Software Product Quality

Each team member, in the individual report, will be expected to highlight Quality Criteria for their section of the code.

Aspects of Quality

- Quality is not absolute
- Quality is multidimensional
- Quality is subject to constraints
- Quality is about acceptable compromises
- Quality criteria are not independent, but interact with each other causing conflicts.

Defining Quality

'The totality of features and characteristics of a product or service that bear on its ability to satisfy specified or implied needs.' (ISO, 1986)

- This associates quality with the ability of the product or service to fulfil its function.
- This is achieved through the features and characteristics of the product.
- Quality is associated both with having the required range of attributes and achieving satisfactory performance within each attribute.

Evaluating Quality

- Hierarchical Quality Models
  - Both Boehm and McCall produced hierarchical quality models in the late 1970's
  - Although Boehm's set of criteria is larger than McCall's many characteristics are closely related
  - Both models were used as the basis of ISO 9126 Standard - *Software Product Evaluation: Quality Characteristics and Guidelines for their Use*

McCall's Quality Model

- McCall's model [1] is aimed at system developers to be used during the development process

- Based on three classes of software work:
  - Product Operation – factors affecting normal operation of software
  - Product Revision – factors affecting error correction & adaptation
  - Product Transition – factors affecting ease of moving to new environment
- Structure of Hierarchical Quality Model
  - High level qualities – Quality Factors (Characteristics)
  - Each factor can be decomposed into further criteria (attributes)
  - Quality Criteria may be determined through metrics

External attributes – can only be measured in relation to environment

Can only be measured indirectly.

Various metrics possible, depend on criterion.

Can be measured.

McCall's Quality Model:
Quality Factors of Product Operation

- **Correctness** - the extent to which a program fulfils its specification.
  - Completeness: The degree to which a full implementation of the required functionality has been achieved
    - Use Validation Testing to check program meets all functional requirements
  - Consistency: Use of uniform design and implementation techniques and notations throughout a project
    - Adopt same programming style, consistent interface design
  - Traceability: Ability to link software components to requirements

McCall's Quality Model:

Quality Factors of Product Operation

- **Reliability** - its ability not to fail.
  - **Accuracy:** The precision of computations and output
    - Check in Unit Tests
    - Check for common errors
  - **Consistency:** Use of uniform design, implementation, and notations
  - **Error tolerance:** The degree to which continuity of operation is ensured under adverse conditions
    - Exception handling to ensure program can continue
  - **Simplicity:** The ease with which the software can be understood
    - Keep code simple and well structured
    - Follow a good programming style

McCall's Quality Model:

Quality Factors of Product Operation

- **Efficiency** – the amount of computing resources and code required by a program to perform its function.
  - **Execution efficiency:** Run-time efficiency of the software
  - **Storage efficiency:** Run-time storage requirements of the software

McCall's Quality Model:

Quality Factors of Product Operation

- **Integrity** - the protection of the program from unauthorised access.
  - **Access audit:** Ease with which software and data can be checked for compliance with standards or other requirements
  - **Access control:** Provisions for control & protection of software & data

McCall's Quality Model:

Quality Factors of Product Operation

- **Usability** – Effort required to learn, operate, prepare input and interpret output of a program.
  - **Communicativeness:** Provision of useful inputs and outputs that can be easily assimilated.
    - Use Schneiderman's principles & guidelines when designing interfaces [2]
  - **Operability:** Ease of operation of the software
    - Good use of widgets to minimise user input
    - Anticipate main user errors
  - **Training:** Ease with which new users can be made to use the system
    - Use widgets which are familiar to user and appropriate to the task

McCall's Quality Model:

Quality Factors of Product Revision

- **Maintainability** - the effort required to locate and fix a fault in an operating program.
  - **Conciseness:** Compactness of the source code, in terms of lines of code
    - Python as a scripting language needs less code to be developed
  - **Modularity:** Provision of highly independent modules
    - Classes – all attributes and methods relate only to class
    - Functions & methods perform one function only
    - Pass only simple data as parameters to ensure loose coupling
  - **Self-documentation:** Provision of in-line documentation that explains the implementation of components
    - Self documenting code, good naming conventions, comments state why not what, docstrings
  - *Consistency and Simplicity*

McCall's Quality Model:

Quality Factors of Product Revision

- **Testability** - the ease of testing the program, to ensure that it is error-free and meets its specification.
  - **Instrumentation:** Degree to which the software provides for measurements of its use or identification of errors.
    - Automated unit test code at end of modules
  - **Modularity, Self-documentation and Simplicity**

McCall's Quality Model:

Quality Factors of Product Revision

- **Flexibility** - the ease of making changes required in an operating program.
  - **Expandability:** Degree to which storage requirements or software functions can be expanded
  - **Generality:** Breadth of the potential application of software components
  - **Modularity and Self-documentation**

McCall's Quality Model:

Quality Factors of Product Transition

- **Portability** - the effort required to transfer a program from one environment to another.
  - **Hardware independence:** Degree to which the software is dependent on the underlying hardware
  - **Software system independence:** Degree to which the software is independent of its software environment – operating system, libraries, database management system…
  - **Modularity**
  - **Self-documentation:** In-line documentation

McCall's Quality Model:

Quality Factors of Product Transition

- **Reusability** - the ease of reusing software in a different context.
  - **Generality:** Breadth of potential application of components
  - **Hardware independence**
  - **Software system independence**
  - **Modularity**
  - **Self-documentation**

McCall's Quality Model:
Quality Factors of Product Transition

- **Interoperability** - the effort required to couple the system to another system.
  - **Communication commonality:** Degree to which standard protocols and interfaces are used
  - **Data commonality:** Use of standard data representations
  - **Modularity**

	Correctness	Reliability	Efficiency	Integrity	Usability	Maintainability	Testability	Flexibility	Portability	Reusability	Interoperability
Access Audit				■							
Access Control				■							
Accuracy		■									
Comm. Commonality											■
Completeness	■										
Communi-cativeness					■						
Conciseness						■					
Consistency	■	■				■					
Data Commonality											■
Error Tolerance	■										
Execution Efficiency			■								
Expandability								■			
Generality								■		■	
Hardware Independence									■	■	
Instrumentation							■				
Modularity						■	■	■	■	■	■
Operability					■						
Self-documentation						■	■	■	■	■	
Simplicity		■				■	■				
SW System Independence									■	■	
Storage Efficiency			■								
Traceability	■										
Training					■						

Adapted from [3]

Problems

- Many criteria cannot be measured objectively
- Many measures are subjective
- Objective measures exist for e.g. efficiency and reliability
- How do you measure?
- When do you measure?
  - Before or after system is built?
- Completeness of taxonomy
- Quality factors overlap → trade-offs between factors

Trade-offs

Adapted from [3]

References

[1] McCall, J. A., Richards, P. K., and Walters, G. F., "Factors in Software Quality", Nat'l Tech.Information Service, no. Vol. 1, 2 and 3, 1977.

[2] Ben Shneiderman *"Designing the User Interface - Strategies for effective Human-Computer Interaction"* Addison-Wesley

[3] Hans van Vliet, "Software Engineering. Principles and Practice", Wiley 1993