



A SURVEY BASED SOFTWARE QUALITY MODEL



Authors

Vishal Sadana

Department of Computer Science

University of Missouri-Rolla

Rolla, MO 65401

Email: vs7b2@umr.edu



Presentation Outline

- What is Software Quality
- Importance and Measurement of Software Quality
- Types of Software Quality Metrics
- Quality Models and Types of Quality Models
- Description of the Proposed Metric
- Improvement in Quality using the proposed metric
- Conclusion
- Future Work
- References



Defining Software Quality

- **Roger Pressman** has defined software quality as “Conformance to explicitly defined functional and implicit characteristics that are expected of professionally developed software”
- **Al Davis** defines quality as “Quality isn’t about zero defects or measurable improvements in defect rates, nor is it about meeting documented requirements. Its no more and no less that satisfying customer needs (whether or not the needs are properly documented)



Defining Software Quality (Contd.)

- **Robert Glass** writes “Quality is a collection of ‘ilities’”:
 - **Reliability** – the ability to operate free
 - **Modifiability** – the ability to adapt changes
 - **Understandability**- the ability to understand the software
 - **Efficiency**- the speed of the software
 - **Usability**- the ability to use the software easily
 - **Testability**- the ability to construct and execute test cases easily
 - **Portability** – the ability to move the software



Defining Software Quality (Contd.)

- **Gerald Weinberg** mentioned “Quality is defined as value to some person. The tricky bit is working out who that person should be. This is how a User’s high quality system can be low quality to someone else. It must be first decided who is measuring it and then decide how they assess quality”



Importance of Software Quality

- Software is now used in many demanding applications and software defects have caused serious damage and even physical harm. These software can be
 - Software to fly airplanes or to drive automobiles
 - Software to control air traffic, run factories or operate power plants

People have been killed by defective software



Measurement of Software Quality

- The aim of software quality engineering is to investigate the relationships among in-process metrics, project characteristics, and end-product quality
- In general, software quality metrics are more closely associated with process and product metrics



Product Quality Metrics

- **Product Quality** is usually measured by the number of “bugs” in the software or by how long the software can run before encountering a “crash”
 - Mean Time To Failure
 - Defect Density Metric



Mean Time To Failure

- The MTTF metric is often used with the safety critical systems such as airline traffic control systems
- An estimate of the average, or mean time until a design's component's first failure or disruption in the operation of the product occurs
- It is a useful measurement of Reliability



Defect Density Metric

- The most commonly used means of measuring software quality of commercial software
- It is the number of defects over the opportunities for error or size

$$\text{Defect Density Metric} = \frac{\text{No. of defects in a Code}}{\text{Size of the Code}}$$

- It is a better indicator of testing severity than quality



Process Quality Metrics

- **Process Quality Metrics** simply means tracking defect arrival during formal machine testing (testing after code is integrated)
 - Defect Density During Machine Testing
 - Defect Arrival Pattern During Machine Testing
 - Phase-Based Defect Removal Pattern



Defect Density During Machine Testing

- Highly useful to monitor the quality of subsequent releases of a product in the same development organization
- Higher defect rates found during testing indicates that the software has experienced higher error injection during its development process
- Release-to-Release comparisons are not affected by the external factors



Defect Density During Machine Testing (Contd.)

- If the defect rate during testing is the same or lower than that of the previous release, then ask
 - Does the testing for the current release deteriorate?
- If the defect rate during testing is significantly higher than that of the previous release
 - Did we work to improve the quality?



Defect Arrival Pattern During Machine Testing

- The objective is to look for defect arrivals or times between failures
- This metric can be further divided as
 - All the defects reported may not be valid defects, hence, remove the redundant defects
 - Record the pattern of valid defect arrivals
 - Fix the bug before the code is tested once again



Phase-Based Defect Removal Pattern

- This is an extension to the test defect density pattern
- Requires tracking of defects at all phases of the development cycle, including the design reviews, code inspections and formal verification
- As large number of defect in the code result due to design flaws, conducting formal reviews enhances the defect removal capability, thus, producing minimum errors



Defining a Quality Model

- **Azuma** has defined a Quality Model as “the set of characteristics and the relationships between them which provide the basis for specifying quality requirements and evaluating quality”



Types of Quality Models

- The McCall Model
- The Boehm Model
- The FURPS Model
- The ISO 9126 Model
- The Dromey Model



The McCall's Model (1977)

- The McCall Model aimed at system developers
- It is used during development process
- It identifies 3 areas of software work



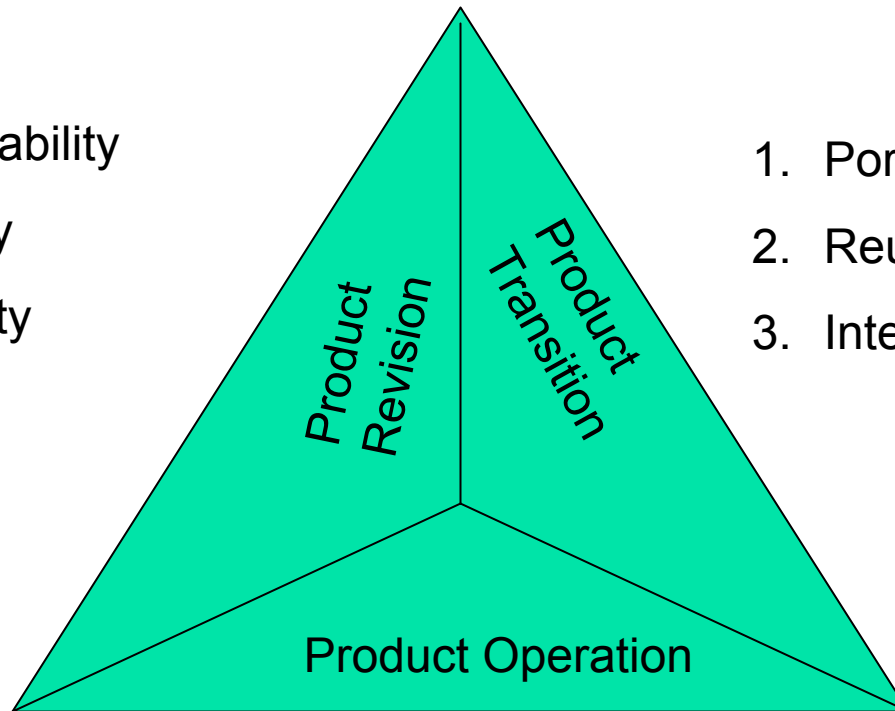
The McCall's Model (Contd.)

- **Product Operation** – refers to the product's ability to be quickly understood, efficiently operated and capable of providing the results required by the user
- **Production Revision** - is related to error correction and system adaptation
- **Product Transition** – distributed processing, rapid change in hardware

The McCall's Model (Contd.)

1. Maintainability
2. Flexibility
3. Testability

1. Portability
2. Reusability
3. Interoperability



1. Correctness
2. Reliability
3. Efficiency
4. Integrity
5. Usability



The McCall's Model (Contd.)

- **Usability** is the ease of use of the software *Can I run it?*
- **Integrity** is the protection of the program *Is it secure?*
- **Efficiency** is concerned with the use of resources *Will it run on my machine well?*
- **Correctness** is the fulfillment of specifications *Does it do what I want?*
- **Reliability** is the ability not to fail *Does it do accurately all the time?*



The McCall's Model (Contd.)

- **Maintainability** is the effort to locate and fix the fault *Can I fix it?*
- **Flexibility** is the ease of making changes *Can I change it?*
- **Testability** is the ease of testing the program *Can I test it?*
- **Portability** is the effort required to transfer from one environment to another *Will I be able to use it on another machine?*
- **Reusability** is the ease of reusing the s/w *Can I use it again?*



The McCall's Model (Contd.)

- Doesn't say much about Functionality



The Boehm's Model

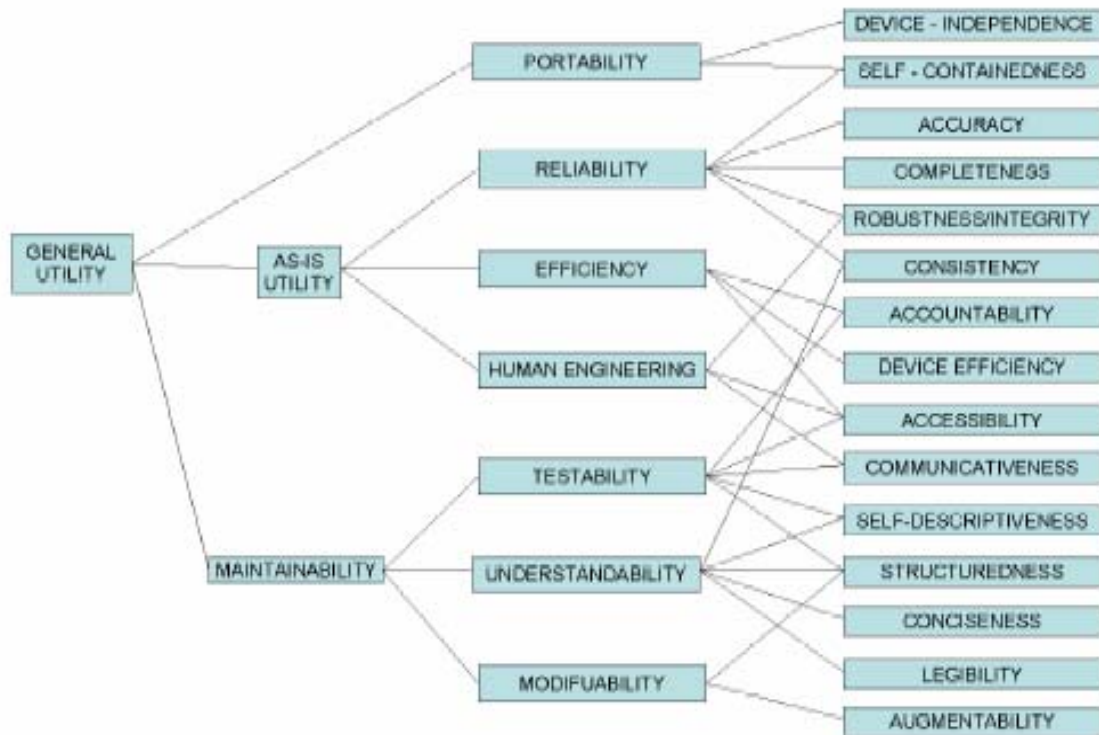
- This model was originally proposed in 1973 by Boehm and his coworkers used a basic outline for relating measurable to the aspects of quality
- This model asserts that QUALITY software satisfies the needs of the users, designers, testers and maintainers
- Measures were used to determine to assess the depth of agreement with a particular criterion that affects the fundamental quality factors



The Boehm's Model (Contd.)

- Factors included
 - Portability
 - Reliability
 - Efficiency
 - Human Engineering
 - Testability
 - Understandability
 - Modifiability

The Boehm's Model (Contd.)





The Boehm's Model (Contd.)

- Emphasis on the maintainability of a software product
- Includes considerations involved in the evaluation of a software product with respect to the utility of the program



The FURPS Model

- This model was proposed by Robert Grady and Hewlett-Packard Co. decomposes characteristics into two different categories of requirements:
 - **Functional Requirements (F):** Defined by input and expected output
 - **Non-functional Requirements (NF):** Usability, Reliability, Performance, Supportability



The FURPS Model (Contd.)

- Does not take in to account the portability of a software product



The Dromey's Model

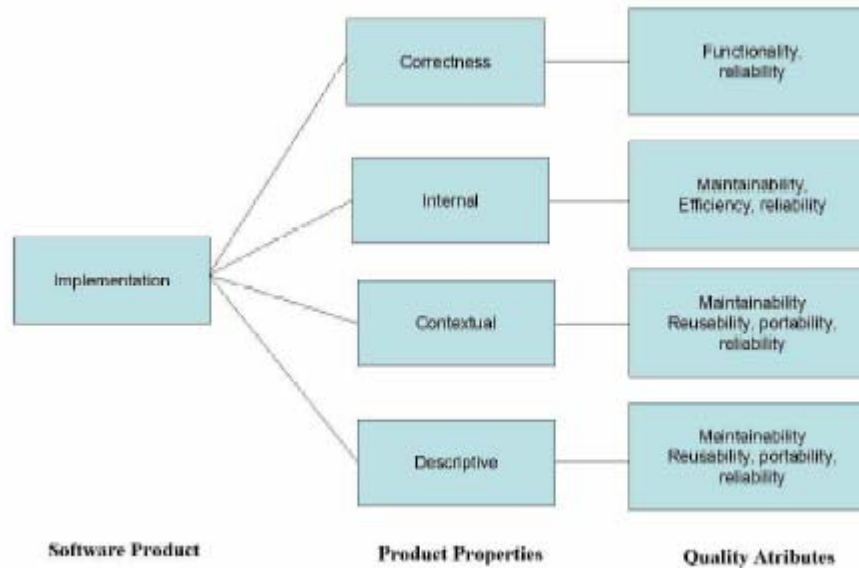
- Proposed in 1996 because higher level quality attributes cannot be measured or built directly
- Higher level quality can be achieved by building components which represent a complete set of a product's properties
- This model points the properties of the software product that affects the attributes of quality



The Dromey's Model

- Dromey mentioned that high-level quality attributes, such as Reliability and Maintainability cannot be built into the software
- Therefore, identify a set of properties so as to cover these requirements

The Dromey's Model (Contd.)





The Dromey's Model (Contd.)

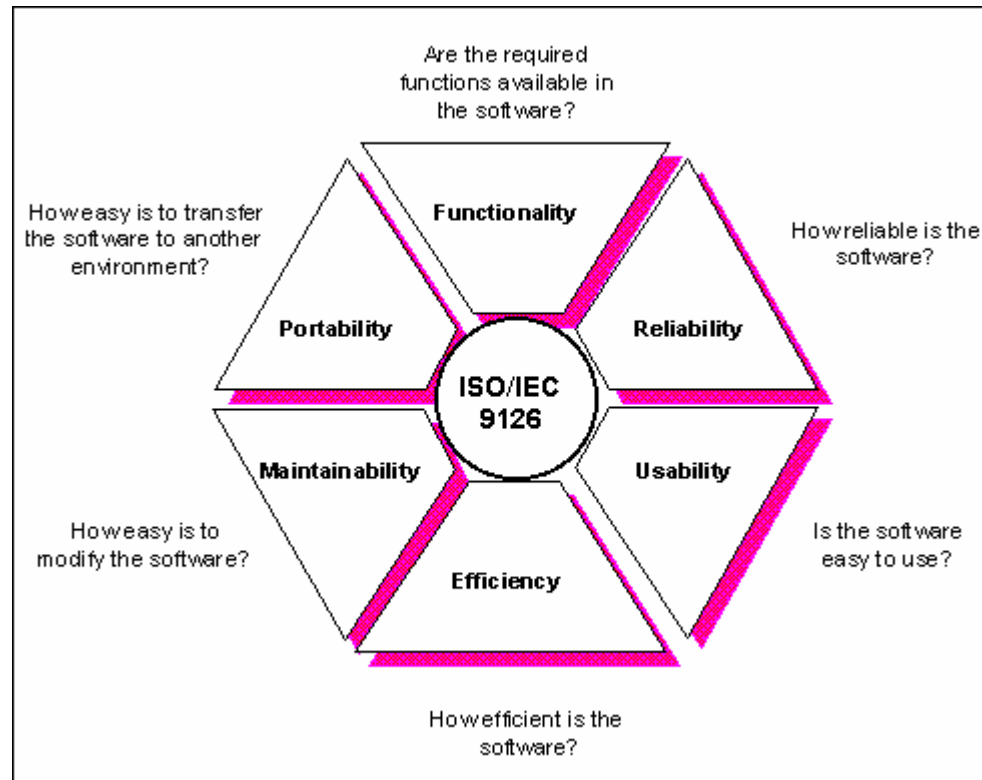
- This model does not consider the efficiency of a software to determine the quality of a software



The ISO 9126 Model

- The ISO 9126 model was proposed in 1991 and defines the product quality as a set of product characteristics, these are:
 - **External characteristics:** How the product works in its environment e.g. Usability, Reliability
 - **Internal characteristics:** How the product was developed e.g. size, test and failure rate

The ISO 9126 Model (Contd.)





The ISO 9126 Model (Contd.)

- **Functionality**
 - Suitability, Accuracy, Interoperability, Compliance, Security
- **Reliability**
 - Maturity, Fault Tolerance, Recoverability
- **Usability**
 - Understandability, Learnability, Operability
- **Efficiency**
 - Time Behavior, Resource Behavior
- **Portability**
 - Adaptability, Changeability, Stability, Testability



Comparison of Quality Models

Quality Characteristics	Boehm	McCall	FURPS	ISO 9126	Dromey
Testability	x	x		x	
Correctness		x			
Efficiency	x	x	x	x	x
Understandability	x			x	
Reliability	x	x	x	x	x
Flexibility		x	x (extensibility, adaptability, maintainability)		
Functionality			x	x	x
Human Engineering	x				
Integrity		x		x(Security)	
Interoperability		x		x(Functionality)	
Process Maturity					x
Maintainability	x	x	x	x	x
Changeability	x				
Portability	x	x		x	x
Reusability		x			x



Our Metric

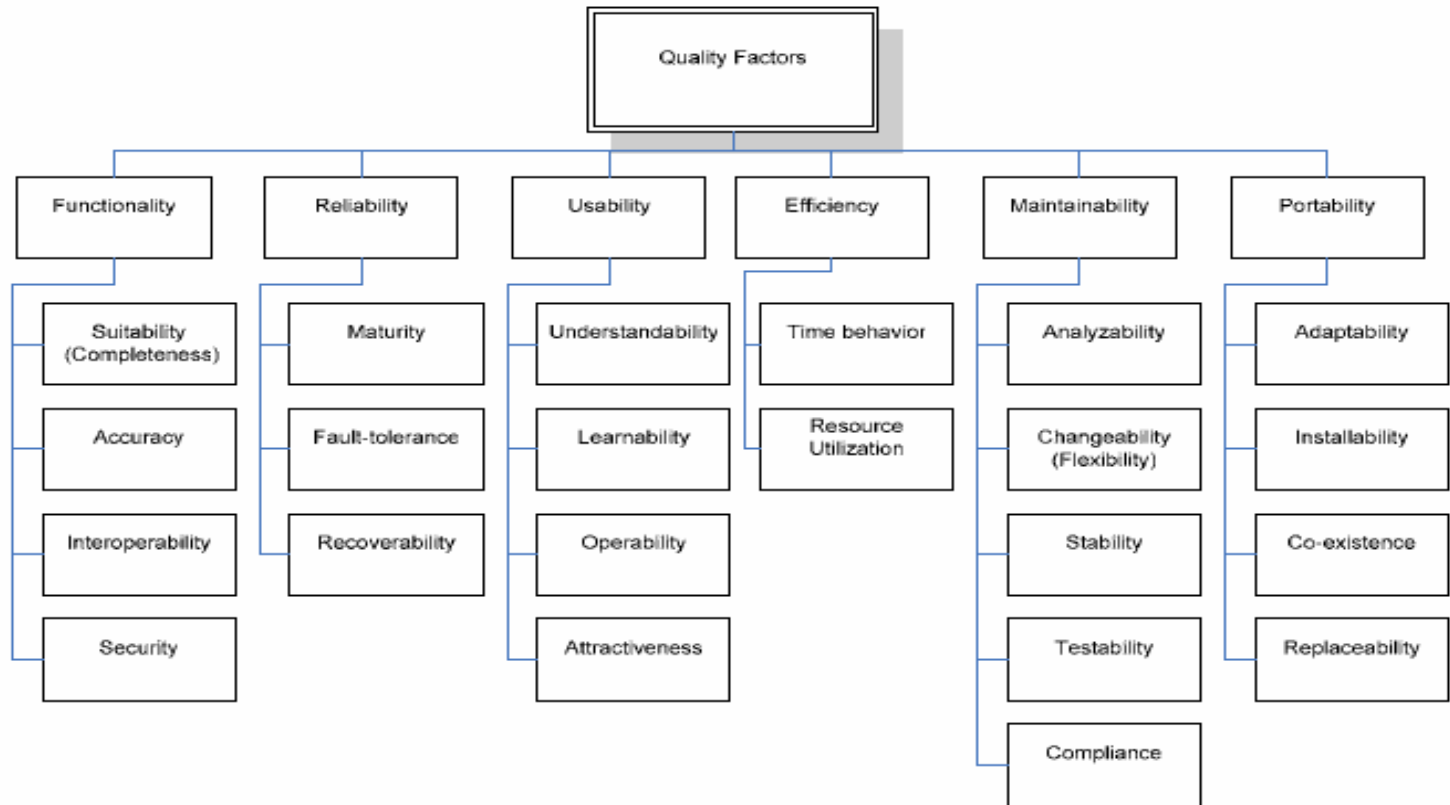
- An End-Product Quality Metric
- Users plays a role in the measurement of Software Quality
- Users have a direct and equal impact on the software quality
- Produces a unique QF for all software and all versions of a software
- To balance the user's point of view and to determine the overall quality, QM's and QA team members also evaluate the software
- Can be used to improve the quality of later versions of the software



Calculating the Quality Factor

- ISO 9126 Quality Model- An Internationally recognized model and being followed by majority of the organization around the world
- During the Requirement Analysis phase, categorize the requirements as per the characteristic and the sub-characteristic in the ISO 9126 Model
- Provide the Questionnaire to the Users, QM's and QA team members
- The questionnaires are collected and consolidated
- Quality Factor is obtained by using a series of formulas
- Categorize the software based on the value of QF

The ISO 9126 Quality Model





Defining the sub-characteristics in the ISO 9126 Quality Model

CHARACTERISTI NAME	SUB-CHARACTERISTIC NAME	DEFINTION
FUNCTIONALITY	SUITABILITY	The capability of the software to provide an appropriate set of functions for specified tasks and user objectives
	ACCURACY	The capability of the software to provide the right or agreed results
	INTEROPERABILITY	The capability of the software to interact with one or more specified systems
	SECURITY	The capability of the software to protect information and data from unauthorized access
RELIABILITY	MATURITY	The capability of the software product to avoid failure as a result of faults in the software
	FAULT TOLERANCE	The capability of the software to maintain a specified level of performance in cases of software faults
	RECOVERABILITY	The capability of the software to re-establish a specified level of performance and recover the data in case of a failure



Defining the sub-characteristics in the ISO 9126 Quality Model (Contd.)

USABILITY	UNDERSTNDABILITY	The capability of the software product to be understood, learned, used and appear attractive to the user
	LEARNABILITY	The capability of the software to enable the user to learn its application
	OPERABILITY	The capability of the software to enable the user to operate and control it
	ATTRACTIVENESS	The capability of the software to be attractive to the user
EFFICIENCY	TIME BEHAVIOR	The capability of the software to provide appropriate response while performing its function
	RESOURCE UTILISATION	The capability of the software product to use appropriate amounts and types of resources



Defining the sub-characteristics in the ISO 9126 Quality Model (Contd.)

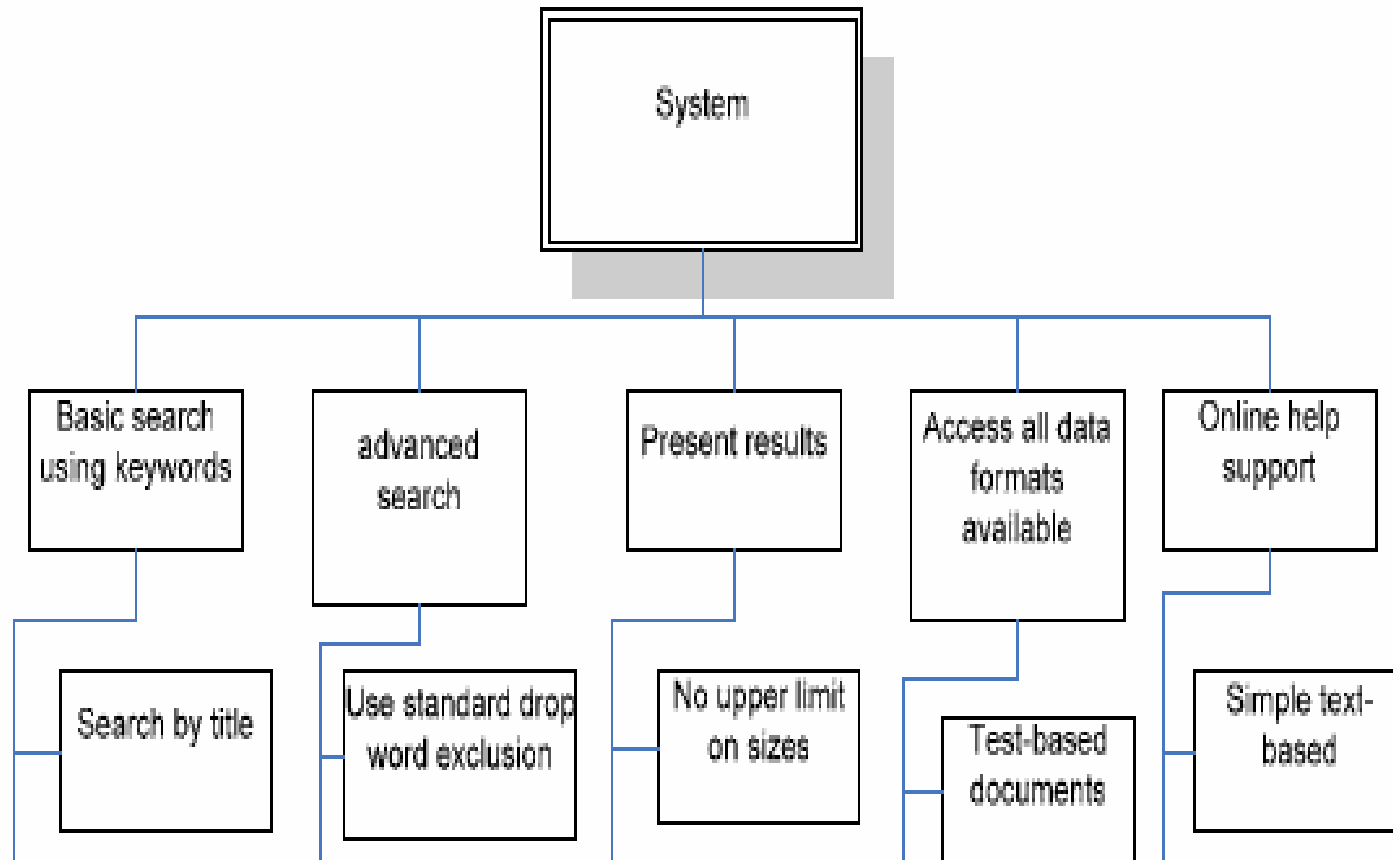
MAINTAINABILITY	EFFICIENCY	The capability of the software to follow standards of efficiency
	ANALYSABILITY	The capability of the software to be diagnosed for causes of failures in the software, or for the parts to be modified to be identified
	CHANGEABILITY	The capability of the software to enable a specified modification to be implemented
	STABILITY	The capability of the software to avoid unexpected effects from changes made in the software
	TESTABILITY	The capability of the software product to enable modified software to be validated
PORTABILITY	ADAPTABILITY	The capability of the software to be adapted for different environments
	INSTALLABILITY	The capability of the software to be installed in a specified environment
	CO-EXISTENCE	The capability of the software to co-exist with other independent software
	REPLACEABILITY	The capability of the software to be used in place of another software for the same purpose



Categorization of Requirements

- We have taken an example of a Search Engine for this phase.
- The collected requirements are categorized according to the characteristic and sub-characteristic to which they belong
- Provide the list of categorized documents to the evaluators

Functional Hierarchy of a Search Engine





Categorized Requirements

- The “search by title” functionality should be suitable
- The “advanced search” functionality should be accurate
- The “online help support” functionality should be reliable
- The “basic search using keywords” should be accurate



Questionnaire

NAME OF SOFTWARE:

USER/QM/QA ID:

- 1) Rank the sub- characteristics according to your opinion after evaluating the software with the help of the Categorized Requirements and the Quality Model. Your rankings should be in range of 0-3.

Char.	Sub-Char.	Rank	Char.	Sub-char.	Rank
Functionality	Suitability		Efficiency	Time behavior	
	Accuracy			Resource Utilization	
	Interoperability		Maintainability	Analyzability	
	Security			Changeability	
Reliability	Maturity			Stability	
	Fault-tolerance			Testability	
	Recoverability			Compliance	
Usability	Understandability		Portability	Adaptability	
	Learnability			Installability	
	Operability			Co-existence	
	Attractiveness			Replaceability	

- 2) Comments:



The ISO Rating Levels

MARK	RATING	GLOBAL RATING
3	Excellent	Satisfactory
2	Good	
1	Fair	
0	Poor/Absent	



The Analysis Part

- We have used a Hypothetical data involving 1000 people who are assumed to participate for the evaluation of quality of a particular software. **This is not a Real Data**
 - 900 Users
 - 25 Quality Managers
 - 75 QA Team Members
- Rankings were assigned to the sub-characteristics on with the help of the requirements and based on the performance of the Software



Specimen of a Completed Questionnaire

NAME OF SOFTWARE: COMPANY B'S SOFTWARE
USER/QM/QA ID: 9710051

- 1) Rank the sub- characteristics according to your opinion after evaluating the software with the help of the Categorized Requirements and the Quality Model. Your rankings should be in range of 0-3.

Char.	Sub-Char.	Rank	Char.	Sub-char.	Rank
Functionality	Suitability	1	Efficiency	Time behavior	2
	Accuracy	2		Resource Utilization	3
	Interoperability	3	Maintainability	Analyzability	3
	Security	0		Changeability	0
Reliability	Maturity	1		Stability	1
	Fault-tolerance	2		Testability	2
	Recoverability	3		Compliance	2
Usability	Understandability	0	Portability	Adaptability	3
	Learnability	2		Installability	2
	Operability	2		Co-existence	3
	Attractiveness	2		Replaceability	3

- 2) Comments:

Consolidated Data after Questionnaires were collected

Char.	Sub-char.	User		QM		QA		Total		A
		No.	Sum	No.	Sum.	No.	Sum	No.	Sum	
FUNC.	Suitability	900	1700	25	35	75	35	1000	1770	1.77
	Accuracy	900	1900	25	65	75	65	1000	2030	2.03
	Interoperability	900	2200	25	30	75	50	1000	2280	2.28
	Security	900	1000	25	45	75	50	1000	1095	1.09
RELY.	Maturity	900	2200	25	30	75	45	1000	2275	2.27
	Fault-tolerance	900	800	25	15	75	45	1000	860	0.86
	Recoverability	900	1400	25	50	75	35	1000	1485	1.48
USAY.	Understandability	900	800	25	25	75	30	1000	855	0.85
	Learnability	900	100	25	50	75	20	1000	170	1.70
	Operability	900	800	25	50	75	20	1000	870	0.87
	Attractiveness	900	1100	25	65	75	65	1000	1230	1.23
EFFY.	Time Behavior	900	2200	25	65	75	65	1000	2330	2.33
	Resource Utilization	900	1400	25	20	75	50	1000	1470	1.47
MANY.	Analyzability	900	800	25	20	75	50	1000	870	0.87
	Changeability	900	800	25	30	75	35	1000	865	0.86
	Stability	900	1200	25	35	75	50	1000	1285	1.28
	Testability	900	2700	25	45	75	15	1000	2760	2.76
	Compliance	900	1100	25	45	75	30	1000	1175	1.17
PRTY.	Adaptability	900	800	25	50	75	45	1000	895	0.89
	Installability	900	1400	25	50	75	30	1000	1480	1.48
	Co-existence	900	1000	25	65	75	65	1000	1230	1.23
	Replaceability	900	1000	25	35	75	20	1000	1055	1.05

Prioritization of Characteristics

Char.	Sub-char.	Users	QM	QA	A	QVC	Pr.	FQVC
FUNC.	Suitability	1700	35	35	1.77	7.17	6	43.02
	Accuracy	1900	65	65	2.03			
	Interoperability	2200	30	50	2.28			
	Security	1000	45	50	1.09			
RELY.	Maturity	2200	30	45	2.27	4.61	2	9.22
	Fault-tolerance	800	15	45	0.86			
	Recoverability	1400	50	35	1.48			
USAY.	Understandability	800	25	30	0.85	4.65	3.5	16.27
	Learnability	100	50	20	1.70			
	Operability	800	50	20	0.87			
	Attractiveness	1100	65	65	1.23			
EFFY.	Time Behavior	2200	65	65	2.33	3.80	1	3.80
	Resource Utilization	1400	20	50	1.47			
MANY.	Analyzability	800	20	50	0.87	6.94	5	34.70
	Changeability	800	30	35	0.86			
	Stability	1200	35	50	1.28			
	Testability	2700	45	15	2.76			
	Compliance	1100	45	30	1.17			
PRTY.	Adaptability	800	50	45	0.89	4.65	3.5	16.27
	Installability	1400	50	30	1.48			
	Co-existence	1000	65	65	1.23			
	Replaceability	1000	35	20	1.05			



Determining the Total Quality Value of the Software

- The Total Quality Value of the Software is equal to the sum of the Final Quality Value of each Characteristic

$$TQVS = \sum \text{FQVC OF EACH CHARACTERISTIC}$$

$$TQVS = 43.02 + 9.22 + 16.27 + 3.80 + 34.7$$

$$TQVS = 107.01$$



Determining the Quality Factor

- Quality Factor (QF) = $TQVS / ITQVS$
ITQVS is the Ideal Total Quality Value of the Software
- ITQVS can be calculated when all the sub-characteristics attain a rank of “3” by the Users, QMs and the QAs



Determining the Quality Factor (Contd.)

- For the present example

$$\text{TQVS} = 107.01$$

$$\text{ITQVS} = 258$$

- $\text{QF} = \text{TQVS}/\text{ITQVS}$

$$\text{QF} = 107.01/258$$

$$\text{QF} = 0.41$$



Determining the Quality from QF

- If $0.0 \leq QF \leq 0.25$
Software Quality is “Poor”
- If $0.26 \leq QF \leq 0.50$
Software Quality is “Fair”
- If $0.51 \leq QF \leq 0.75$
Software Quality is “Good”
- If $0.76 \leq QF \leq 1.00$
Software Quality is “Excellent”



Conclusion

- Users plays a role in the measurement of Software Quality
- Users have a direct and equal impact on the software quality
- Identifies the requirements which needs improvement
- Can be used to improve the quality of later versions of the software
- Requires a good amount of time for the evaluation



Future Work

- To measure the effectiveness of our metric by using realistic data
- To compare the efficiency of this metric with the existing Software Quality Metrics