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Time of error fixes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Code Location of Fault</td>
<td></td>
</tr>
</tbody>
</table>
Quality Model: ISO 9126

Are the required functions available in this software?

Functionality

How easy is to transfer the software to another environment?

Portability

How reliable is the software?

Reliability

How easy is to change the software?

Maintainability

Is the software easy to use?

Usability

How efficient is the software?

Efficiency

ISO 9126
Flexible Quality Model

- No need to stick to a fixed (published) model.
- You can define your own quality model based on users (customers and company) consensus. The model will be composed of quality attributes which are important for the given product.
- The model should be verified by actual measurement.
Process-oriented Quality Models

- **Purpose:** To characterize process quality
- **Measure** process and product attributes

**Examples of Process Attributes:**
- Effort (person-months)
- Duration (months)

**Examples of Product Attributes:**
- Code size (#classes)
- Test set size (#test cases)
- Correctness (#defects)
Example: Defect Distributions

- Combination of (i) defect number, (ii) injection to detection period, and (iii) defect origin can be used to measure development (process) quality.
- Goal: Reduce number of defects that cross process step boundaries.
Development effort distribution over process steps should make rework shares explicit.

Goal: Detect and reduce rework.
Software Quality Standards
Software Quality Standards

- Software quality management standards (e.g., ISO 9000 series)
  → covered in SENG 511

- Software process quality assessment standards (e.g., ISO15504 / SPICE)
  → covered in SENG 511

- Software quality measurement standards (e.g., ISO 9126)
  → explained here
ISO 9126: Parts

ISO 9126:

“A set of attributes of a software product by which its quality is described and evaluated. A software quality characteristic may be refined into multiple levels of sub-characteristics.”
ISO 9126: Quality Lifecycle

- **Internal metrics** do not rely on software execution (→ static measures)
- **External metrics** are applied to running software
- **Quality-in-use metrics** are applied when the final product is used in real conditions

Ideally, internal quality determines external quality and external quality determines quality in use.
ISO 9126 – Internal & External Base Measures

### Internal Base Measures

<table>
<thead>
<tr>
<th>Measure Name</th>
<th>Unit of Measurement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Number of Functions</td>
<td>Function (number of)</td>
</tr>
<tr>
<td>2 Number of Data Items</td>
<td>Item (number of)</td>
</tr>
<tr>
<td>3 Number of Data Formats</td>
<td>Format (number of)</td>
</tr>
<tr>
<td>4 Number of Interface Protocols</td>
<td>Protocol (number of)</td>
</tr>
<tr>
<td>5 Number of Access Types</td>
<td>Access-Type (number of)</td>
</tr>
<tr>
<td>6 Number of Access Controllability Requirements</td>
<td>Requirement (number of)</td>
</tr>
<tr>
<td>7 Number of Instances of Data Corruption</td>
<td>Instance (number of)</td>
</tr>
<tr>
<td>8 Number of Compliance Issues</td>
<td>Item (number of)</td>
</tr>
<tr>
<td>9 Number of Additional Requirements Compliance</td>
<td>Interface (number of)</td>
</tr>
<tr>
<td>10 Number of Faults</td>
<td>Fault (number of)</td>
</tr>
<tr>
<td>11 Number of Test Cases</td>
<td>Test-Case (number of)</td>
</tr>
<tr>
<td>12 Number of Restorations</td>
<td>Requirement (number of)</td>
</tr>
<tr>
<td>13 Number of Input Items Which Could Check for Valid Data</td>
<td>Item (number of)</td>
</tr>
<tr>
<td>14 Number of Operations</td>
<td>Operation (number of)</td>
</tr>
<tr>
<td>15 Number of Messages Implemented</td>
<td>Message (number of)</td>
</tr>
<tr>
<td>16 Number of Interface Elements</td>
<td>Element (number of)</td>
</tr>
<tr>
<td>17 Response Time</td>
<td>Second or Millisecond</td>
</tr>
<tr>
<td>18 Transaction Time</td>
<td>Second or Millisecond</td>
</tr>
<tr>
<td>19 I/O Utilization (Number of Buffers)</td>
<td>Buffer (number of)</td>
</tr>
<tr>
<td>20 Memory Utilization</td>
<td>Byte</td>
</tr>
<tr>
<td>21 Number of Lines of Code Directly Related to System Calls</td>
<td>Line (number of)</td>
</tr>
<tr>
<td>22 Number of I/O Related Errors</td>
<td>Error (number of)</td>
</tr>
<tr>
<td>23 Number of Memory Related Errors</td>
<td>Error (number of)</td>
</tr>
<tr>
<td>24 Number of Items Required to be Logged</td>
<td>Item (number of)</td>
</tr>
<tr>
<td>25 Number of Modifications Made</td>
<td>Modification (number of)</td>
</tr>
<tr>
<td>26 Number of Variables</td>
<td>Variable (number of)</td>
</tr>
<tr>
<td>27 Number of Diagnostic Functions Required</td>
<td>Function (number of)</td>
</tr>
<tr>
<td>28 Number of Entities</td>
<td>Entity (number of)</td>
</tr>
<tr>
<td>29 Number of Built-in Test Functions Required</td>
<td>Function (number of)</td>
</tr>
<tr>
<td>30 Number of Test Dependencies on Other System</td>
<td>Dependency (number of)</td>
</tr>
<tr>
<td>31 Number of Diagnostic Checkpoints</td>
<td>Checkpoint (number of)</td>
</tr>
<tr>
<td>32 Number of Data Structures</td>
<td>DataStructure (number of)</td>
</tr>
<tr>
<td>33 Total Number of Setup Operations</td>
<td>Operation (number of)</td>
</tr>
<tr>
<td>34 Number of Installation Steps</td>
<td>Step (number of)</td>
</tr>
</tbody>
</table>

### External Base Measures

<table>
<thead>
<tr>
<th>Measure Name</th>
<th>Unit of Measurement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Number of Functions</td>
<td>Function (number of)</td>
</tr>
<tr>
<td>2 Operation Time</td>
<td>Minute</td>
</tr>
<tr>
<td>3 Number of Incorrect Operations Encountered by Users</td>
<td>Case (number of)</td>
</tr>
<tr>
<td>4 Total Number of Data Formats</td>
<td>Format (number of)</td>
</tr>
<tr>
<td>5 Number of Illegal Operations</td>
<td>Operation (number of)</td>
</tr>
<tr>
<td>6 Number of Items Requiring Compliance</td>
<td>Item (number of)</td>
</tr>
<tr>
<td>7 Number of Additional Requirements Compliance</td>
<td>Interface (number of)</td>
</tr>
<tr>
<td>8 Number of Faults</td>
<td>Fault (number of)</td>
</tr>
<tr>
<td>9 Number of Failures</td>
<td>Failure (number of)</td>
</tr>
<tr>
<td>10 Product Size</td>
<td>Byte</td>
</tr>
<tr>
<td>11 Number of Test Cases</td>
<td>Case (number of)</td>
</tr>
<tr>
<td>12 Number of Statements</td>
<td>Statement (number of)</td>
</tr>
<tr>
<td>13 Time to Report</td>
<td>Minute</td>
</tr>
<tr>
<td>14 Down Time</td>
<td>Minute</td>
</tr>
<tr>
<td>15 Number of Restarts</td>
<td>Restart (number of)</td>
</tr>
<tr>
<td>16 Number of Restorations Required</td>
<td>Restoration (number of)</td>
</tr>
<tr>
<td>17 Number of Terminals</td>
<td>Terminal (number of)</td>
</tr>
<tr>
<td>18 Number of I/O Data Items</td>
<td>Item (number of)</td>
</tr>
<tr>
<td>19 Ease of Function Learning</td>
<td>Minute</td>
</tr>
<tr>
<td>20 Number of Tasks</td>
<td>Task (number of)</td>
</tr>
<tr>
<td>21 Help Frequency</td>
<td>Access (number of)</td>
</tr>
<tr>
<td>22 Extra Correction</td>
<td>Minute</td>
</tr>
<tr>
<td>23 Number of Screens or Forms</td>
<td>Screen (number of)</td>
</tr>
<tr>
<td>24 Number of Code Lines of Changes</td>
<td>Line (number of)</td>
</tr>
<tr>
<td>25 Number of Attempts to Communicate</td>
<td>Attempt (number of)</td>
</tr>
<tr>
<td>26 Total Number of Un savory Compliance Items Specified</td>
<td>Item (number of)</td>
</tr>
<tr>
<td>27 Response Time</td>
<td>Second or Millisecond</td>
</tr>
<tr>
<td>28 Number of Failures</td>
<td>Failure (number of)</td>
</tr>
<tr>
<td>29 Transaction Time</td>
<td>Second or Millisecond</td>
</tr>
<tr>
<td>30 Transmission Time</td>
<td>Second or Millisecond</td>
</tr>
<tr>
<td>31 Number of I/O Related Errors</td>
<td>Error (number of)</td>
</tr>
<tr>
<td>32 User Waiting Time of I/O Device Utilization</td>
<td>Second or Millisecond</td>
</tr>
<tr>
<td>33 Number of Memory Related Errors</td>
<td>Error (number of)</td>
</tr>
<tr>
<td>34 Number of Transmission Related Errors</td>
<td>Error (number of)</td>
</tr>
<tr>
<td>35 Transmission Capacity</td>
<td>Byte</td>
</tr>
<tr>
<td>36 Number of Revised Versions</td>
<td>Version (number of)</td>
</tr>
<tr>
<td>37 Number of Revised Failures</td>
<td>Failure (number of)</td>
</tr>
<tr>
<td>38 Porting User Friendliness</td>
<td>Minute</td>
</tr>
</tbody>
</table>
ISO 9126: Part 1: Quality model

- ISO/IEC 9126-1: (Released in 2001)
  Information technology - Software quality characteristics & metrics - Part 1: Quality characteristics and subcharacteristics.

- This part provides the concepts introduced in the original standard is a recommended quality model which categorizes software quality in six characteristics, which are further subdivided into subcharacteristics. The subcharacteristics have been moved from the annex to become part of the standard. They have been reworded and several new ones added.

- There is also a definition of quality in use which defines the user’s view as a result of using the software.
ISO 9126: Part 2: External metrics

- This part provides external metrics for measuring software quality characteristics.
- An external metric is a quantitative scale and measurement method, which can be used for measuring an attribute or characteristic of a software product, derived from the behaviour of the system of which it is a part.
- External metrics are applicable to an executable software product during testing or operating in later stage of development and after entering to operation process.
ISO 9126: Part 3: Internal metrics

- This part provides internal metrics for measuring software quality characteristics.
- An internal metric is a quantitative scale and measurement method, which can be used for measuring an attribute or characteristic of a software product, derived from the product itself, either direct or indirect.
- Internal metrics are applicable to a non executable software product during designing and coding in early stage of development process.
Effectiveness
- The capability of the software product to enable users to achieve specified goals with accuracy and completeness in a specified context of use.

Productivity
- The capability of the software product to enable users to expend appropriate amounts of resources in relation to the effectiveness achieved in a specified context of use.

Safety
- The capability of the software product to achieve acceptable levels of risk of harm to people, business, software, property or the environment in a specified context of use.

Satisfaction
- The capability of the software product to satisfy users in a specified context of use.
## Quality in use Base Measures

<table>
<thead>
<tr>
<th>Measure Name</th>
<th>Unit of Measurement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Task Effectiveness</td>
<td>(a given weight)</td>
</tr>
<tr>
<td>2 Total Number of Tasks</td>
<td>Task (number of)</td>
</tr>
<tr>
<td>3 Task Time</td>
<td>Minute</td>
</tr>
<tr>
<td>4 Cost of the Task</td>
<td>Dollar</td>
</tr>
<tr>
<td>5 Help Time</td>
<td>Second</td>
</tr>
<tr>
<td>6 Error Time</td>
<td>Second</td>
</tr>
<tr>
<td>7 Search Time</td>
<td>Second</td>
</tr>
<tr>
<td>8 Number of Users</td>
<td>User (number of)</td>
</tr>
<tr>
<td>9 Total Number of People Potentially Affected by the System</td>
<td>Person (number of)</td>
</tr>
<tr>
<td>10 Total Number of Usage Situations</td>
<td>Situation (number of)</td>
</tr>
</tbody>
</table>
Quality Model: ISO 9126

- Specifying software quality for a product yet to be developed is difficult for the customer and/or supplier.
- The customer needs to understand and communicate requirements for the product to be developed.
- The supplier needs to understand the requirement and to assess with confidence whether it is possible to provide the product with the right level of quality.
- **ISO 9126** will serve to eliminate any misunderstanding between customer and supplier.
- **ISO 9126** is the software product evaluation standard. This international standard defines six characteristics that describe, with minimal overlap, software quality.
Quality Model: ISO 9126 /2

1. **Functionality** is the set of attributes that bear on the existence of a set of functions and their specified properties. The functions are those that satisfy stated or implied needs.

2. **Reliability** is the set of attributes that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time.

3. **Usability** is the set of attributes that bear on the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users.

4. **Efficiency** is the set of attributes that bear on the relationship between the level of performance of the software and the amount of resources used, under stated conditions.

5. **Maintainability** is the set of attributes that bear on the effort needed to make specified modifications.

6. **Portability** is the set of attributes that bear on the ability of software to be transferred from one environment.
### Quality Model: ISO 9126 /3

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Functionality</strong></td>
<td>Suitability</td>
</tr>
<tr>
<td></td>
<td>Interoperability</td>
</tr>
<tr>
<td></td>
<td>Accuracy</td>
</tr>
<tr>
<td></td>
<td>Compliance</td>
</tr>
<tr>
<td></td>
<td>Security</td>
</tr>
<tr>
<td><strong>Reliability</strong></td>
<td>Maturity</td>
</tr>
<tr>
<td></td>
<td>Recoverability</td>
</tr>
<tr>
<td></td>
<td>Fault tolerance</td>
</tr>
<tr>
<td></td>
<td>Crash frequency</td>
</tr>
<tr>
<td><strong>Usability</strong></td>
<td>Understandability</td>
</tr>
<tr>
<td></td>
<td>Learnability</td>
</tr>
<tr>
<td></td>
<td>Operability</td>
</tr>
<tr>
<td><strong>Efficiency</strong></td>
<td>Time behaviour</td>
</tr>
<tr>
<td></td>
<td>Resource behaviour</td>
</tr>
<tr>
<td><strong>Maintainability</strong></td>
<td>Analyzability</td>
</tr>
<tr>
<td></td>
<td>Stability</td>
</tr>
<tr>
<td></td>
<td>Changeability</td>
</tr>
<tr>
<td></td>
<td>Testability</td>
</tr>
<tr>
<td><strong>Portability</strong></td>
<td>Adaptability</td>
</tr>
<tr>
<td></td>
<td>Installability</td>
</tr>
<tr>
<td></td>
<td>Conformance</td>
</tr>
<tr>
<td></td>
<td>Replacability</td>
</tr>
</tbody>
</table>

1:n relation between Characteristics and Attributes (Sub-Characteristics)
ISO 9126: Users’ View

- Users are mainly interested in using the software, its performance and the effects of using the software.
- Users evaluate the software without knowing the internal aspects of the software, or how the software is developed.
- Users’ questions may include:
  - Are the required functions available in the software?
  - How reliable is the software?
  - How efficient is the software?
  - Is the software easy to use?
  - How easy is it to transfer the software to another environment?
  - How easy is it to change this software?
ISO 9126: Developers’ View

The process of development requires the user and the developer to use the same software quality characteristics, since they apply to requirement and acceptance.

- When developing off-the-shelf software, the implied needs must be reflected in the quality requirement.

Since developers are responsible for producing software which will satisfy quality requirements they are interested in the intermediate product quality as well as in the final product quality.

- In order to evaluate the intermediate product quality at each phase of the development cycle, the developers have to use different metrics for the same characteristics because the same metrics are not applicable to all phases of the cycle.
ISO 9126: Managers’ View

- A manager is typically more interested in the overall quality rather than in a specific quality characteristic, and for this reason will need to assign weights, reflecting business requirements, to the individual characteristics.

- The manager may also need to balance the quality improvement with management criteria such as schedule delay or cost overrun, because he/she wishes to optimise quality within limited cost, human resources and time-frame.
Software Quality: ISO 9126 Standard
Quality Model: ISO 9126

Are the required functions available in this software?

Functionality

How easy is to transfer the software to another environment?

Portability

How reliable is the software?

Reliability

How easy is to change the software?

Maintainability

Is the software easy to use?

Usability

How efficient is the software?

Efficiency
## Quality Model: ISO 9126

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functionality</td>
<td>Suitability</td>
</tr>
<tr>
<td></td>
<td>Interoperability</td>
</tr>
<tr>
<td></td>
<td>Accuracy</td>
</tr>
<tr>
<td></td>
<td>Compliance</td>
</tr>
<tr>
<td></td>
<td>Security</td>
</tr>
<tr>
<td>Reliability</td>
<td>Maturity</td>
</tr>
<tr>
<td></td>
<td>Recoverability</td>
</tr>
<tr>
<td></td>
<td>Fault tolerance</td>
</tr>
<tr>
<td></td>
<td>Crash frequency</td>
</tr>
<tr>
<td>Usability</td>
<td>Understandability</td>
</tr>
<tr>
<td></td>
<td>Learnability</td>
</tr>
<tr>
<td></td>
<td>Operability</td>
</tr>
<tr>
<td>Efficiency</td>
<td>Time behaviour</td>
</tr>
<tr>
<td></td>
<td>Resource behaviour</td>
</tr>
<tr>
<td>Maintainability</td>
<td>Analyzability</td>
</tr>
<tr>
<td></td>
<td>Stability</td>
</tr>
<tr>
<td></td>
<td>Changeability</td>
</tr>
<tr>
<td></td>
<td>Testability</td>
</tr>
<tr>
<td>Portability</td>
<td>Adaptability</td>
</tr>
<tr>
<td></td>
<td>Installability</td>
</tr>
<tr>
<td></td>
<td>Conformance</td>
</tr>
<tr>
<td></td>
<td>Replacability</td>
</tr>
</tbody>
</table>
ISO 9126: 1. Functionality

- **Suitability:** Attributes of software that bear on the presence and appropriateness of a set of functions for specified tasks.
- **Accuracy:** Attributes of software that bear on the provision of right or agreed results or effects.
- **Interoperability:** Attributes of software that bear on its ability to interact with specified systems.
- **Compliance:** Attributes of software that make the software adhere to application related standards or conventions or regulations in laws and similar prescriptions.
- **Security:** Attributes of software that bear on its ability to prevent unauthorized access, whether accidental or deliberate, to programs and data.
ISO 9126: 2. Reliability

- **Maturity**: Attributes of software that bear on the frequency of failure by faults in the software.

- **Fault tolerance**: Attributes of software that bear on its ability to maintain a specified level of performance in cases of software faults or of infringement of its specified interface.

- **Crash frequency**: The number of the system crashes per unit of time.

- **Recoverability**: Attributes of software that bear on the capability to re-establish its level of performance and recover the data directly affected in case of a failure and on the time and effort needed for it.
ISO 9126: 3. Usability

- **Understandability:** Attributes of software that bear on the users’ effort for recognizing the logical concept and its applicability.

- **Learnability:** Attributes of software that bear on the users’ effort for learning its application (for example, operation control, input, output).

- **Operability:** Attributes of software that bear on the users’ effort for operation and operation control.
ISO 9126: 4. Efficiency

- **Efficiency**: The extent to which a product or process can operate using the fewest possible resources.

  - **Time behaviour**: Attributes of software that bear on response and processing times and on throughput rates in performing its function.

  - **Resource behaviour**: Attributes of software that bear on the amount of resources used and the duration of such use in performing its function.
ISO 9126: 5. Maintainability

- **Analysability:** Attributes of software that bear on the effort needed for diagnosis of deficiencies or causes of failures, or for identification of parts to be modified.

- **Changeability:** Attributes of software that bear on the effort needed for modification, fault removal or for environmental change.

- **Stability:** Attributes of software that bear on the risk of unexpected effect of modifications.

- **Testability:** Attributes of software that bear on the effort needed for validating the modified software.
ISO 9126: 6. Portability

- **Adaptability**: Attributes of software that bear on the opportunity for its adaptation to different specified environments without applying other actions or means than those provided for this purpose for the software considered.

- **Installability**: Attributes of software that bear on the effort needed to install the software in a specified environment.

- **Conformance**: Attributes of software that make the software adhere to standards or conventions relating to portability.

- **Replaceability**: Attributes of software that bear on the opportunity and effort of using it in the place of specified other software in the environment of that software.
1. Functionality: Accuracy

**Definition:**

Attributes of software that bear on the provision of right or agreed results or effects.

- The degree to which a component is free from faults in its specification, design, and implementation.
- The degree to which a component meets specified requirements or user needs and expectations.
- The ability of a component to produce specified outputs when given specified inputs, and the extent to which they match or satisfy the requirements.
1. Functionality: Accuracy /2

- **Measurement:**
  - Number of faults per phase / priority / category / cause
  - Number of faults per time period (fault rate)
  - Number of open problem reports per time period
  - Number of closed problem reports per time period
  - Number of unevaluated (pending) problem reports
  - Age of open problem reports
  - Age of unevaluated problem reports
  - Age of closed problem reports
  - …
1. Functionality: Security

- **Definition:** Attributes of software that bear on its ability to prevent unauthorized access, whether accidental or deliberate, to programs and data.

- Software system security is defined at the following levels:
  - Level 0: no security at all
  - Level 1: firewalls
  - Level 2: encryption
  - Level 3: authentication (digital ID verification)
  - Level 4: intrusion protection
  - Level 5: combination of level 1-4
1. Functionality: Security /2

- Measurement: Security level ($L_{sc}$)

$$L_{sc} = \frac{n_t}{n_{int}}$$

- $n_t$: number of successful intrusions
- $n_{int}$: total number of intrusion attempts

- $L_{sc}$ takes values of
  - $0.1 \sim 0.001$ for level 1 and
  - $10^{-7} \sim 10^{-9}$ for level 4
2. Reliability: Maturity

- **Definition:** Attributes of software that bear on the frequency of failure by faults in the software.

- IEEE 982.2 - 1988 defines the **Software Maturity Index (SMI).** This index is useful for assessing release readiness when changes, additions, or deletions are made to existing software systems.
2. Reliability: Maturity 

**Measurement:**

\[ SMI = M_t - \left( F_c + F_a + F_d \right) / M_t \]

where

- \( M_t \) is the number of software functions/modules in the current release
- \( F_c \) is the number of functions/modules that contain changes from the previous release
- \( F_a \) is the number of functions/modules that contain additions to the previous release
- \( F_d \) is the number of functions/modules that are deleted from the previous release
2. Reliability: Fault Tolerance

**Definition:** Attributes of software that bear on its ability to maintain a specified level of performance in cases of software faults or of infringement of its specified interface.

**Measurement:**
- Hard to measure or predict
- Ideas:
  - Analyse vulnerability of a system to the occurrence to particular classes of failures with the help of fault-tree models
  - Measure degree of exception handling provision in code (e.g., using *try, catch, throw, finally* and *Exception classes* in Java)
3. Usability (UA)

- **Definition**: The set of attributes that bear on the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users.

- Usability can be defined in terms of the number of available functions \( (n_{af}) \) divided by the number of system required functions \( (n_{rf}) \)

\[
UA = \left( \frac{n_{af}}{n_{rf}} \right) \times 100\%
\]
5. Maintainability

- **Definition:** The set of attributes that bear on the effort needed to make specified modifications.

- It can be defined in terms of the probability that the system can be restored within a given time after a failure.

- Maintainability can be measured by:
  - System recovery time ($T_{rc}$)
  - System service degradation rate ($R_{sd}$)
  - Time to switch ($T_{sw}$)
System Recovery Time \((T_{rc})\)

- **Definition:** System Recovery Time

\[
T_{rc} = \tau_r - \tau_f
\]

- **Unit:** minute or hour or day

- \(\tau_r\) : time to system recovery
- \(\tau_f\) : time to system failure
Service Degradation Rate ($R_{sd}$)

- **Definition:** Service Degradation Time

\[ R_{sd} = \left( \frac{n_{fu}}{N} \right) \times 100\% \]

- $n_{fu}$: number of unrecoverable functions (after maintenance)
- $N$: total number of functions

- **Unit:** normalized 0-100%
Time to Switch ($T_{sw}$)

- **Definition:** Time to Switch

\[ T_{sw} = \tau_{sw} - \tau_f \]

- $\tau_{sw}$: time to stand-by system activated
- $\tau_f$: time to system failure

- **Unit:** minute or hour or day
5. Maintainability: Analyzability

- **Definition:** Attributes of software that bear on the effort needed for diagnosis of deficiencies or causes of failures, or for identification of parts to be modified.

- **Metrics:**
  - Cyclomatic number
  - Number of statements
  - Comments rate

Note: Analyzability is supposed to be increased by development techniques that support “separation of concern” or by Aspect Oriented Software (AOS) development.
5. Maintainability: Stability

- **Definition:** Attributes of software that bear on the risk of unexpected effect of modifications.

- **Metrics:**
  - Number of parameters referenced
  - Number of global variables
  - Number of parameters changed
  - Number of called relationships
5. Maintainability: Changeability

- **Definition:** Attributes of software that bear on the effort needed for modification, fault removal or for environmental change.

- **Metrics:**
  - Number of jumps (go to’s / knots in the CFG)
  - Number of nested levels
  - Average size of statement
  - Number of variables
5. Maintainability: Testability

- **Definition:** Attributes of software that bear on the effort needed for validating the modified software.

- **Metrics:**
  - Number of non-cyclic path
  - Number of nested levels
  - Cyclomatic number
  - Number of call-paths
6. Portability

- **Definition:** The set of attributes that bear on the ability of software to be transferred from one environment to another.

- The principal role of portability metrics is to help characterize the costs and benefits of incorporating portability in a software design, or of porting an existing software unit.
6. Portability

- Total development cost in an environment \( e_1 \) without portability

\[
C_{dev}(req, e_1) = \\
C_{design}(req) + C_{code}(req, e_1) + C_{test}(req, e_1) + C_{doc}(req, e_1)
\]

- The total cost to develop an original portable design plus an implementation for environment \( e_1 \), is:

\[
C_{devp}(req, e_1) = C_{dev}(req, e_1) + C_{pa}(req)
\]

Portability cost
6. Portability

- The cost of redevelopment of a software unit ($su$) with the same specifications, targeted for a new environment $e_2$ is:

$$C_{rdev}(req, e_2) = C_{rdesign}(req) + C_{rcode}(req, e_2) + C_{rtest}(req, e_2) + C_{rdoc}(req, e_2)$$

- Where, usually $Cr_x \leq C_x$

- The cost to port a software unit ($su$) to a new environment is composed primarily of components for manual modification, test, and documentation

$$C_{port}(su, e_2) = C_{mod}(su, e_2) + C_{ptest}(req, e_2) + C_{pdoc}(req, e_2)$$
6. Portability

- In general we expect that $C_{ptest} < C_{rtest}$ and $C_{pdoc} < C_{rdoc}$

- Furthermore, if a portable design has not been developed, it is not unlikely that
  
  $C_{mod} > C_{rdesign} + C_{rcode}$

  since the modifications begin with code designed for a different target, whereas the redevelopment begins with more generic specifications.

- If there is an effective portable design, however, the inequality becomes
  
  $C_{mod} << C_{rdesign} + C_{rcode}$
6. Portability /5

- **Definition:** degree of portability (DP) of a software unit (su).

\[ DP(su) = 1 - \left( \frac{C_{port}(su,e_2)}{C_{rdev}(req,e_2)} \right) \]

- DP has a maximum value of 1, representing perfect portability (\(C_{port} = 0\)).
- Portability is cost-effective if and only if \(DP > 0\).
- To appreciate its significance in a particular case it would be necessary to state the values of \(C_{port}\) and \(C_{rdev}\) as well.
The proposed quality model is very comprehensive, but

- ... it’s not fully clear from the definitions how to precisely distinguish sub-categories
- ... for many of the sub-characteristics the suggested measures are either trivial or hard to apply
A new series of standards is currently under development called **Software Product Quality Requirements and Evaluation (SQuaRE - ISO 25000)**

This series of standards will replace ISO 9126 and 14598 standards

Note: the new standard will replace the word ”metric” by “measure”
Software Quality: Measuring Customer Satisfaction
TQM: Customer Satisfaction

Studies show that

- It is five times more costly to recruit a new customer than it is to keep an old customer and

- Dissatisfied customers tell 7 to 20 people about their experiences, while satisfied customers tell only 3 to 5
Total quality management (TQM) is aimed at long-term business success by linking quality with customer satisfaction.

With ever-increasing market competition, customer focus is the only way to retain the existing customers and to expand market share.
Methods to Collect Data

- How to collect data from customers?
- Three common methods to gather data:
  - *Personal face-to-face interviews*
  - *Telephone interviews*
  - *Mail questionnaires* (self-administered)
- All require a kind of *sampling* to select a subset of the total customers.
Methods to Collect Data /2

Comparison of the data gathering techniques

<table>
<thead>
<tr>
<th>Type of survey</th>
<th>In Person Interview</th>
<th>Phone</th>
<th>Mail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>--</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>Sampling</td>
<td>+--</td>
<td>+</td>
<td>--</td>
</tr>
<tr>
<td>Response Rate</td>
<td>+--</td>
<td>++</td>
<td>--</td>
</tr>
<tr>
<td>Speed</td>
<td>+--</td>
<td>+</td>
<td>--</td>
</tr>
<tr>
<td>Reliability</td>
<td>++</td>
<td>+</td>
<td>--</td>
</tr>
<tr>
<td>Observations</td>
<td>++</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Length of Interview</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Exhibits</td>
<td>+</td>
<td>-</td>
<td>++</td>
</tr>
<tr>
<td>Validity</td>
<td>++</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

Figure from Kan’s Book

- = Disadvantage
-- = Worst
+ = Advantage
++ = Best
+- = Could be either an Advantage or Disadvantage
Sampling Methods

- For any survey, the sampling design is of utmost importance in obtaining unbiased, representative data.

- 4 basic types of probability sampling:
  1. Simple random sampling
  2. Systematic sampling
  3. Stratified sampling
  4. Cluster sampling
1. Simple Random Sampling

- A sample of size $n$ is drawn from a population in such a way that every possible sample of size $n$ has the same chance of being selected.
- To take a simple random sample, each individual in the population must be listed once and only once.
- A mechanical procedure (i.e., using a random number table, or random number generating program) is used to draw the sample.
- To avoid repeated drawing of the same individual, the sampling is done without replacement.
- On each successive draw the probability of an individual being selected increases slightly because there are fewer and fewer individuals left unselected from the population.
- If, on any given draw, the probabilities are equal of all remaining items being selected, then we have a simple random sample.
2. Systematic Sampling

- In systematic sampling instead of using a table of random numbers, one simply goes down a list taking every \( k \)th individual, starting with a randomly selected case among the first \( k \) individuals.

- \( k \) is the ratio between the size of the population and the size of the sample to be drawn. (\( 1/k \) : sampling fraction)

- **Example:** draw a sample of 500 customers from a population of 20,000.
  - \( k = 20000/500 = 40 \)
  - Starting with a random number between 1 and 40 (say, 18), then we would draw every fortieth on the list (58, 98, 138, …).
In a stratified sample items are classified into non-overlapping groups, called *strata*, and then simple random samples are selected from each stratum.

The strata are usually formed based on important variables pertaining to the parameter of interest.

**Example:** customers with high-speed internet systems may have a set of satisfaction criteria for software products that is different from those who have low-speed systems. A stratified sample should include *customer type* as one of the stratification variables.

Stratified samples can be designed to yield greater accuracy for the same cost, or for the same accuracy with less cost.
4. Cluster Sampling

- Sometimes it is much more cost effective to divide the population into a large number of groups, called *clusters*, and to sample among the clusters.

- A cluster sample is a simple random sample in which each sampling unit is a cluster of elements.

- Usually geographical units such as cities, districts, schools, or work plants are used as units for cluster sampling.

- **Example:** if a company has many branch offices throughout the world and an in-depth face-to-face interview with a sample of its customers is desired, then a cluster sample using branch offices as clusters (of customers) may be the best sampling approach.
How large a sample is sufficient?

The answer depends on the **confidence level** we want and the **margin of error** we can tolerate.

---

**Sample Size**

- Smaller error margin
- Larger sample size
- Higher level of confidence

How to bind sample size with customer satisfaction?
Sample Size /2

- Sample size for simple random sampling:
  \[ n = \frac{N_x Z^2 \times p(1-p)}{NB^2 + [Z^2 \times p(1-p)]} \]

- \( N \): population size
- \( Z \): Z statistic from normal distribution:
  - for 80% confidence level, \( Z = 1.28 \)
  - for 85% confidence level, \( Z = 1.45 \)
  - for 90% confidence level, \( Z = 1.65 \)
  - for 95% confidence level, \( Z = 1.96 \)
- \( p \): estimated satisfaction level
- \( B \): margin of error.
Sample sizes for 10,000 customers for 80% - 95% levels of confidence (Z) and 5% and 3% margin of error (B)

<table>
<thead>
<tr>
<th>Expected Customer Satisfaction ($p$)</th>
<th>80% confidence</th>
<th>85% confidence</th>
<th>90% confidence</th>
<th>95% confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>80%</td>
<td>±5% 104</td>
<td>±3% 283</td>
<td>±5% 133</td>
<td>±5% 360</td>
</tr>
<tr>
<td>85%</td>
<td>±3% 83</td>
<td>±5% 227</td>
<td>±3% 106</td>
<td>±3% 289</td>
</tr>
<tr>
<td>90%</td>
<td>±5% 50</td>
<td>±3% 161</td>
<td>±5% 75</td>
<td>±5% 206</td>
</tr>
<tr>
<td>95%</td>
<td>±3% 31</td>
<td>±5% 86</td>
<td>±3% 40</td>
<td>±3% 110</td>
</tr>
</tbody>
</table>
Example

Suppose that you are going to design a questionnaire to be sent to customers who bought your software product. The population size is 10,000 customers and margins of error is supposed to be 5%.

a) Calculate the sample size for the expected customer satisfaction of 90% and confidence factor of 80%.

\[ n = 59 \]

\[ N = 10,000 \]
\[ Z = 1.28 \text{ for 80% confidence level} \]
\[ p = 0.9 \]
\[ B = 0.05 \]

\[ n = \frac{N \cdot Z^2 \times p(1 - p)}{N \cdot B^2 + [Z^2 \times p(1 - p)]} = \frac{10,000 \times (1.28)^2 \times 0.9 \times 0.1}{10,000 \times (0.05)^2 + (1.28)^2 \times 0.9 \times 0.1} \]

\[ n = 59 \]
Example (cont’d)

b) Suppose that you change the sample size to \( n=136 \) and the other parameters except the confidence level remain unchanged. Calculate the new confidence level.

\[
\begin{align*}
N &= 10,000 \\
n &= 136 \\
B &= 0.05 \\
\frac{N Z^2 \times p (1-p)}{N B^2 + \left[Z^2 \times p (1-p)\right]} &= \frac{10,000 \times Z^2 \times 0.09}{10,000 \times (0.05)^2 + Z^2 \times 0.09} \\
Z^2 &= \frac{3400}{887.76} = 3.829 \\
Z &= 1.96
\end{align*}
\]

Confidence level is 95% for this value of \( Z \).
Analyze Satisfaction Data

- The five-point satisfaction scale (very satisfied, satisfied, neutral, dissatisfied, and very dissatisfied) is often used in customer satisfaction surveys.

- *Percent satisfied* and *percent dissatisfied* are common metrics.

- Satisfaction data is collected for the whole software as well as its various attributes, such as CUPRIMDA
  
  U: usability  
  P: performance  
  R: reliability  
  I: installability  
  M: maintainability  
  D: documentation  
  A: availability  
  C: capability
Example

- Analyzing the relationship between satisfaction level with specific attributes and overall satisfaction for a software product.

- Overall customer satisfaction is the dependent variable, and satisfaction levels with CUPRIMDA are the independent variables.

- The purpose of the analysis is to determine the priority for improvement by assessing the extent to which each of the UPRIMDA parameters affects overall customer satisfaction.
## Example: Collected Data

- **Correlation Matrix, Means, and Standard Deviations**

<table>
<thead>
<tr>
<th></th>
<th>Overall</th>
<th>U (usability)</th>
<th>P (performance)</th>
<th>R (reliability)</th>
<th>I (installability)</th>
<th>M (maintainability)</th>
<th>D (documentation)</th>
<th>A (availability)</th>
</tr>
</thead>
<tbody>
<tr>
<td>U (usability)</td>
<td>0.61</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P (performance)</td>
<td>0.43</td>
<td>0.46</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R (reliability)</td>
<td>0.63</td>
<td>0.56</td>
<td>0.42</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I (installability)</td>
<td>0.51</td>
<td>0.57</td>
<td>0.39</td>
<td>0.47</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M (maintainability)</td>
<td>0.40</td>
<td>0.39</td>
<td>0.31</td>
<td>0.40</td>
<td>0.38</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D (documentation)</td>
<td>0.45</td>
<td>0.51</td>
<td>0.34</td>
<td>0.44</td>
<td>0.45</td>
<td>0.35</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A (availability)</td>
<td>0.39</td>
<td>0.39</td>
<td>0.52</td>
<td>0.46</td>
<td>0.32</td>
<td>0.28</td>
<td>0.31</td>
<td></td>
</tr>
</tbody>
</table>

| Mean             | 4.20    | 4.18          | 4.35            | 4.41            | 3.98               | 4.15                 | 3.97              | 4.57            |
| Standard deviation| 0.75    | 0.78          | 0.75            | 0.66            | 0.90               | 0.82                 | 0.89              | 0.64            |
| % Satisfaction   | 85.5    | 84.1          | 91.1            | 93.8            | 75.3               | 82.9                 | 73.3              | 94.5            |

*Example from Kan’s Book*

- Lowest satisfaction
Using regression analysis one can determine the degree of contribution of each attribute (i.e., regression ratio) to the overall customer satisfaction.

Odds ratio indicates relative importance of the attribute in the model.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Regression Coefficient</th>
<th>Odds Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>R (reliability)</td>
<td>1.216</td>
<td>11.4</td>
</tr>
<tr>
<td>U (usability)</td>
<td>0.701</td>
<td>4.1</td>
</tr>
<tr>
<td>A (availability)</td>
<td>0.481</td>
<td>2.6</td>
</tr>
<tr>
<td>I (installability)</td>
<td>0.410</td>
<td>2.3</td>
</tr>
<tr>
<td>M (maintainability)</td>
<td>0.376</td>
<td>2.1</td>
</tr>
<tr>
<td>P (performance)</td>
<td>0.321</td>
<td>1.9</td>
</tr>
<tr>
<td>D (documentation)</td>
<td>0.164</td>
<td>1.4</td>
</tr>
</tbody>
</table>

Example from Kan’s Book
Example: Visualize Results

Priority

Candidates for further improvement

Example from Kan’s Book
Example: Improvement Priority

- How to determine the priority of improvement among the specific quality attributes?
  - Determine the order of significance of each quality attribute on overall satisfaction by statistical modeling (such as the regression model in the example).
  - Plot the coefficient of each attribute from the model (Y-axis) against its satisfaction level (X-axis).
  - Use the plot to determine priority by:
    - Going from top to bottom, and
    - Going from left to right, if the coefficients of importance have the same values.
Question: How Good is Good Enough?

- How much customer satisfaction is good enough?
- Should my company invest $2,000,000 to improve satisfaction from 85% to 90%?
- Given that my company’s customer satisfaction is at 95%, should I invest another million dollars to improve it further or should I do this later?
Babich’s Study /1

Babich (1992) Model based on a simplified model of customer satisfaction and market share that contains only three companies: A, B, and C. When customers are dissatisfied with company A, they choose company B or C, etc.

\[ A_{t+1} = A_t (1 - x) + B_t y \left( \frac{A_t}{A_t + C_t} \right) + C_t z \left( \frac{A_t}{A_t + B_t} \right) + G \left( \frac{A_t}{A_t + B_t + C_t} \right) \]

\[ B_{t+1} = B_t (1 - y) + A_t x \left( \frac{B_t}{B_t + C_t} \right) + C_t z \left( \frac{B_t}{A_t + B_t} \right) + G \left( \frac{B_t}{A_t + B_t + C_t} \right) \]

\[ C_{t+1} = C_t (1 - z) + A_t x \left( \frac{C_t}{B_t + C_t} \right) + B_t y \left( \frac{C_t}{A_t + C_t} \right) + G \left( \frac{C_t}{A_t + B_t + C_t} \right) \]
Babich’s Study /2

- A : number of A customers
- B : number of B customers
- C : number of C customers
- G : number of new customers to market
- x : dissatisfaction level with A products
- y : dissatisfaction level with B products
- z : dissatisfaction level with C products
- t : time
Babich’s Study /3

- Market shares of the three companies assuming satisfaction levels of 95%, 91%, and 90% for A, B, and C, respectively, over a number of time periods and equal initial market share.

- After 12 time periods the 95% satisfaction product (company A) would basically own the market.

Figure from Kan’s Book
If the satisfaction levels of companies B and C were 98% and 99%, respectively, and company A’s satisfaction level remained at 95%, company A’s product would have less than 10% market share in 24 time periods.
Answer: How Good is Good Enough?

- From Babich’s simple model and examples the answer is:
  
  *You have to be better than your competitors!*

- Therefore, it is important to measure not only one’s customer satisfaction level, but also the competitors satisfaction level.

- Indeed, many companies have been doing exactly that.
Software Quality: Software Quality Assurance (SQA)
Dependable Software

- Dependable software products are those that run correctly and consistently, have few remaining defects, handle abnormal situations properly, and need few installation effort.

- Developing dependable software (usually) requires:
  - Devising Software Quality Assurance (SQA) program
  - Establishing Software Reliability Engineering (SRE) process
Quality Assurance: Definition

ISO 12207:

“All the planned and systematic activities implemented within the quality system, and demonstrated as needed, to provide adequate confidence that an entity will fulfill requirements for quality”

- Software quality Assurance (SQA) is a planned and systematic approach to ensure that both software process and software product conform to the established standards, processes, and procedures.
- The goals of SQA are to improve software quality by monitoring both software and the development process to ensure full compliance with the established standards and procedures.
Quality Assurance: Purposes

- **Internal:**
  - within an organization, quality assurance provides confidence to management

- **External:**
  - in contractual situations, quality assurance provides confidence to the customer or others.
Quality Assurance: Types

- **Constructive QA:**
  - Creation of software quality
  - Avoidance of quality-relevant risks
  - Design of software such that it becomes testable (built-in testing)

- **Analytical QA:**
  - Checking/assessing software quality (inspections and tests)
  - Mitigation of quality-relevant risks
  - Correction of software as soon as possible
Constructive QA

How to do Constructive QA?

- Selection and implementatation of SW Process Models
  - includes the planning of all QA relevant activities

- Applications of SW Engineering principles in all development steps: Abstraction, Hierarchisation, Modularisation, Formalization, Automization, ...
  - e.g. Requirements: Explicit documentation of requirements involving all possible relevant views (→ UML: Use Cases)
  - e.g. Design: Low coupling, High cohesion, ...
  - e.g. Implementation: Low structural complexity, programming guidelines, reuse, ...
Analytic QA

How to do Analytic QA?

- Static analyses (complexity, nesting levels, structuredness, etc.)
- Verification (inspections, tests [development perspective])
- Validation (tests [user perspective])
SQA: Role

- The role of SQA is to give management the assurance that the officially established process is actually being implemented.

- SQA ensures that:
  - An appropriate development methodology is in place.
  - The projects use standards and procedures in their work.
  - Reviews and audits are conducted.
  - Documents are produced to support maintenance and enhancement.
  - Software configuration management is set up to control change.
  - Testing is performed and passed.
  - Deficiencies and deviations are identified, documented and brought to management’s attention.
SQA: Goals

The goals of SQA is to reduce the risks by:

- Appropriately monitoring the software and the development process.
- Ensuring full compliance with standards and procedures for software and process.
- Ensuring that inadequacies in product, process, or standards are brought to management’s attention so that they can be fixed.

SQA is not responsible for producing quality products. It is responsible for auditing the quality actions and for alerting management to any deviations.
SQA: Responsibilities

To achieve its goals, SQA is responsible for:

- Review all development and quality plans for completeness.
- Participate as inspection moderators in design and code inspections.
- Review all test plans for adherence to standards.
- Review samples of all test results to determine adherence to plans.
- Periodically audit the performance to determine adherence to standards.
- Participate in all project phase reviews and write down any nonconformance.
Establishing SQA Program

How to establish SQA program?

1. Quality Planning
   - Documentation
   - Standards and Procedures
2. Quality Monitoring and Control
3. Quality Assessment (Evaluation)
4. Quality Improvement
Creating SQA Plan /1

- SQA plan specifies its goals, tasks to be performed, and the standards and procedures against which the development work is to be measured.

- IEEE standard for SQA plan preparation contains the following:
  - Purpose
  - Reference Documents
  - Management
  - Documentation
  - Standards, Practices, and Conventions
  - Reviews and Audits
IEEE standard for SQA plan preparation contains the following (contd.):

- Software Configuration Management
- Problem Reporting and Corrective Action
- Tools, Techniques, and Methodologies
- Code Control
- Media Control
- Supplier Control
- Records Collection, Maintenance, and Retention
SQA: Documentation

- Documentation step describes the documents to be produced and how it is to be previewed.
- Documentation includes:
  - *Software requirement specification*: specifies each software function, performance parameter, interface, or other attribute with sufficient precision to permit its verification.
  - *Software design description*: describes the major components, databases, and internal interfaces.
SQA: Documentation

- **Software verification and validation plan**: describes the methods used to verify that the requirements are implemented in the design, that the design is implemented in the code, and that the code meets the requirements.

- **Software verification and validation report**: reports on the SQA verification and validation activities.

- **User Documentation**: describes the required installation, operation, and maintenance of the software.

- **Other documents**: include software development plan, software configuration management plan, standards and procedures manual.

- **Review plan**: describes the planned review methods.
SQA: Standards & Procedures

- **Standards** are the criteria to which software products are compared \((i.e.,\) standards define what should be done).  
  
- **Procedures** are the criteria to which development and control processes are compared \((i.e.,\) procedures define how the work is actually to be done, by whom, when and what is done with the results).
SQA: Standards & Procedures

- Minimum requirement for standards include:
  - **Documentation Standards:** specify form and content for planning, control, and product documentation and provide consistency throughout a project.
  - **Design Standards:** specify the form and content of the design product. They provide rules for translating the software requirements into the software design and for representing it in the design documentation.
  - **Code Standards:** specify the language in which the code is to be written and define any restrictions on use of language features. They define legal language structure, style conventions, rules for data structures, and internal code documentation.
SQA: Activities

SQA activities include:

- **Production evaluation** is to ensure that standards are followed. It assures that clear and achievable standards exist and evaluate compliance of software product with the standards.

- **Process monitoring** is to ensure that appropriate steps to carry out process are being followed. SQA monitors processes by comparing actual steps performed with established procedures.

- **Audit** looks at a process or product in depth, comparing them with established standards and procedures.
SQA: Project Specific Concerns

- Each project has its specific attributes and SQA program should be tailored to accommodate to the project needs.

- Common project specific characteristics are:
  - Mission critical of project
  - Schedule and budget
  - Size and complexity of project
  - Size and complexity of project staff organization
SQA: Project Specific Concerns

- The relationship between SQA program and mission critical level is straightforward. The more critical the project, the more important and formal the SQA.

- The relationship between SQA and budget and schedule is not so intuitive; the tighter the budget and schedule, the more critical it is to have a well planned and effective SQA program.

- The project organization structure influences the SQA program.
  - For large or dispersed staff, more formal SQA program is required.
  - A small, centralized development staff can easily inform each other of the nonconformance and helping each other in meeting standards.
SQA: Other Considerations

- To make SQA program successful, the first step is to get senior management commit to it. Without management support, SQA can not be effective.
- Even with senior managers support experienced SQA staff is a necessity.
- The hardest problems the software managers face is to get experiences staff into SQA. Good engineers are usually preferred to stay with the development group. Usually, software development transfers its poorer performers to SQA and not taking them back. Rotation scheme may be effective.
Demming’s 14 Quality Points

1. Create constancy of purpose for improvement of product or service.
2. Adopt the new philosophy.
3. Cease dependence on inspection to achieve quality.
4. End the practice of awarding business on the basis of price tag alone. Instead, minimize total cost by working with a single supplier.
5. Improve constantly and forever every process for planning, production and service.
6. Institute training on the job.
7. Adopt and institute leadership.
Demming’s 14 Quality Points

8. Drive out fear.
9. Break down barriers between staff areas.
10. Eliminate slogans, exhortations, and targets for the work force.
11. Eliminate numerical quotas for the work force and numerical goals for management.
12. Remove barriers that rob people of pride of workmanship. Eliminate the annual rating or merit system.
13. Institute a vigorous program of education and self-improvement for everyone.
14. Put everybody in the company to work to accomplish the transformation.

W. E. Deming (1900-1993) The Father of Modern Quality
TQM: Total Quality Management

- TQM is a style of management aiming at achieving “long-term” success by linking quality with customer satisfaction.

- Other variations:
  - Total Quality Control (HP)
  - Market Driven Quality (IBM)
  - Experience Factory (Basili)

Figure from Kan’s Book
Practice of TQM: Essentials

- Focus on quality
- Cooperate with customers
- Continuously improve development process
- Encourage constructive critics and empower employees
- Use the problem solving/problem prevention cycle
- Use measurements to back decisions
Experience Factory Organization
(Basili, Rombach, 1988)

QIP
(Basili, Rombach, 1988)

Measurement and Learning Organizations

Data, Lessons learned, ...

Experience Base

Project Organization

Support Organization
(Experience Factory)

Measurement data is made reusable

package characterize

analyze set goals

execute project choose models

Measurement is performed Measurement is planned

far@ucalgary.ca
References

Standards:


Software Quality:


far@ucalgary.ca