BLG 475E: Software Quality and Testing
Fall 2017-18
Software quality models and standards

Dr. Ayse Tosun

Course Outline

- Quality factors
- Quality models
  - McCall's model
  - Boehm's model
  - ISO 9126
  - CMM and CMMI

Example

- "Our new sales information system seems *okay*, the invoices are *correct*, the inventory records are *correct*, the discounts granted to our clients *exactly follow our very complicated discount policy*. **But** our new sales information system frequently fails, usually at least twice a day, each time for 20 minutes or more. Imagine how embarrassing it is to store managers… Softbest, the software house that developed our computerized sales system claims no responsibility…"[1]

[1] From Galvin, 2004

- Does the project fulfill the requirements?
- What are the attributes that are missing in requirements?

Example 2

- "The new version of our loan contract software is really accurate. We have already processed 1200 customer requests, and checked each of the output contracts. There were no errors. **But** we did face a severe unexpected problem – training a new staff member to use this software takes about two weeks. This is a real problem in customers' departments suffering from high employee turnover…The project team says that as they were not required to deal with training issues in time, an additional two or three months of work will be required to solve the problem"

[1] From Galvin, 2004

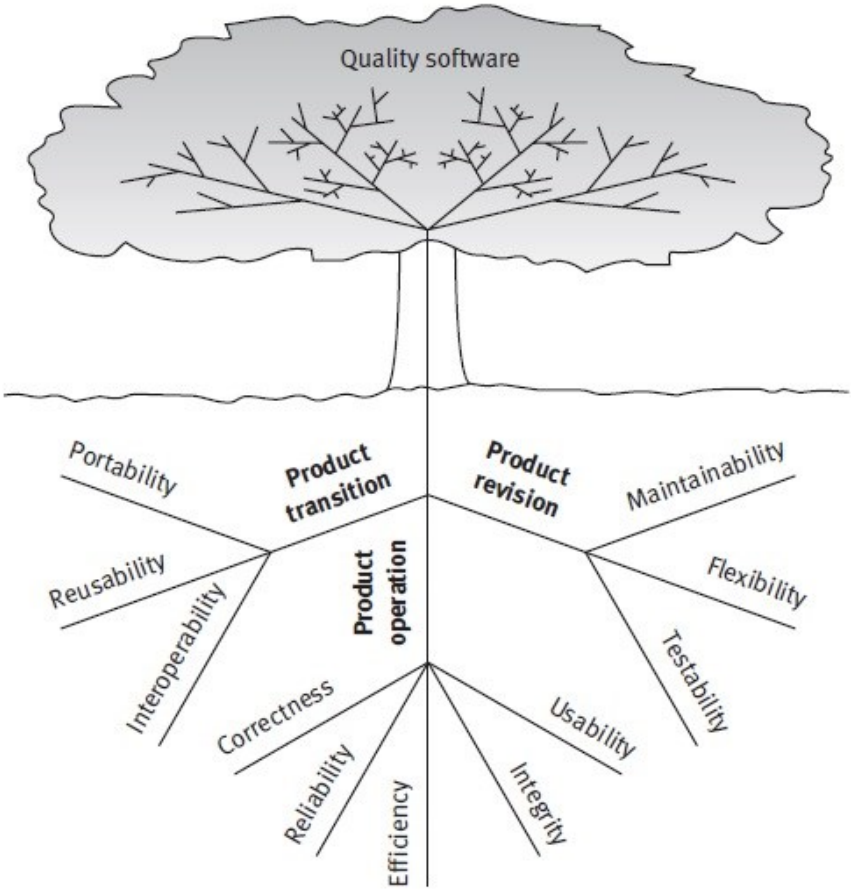
- Does the project fulfill the requirements?
- What are the attributes that are missing in requirements?

What is missing in requirements?

- Remember definition of quality (IEEE, 1991)
  - The degree to which a system, component, or process meets specific requirements
  - The degree to which a system, component, or process meets customer or user needs or expectations
- Requirements: functional / non-functional
- All attributes of software and aspects of the use of software should be covered:
  - Usability, maintainability, full satisfaction of the users

Quality factors

- First quality model proposed by McCall
  - Suggested 11 *quality factors and sub-factors* to model software requirements



Based on McCall et al., 1977

McCall's Product Operation Quality Factors

- Correctness
  - consistency, completeness, traceability
- Reliability
  - consistency, accuracy, fault tolerance
- Efficiency
  - execution time and storage efficiency
- Integrity
  - software system security, access control
- Usability
  - operability, training, communicativeness

Required outputs of software system.
+ accuracy of outputs
+ completeness of the output data
+ up-to-dateness of the information
+ availability of the information
+ standards for coding and development

McCall's Product Revision Quality Factors

- Maintainability
  - simplicity, conciseness, instrumentation, self-descriptiveness
- Flexibility
  - expandability, generality, modularity
- Testability
  - simplicity, instrumentation, self-descriptiveness, modularity

McCall's Product Transition Quality Factors

- Portability
  - Simplicity, software system independence, machine independence
- Reusability
  - Simplicity, generality, software system independence, machine independence
- Interoperability
  - Modularity, communications communality, data communality

McCall's Quality Factor Model

- These quality factors should be measured.
- They should be meaningful for non-technical stakeholders.
- Alternative models based on McCall's
  - 1987, Evans and Marciniak model
  - 1988, Deutsch and Willis model
  - ++ Verifiability (both), Expandibility (both), Safety (DW), Manageability (DW), Survivability (DW)
  - -- Testability (both)
- Evans and Marciniak Model
  - Sub-factors for McCall's quality factors (See Table 3.3 in Galvin, 2004 for mappings)

Boehm's Model

Prime characteristics of quality is what they define as "general utility"

→ How well (easily, reliably, efficiently) can I use it [software system] as-is?

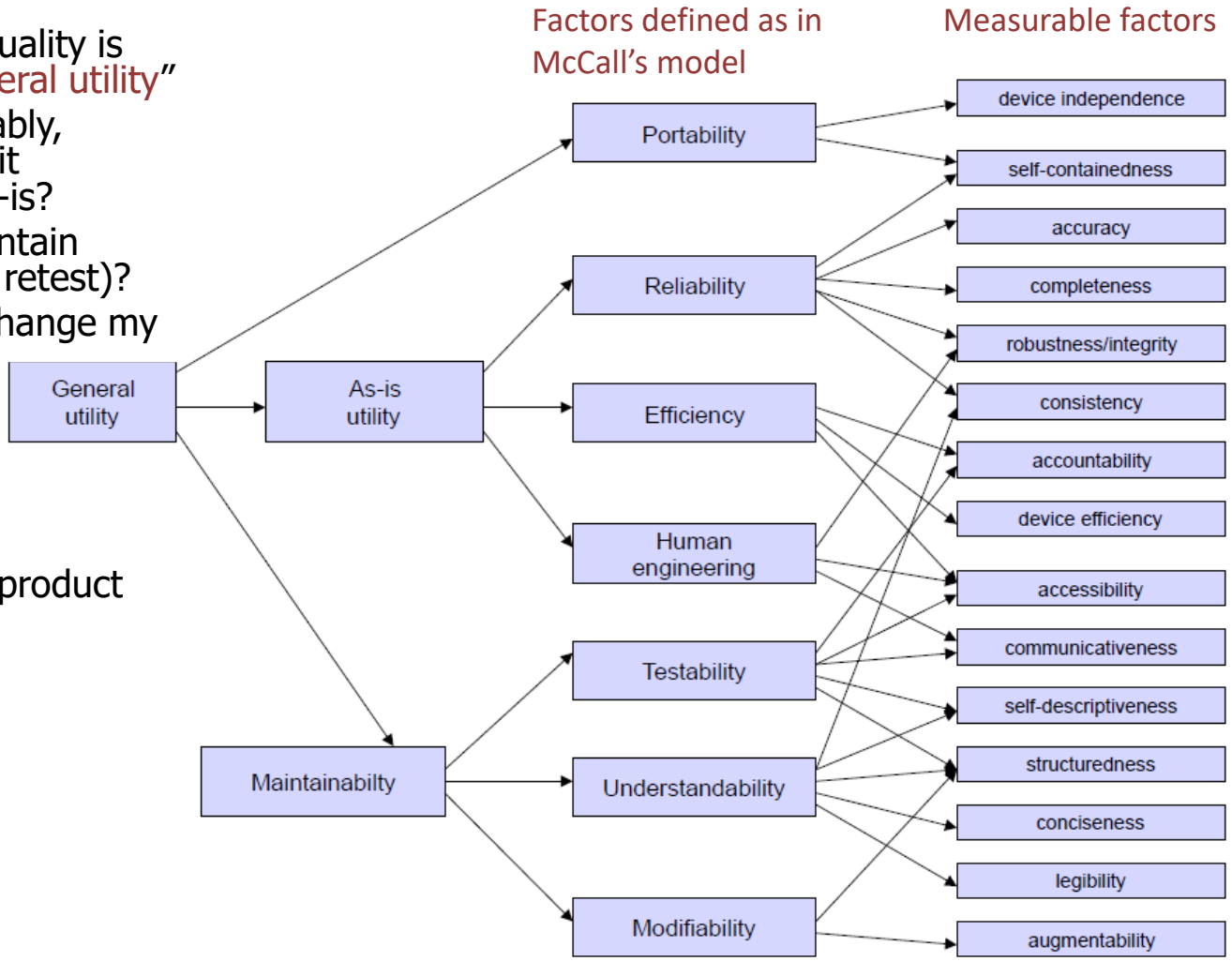
→ How easy is it to maintain (understand, modify, retest)?

→ Can I still use it if I change my environment?

Top to bottom approach

Top: Concerns of users

Bottom: Highly technical product features



Process Maturity Models

- Evaluation of the maturity of an organization
  - Benchmark for SQA practice
- Breakdown of software process into *process areas*
- Provides instruments of measuring maturity of complete process and/or separate process areas
- Examples
  - ISO standards
  - SEI - Capability Maturity Models (CMM)

ISO 9126

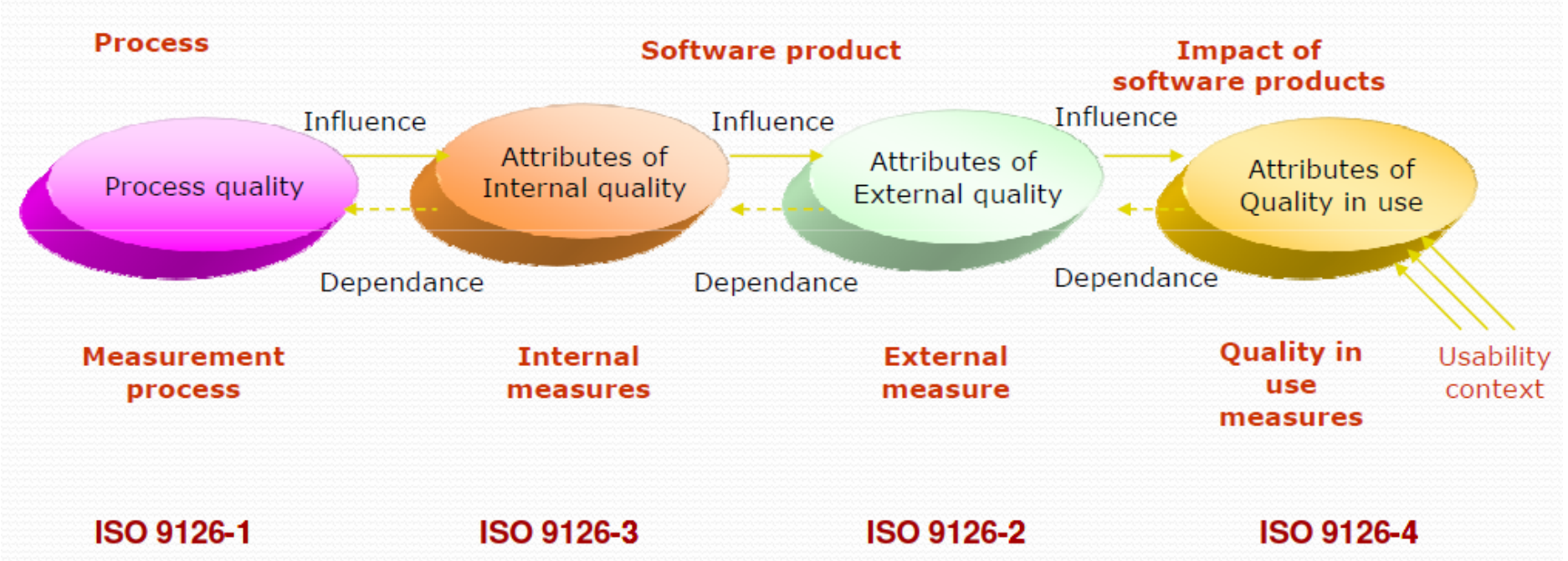
- International Organization for Standards (ISO), 1991
  - Software quality model and a set of guidelines for measuring the characteristics associated with software (ISO 9126)
- 2001, 2003
  - ISO 9126-1 updated quality model
  - ISO 9126-2 external measures
  - ISO 9126-3 internal measures
  - ISO 9126-4 quality in use measures

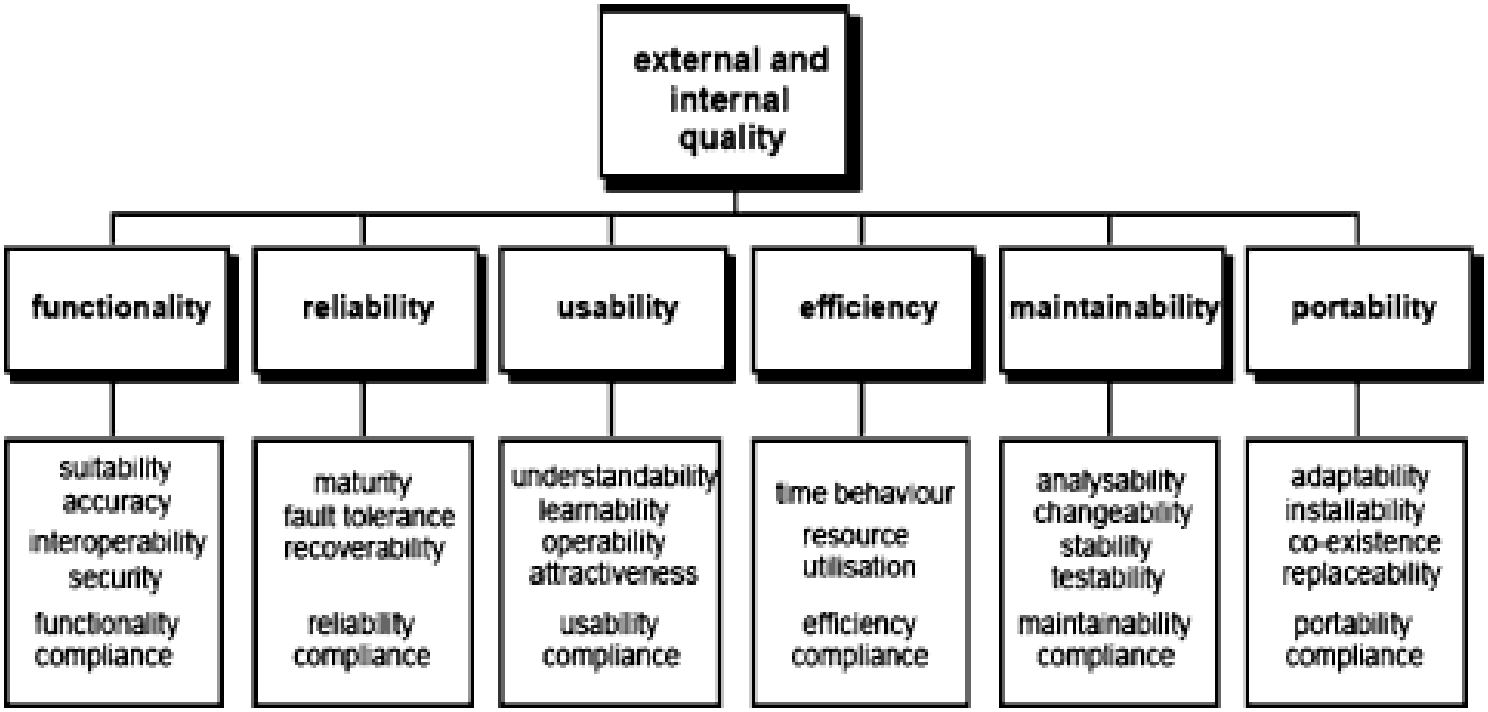
External quality: Measured and evaluated while *testing*

Internal quality: Measured and evaluated while *requirements, design, code implementation, reviewing.*

Quality in use is the **user's view** of the quality of the software product when it is used in a specific environment and a specific context of use. It measures the extent to which users can achieve their goals in a particular environment, rather than measuring the properties of the software itself (ISO/IEC, 2001a).

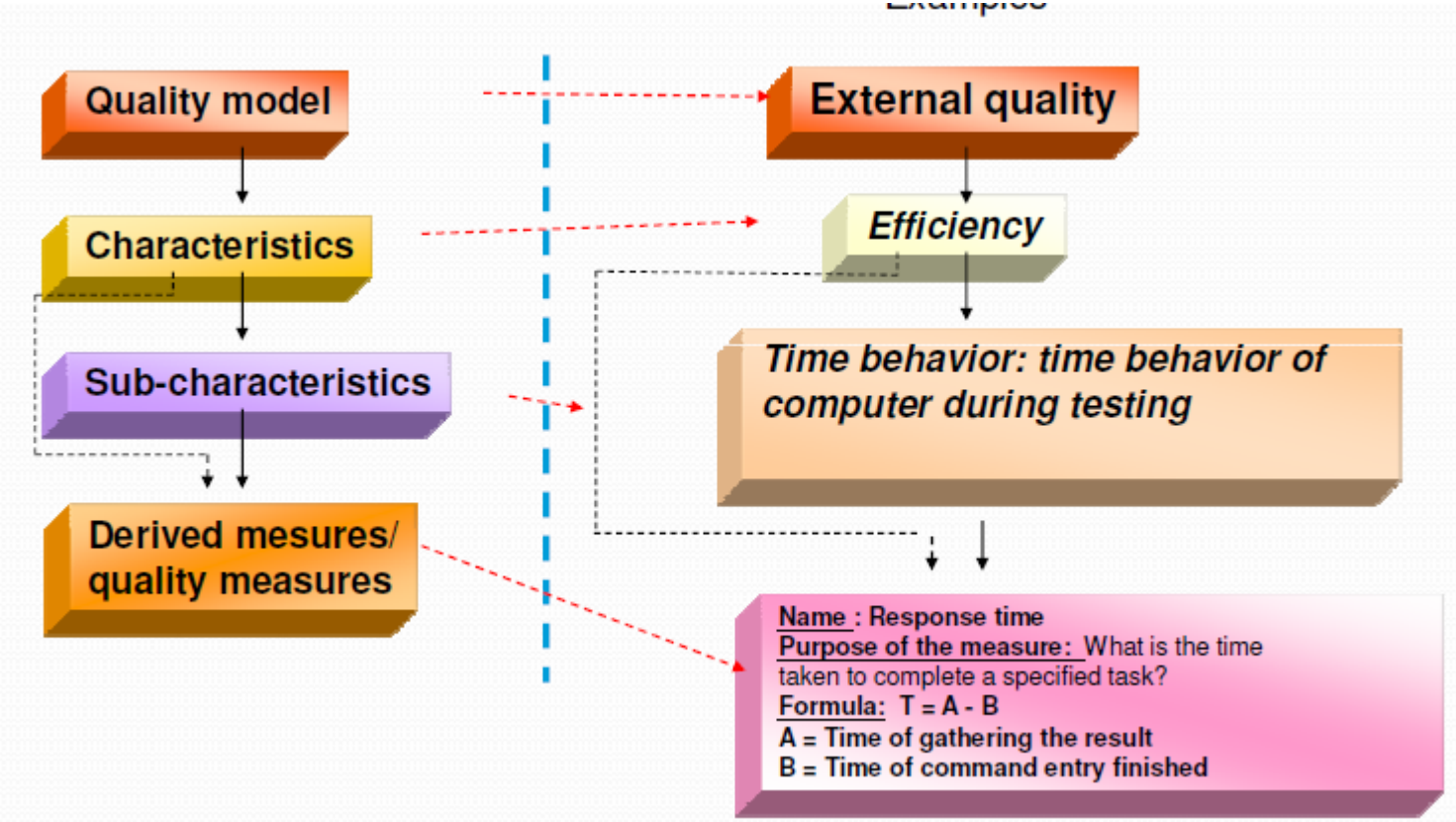
ISO 9126's view on quality




external and internal quality
functionality
reliability
usability
efficiency
maintainability
portability
suitability
accuracy
interoperability
security
functionality compliance
maturity
fault tolerance
recoverability
reliability compliance
understandability
learnability
operability
attractiveness
usability compliance
time behaviour
resource utilisation
efficiency compliance
analysability
changeability
stability
testability
maintainability compliance
adaptability
installability
co-existence
replaceability
portability compliance

Functionality: A set of attributes that relate to the existence of a set of functions and their specified properties.
The functions are those that satisfy stated or implied needs.
- Suitability: Attribute of software that relates to the presence and appropriateness of a set of functions for
specified tasks.
- Accuracy: Attributes of software that bare on the provision of right or agreed results or effects.
- Security: Attributes of software that relate to its ability to prevent unauthorized access, whether accidental or
deliberate, to programs and data.
- Interoperability: Attributes of software that relate to 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.

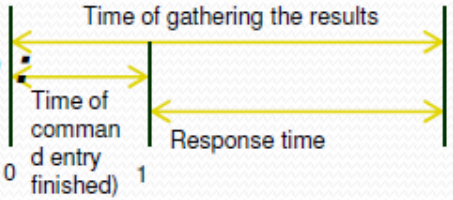
Example


Quality model
Characteristics
Sub-characteristics
Derived mesures/ quality measures
External quality
Efficiency
Time behavior: time behavior of computer during testing
Name : Response time
Purpose of the measure: What is the time taken to complete a specified task?
Formula: T = A - B
A = Time of gathering the result
B = Time of command entry finished

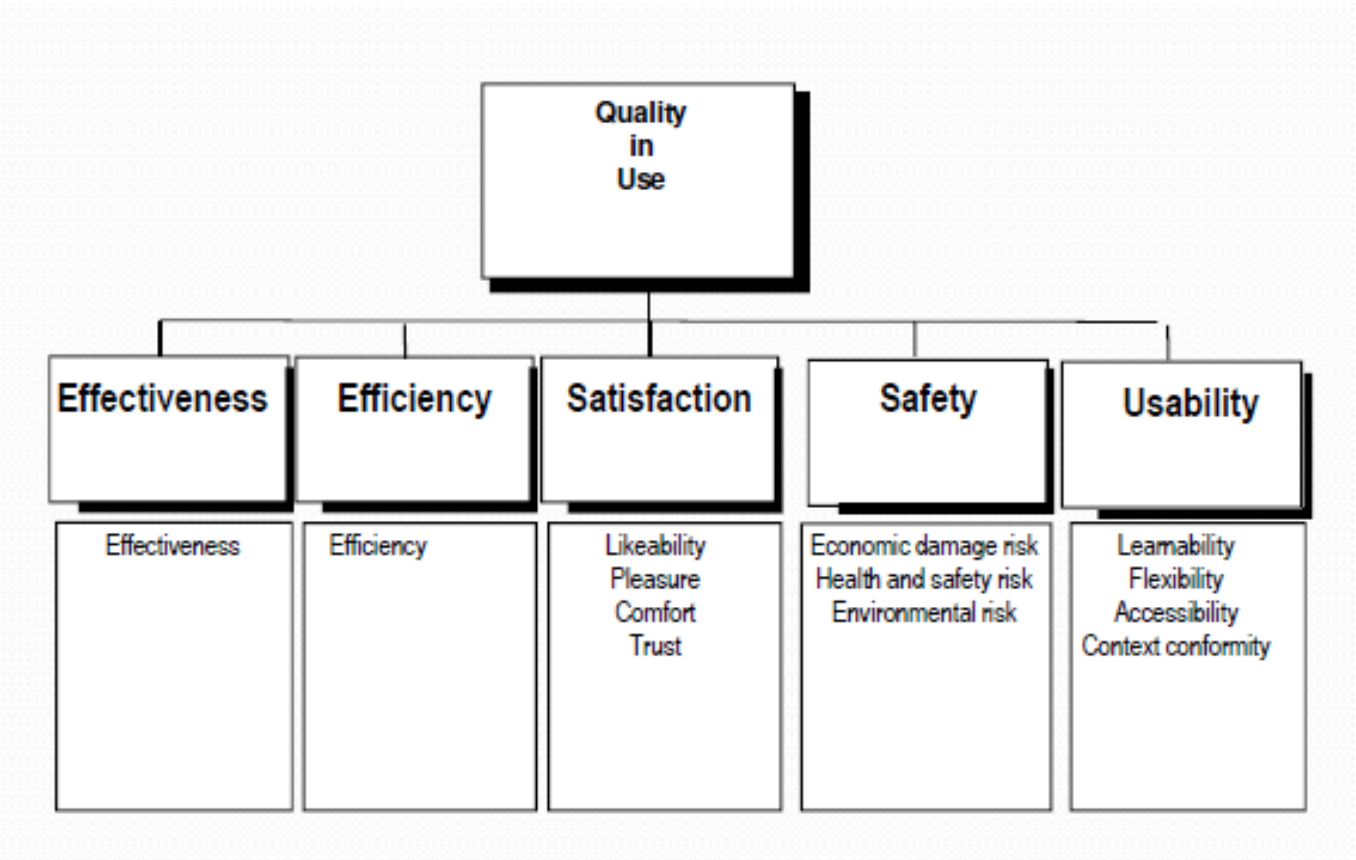
Characteristics: Efficiency
Sub-characteristic: Time behaviour
Example of measure: Response time

External time behaviour metrics a) Response time									
Metric name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measure-ment	ISO/IEC 12207 SLCP Reference	Target audience
Response time	What is the time taken to complete a specified task? How long does it take before the system response to a specified operation?	Start a specified task. Measure the time it takes for the sample to complete its operation. Keep a record of each attempt.	T = (time of gaining the result) - (time of command entry finished)	0 < T The sooner is the better.	Ratio	T= Time	Testing report Operation report showing elapse time	5.3 Sys./Sw. Integration 5.3 Qualification testing 5.4 Operation 5.5 Mainte-nance	User Developer Maintainer SQA

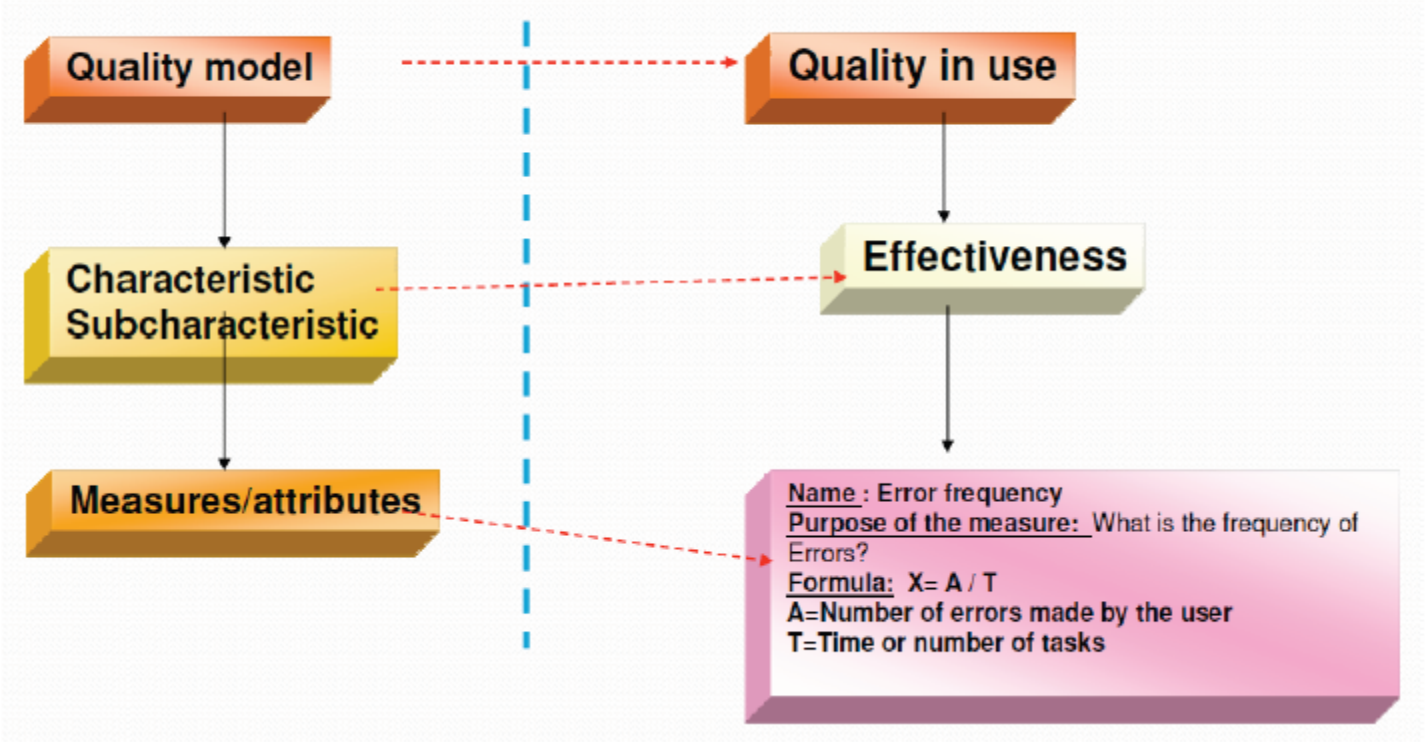
Example :



Quality in Use



Example


Quality model
Characteristic
Subcharacteristic
Measures/attributes
Quality in use
Effectiveness
Name : Error frequency
Purpose of the measure: What is the frequency of Errors?
Formula: X= A / T
A=Number of errors made by the user
T=Time or number of tasks

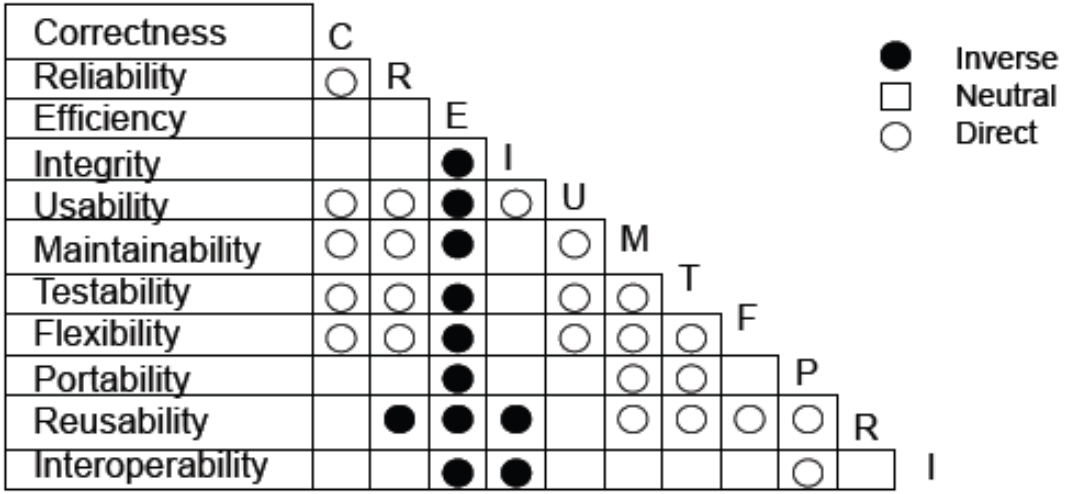
How to apply quality models

Selecting and prioritizing quality factors

Type of the application	quality characteristics (ISO 9126)
√ Human life in danger	Reliability
√ Long-life system	Maintainability
√ Sensitive to change	Maintainability
√ Immature technology	Portability
√ Many changes during life time	Maintainability
√ Real time application	Efficiency, reliability
√ Embedded system	Efficiency, reliability
√ Secure system	Functionality-> security
√ Systems interconnected	Functionality-> interoperability

From Ilkka Tervonen, Lecture notes , 2011

Relationships between quality factors



Examples of the relationships between quality factors

Integrity vs. efficiency (inverse)
The control access to data or software requires additional code and processing leading to a longer runtime and additional storage requirements.
Usability vs. efficiency (inverse)
Improvements in the human/computer interface may significantly increase the amount of code required.
Portability vs. efficiency (inverse)
The use of optimized software or system utilities will lead to a decrease in portability.

Maintainability vs. flexibility (direct)
Maintainable code arises from code that is well structures. This will also assist any modifications or alterations that are required. Thus a direct relationship exists between these properties.

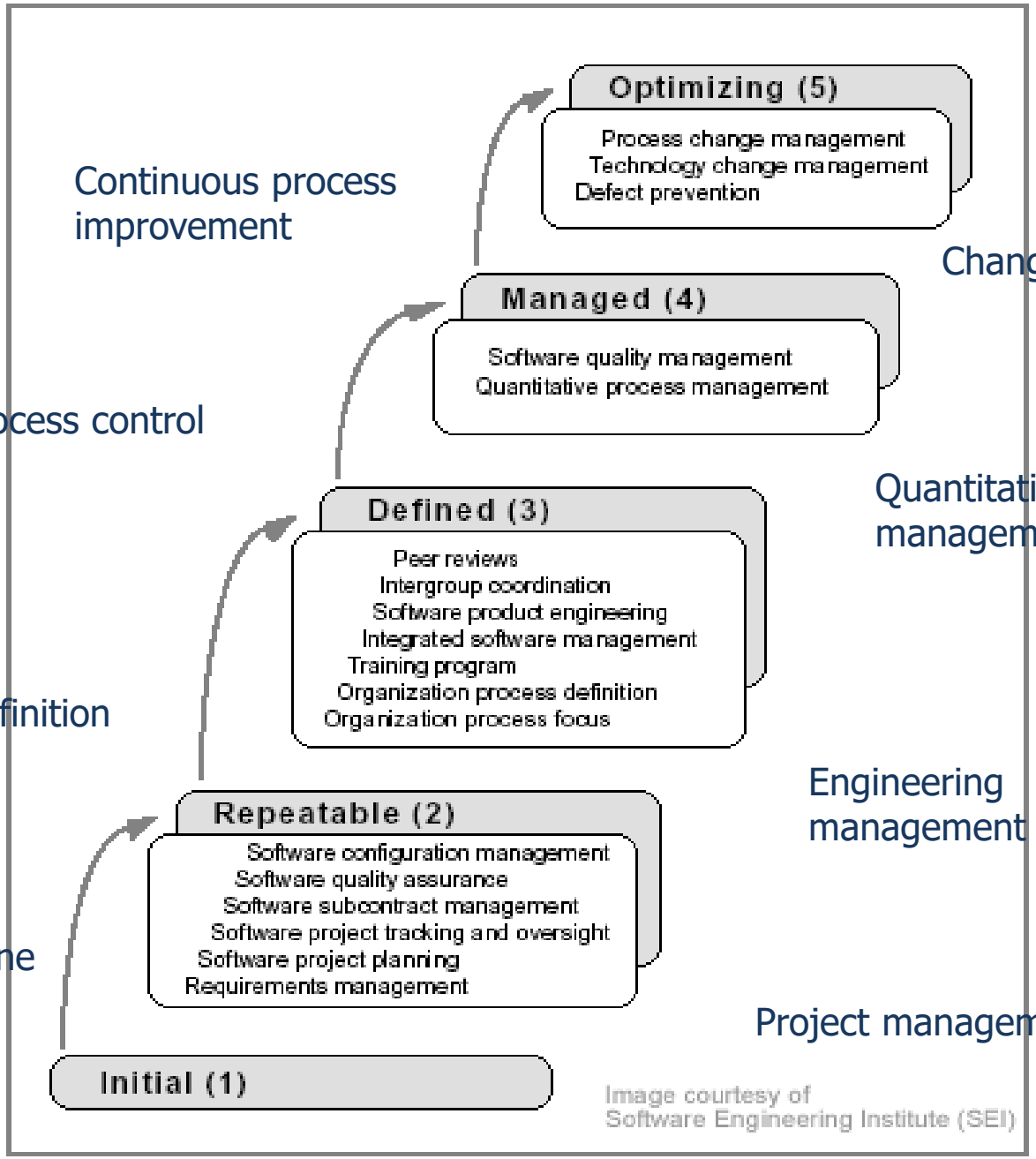
Gillies A., Software Quality, Theory and Management,1997
Perry W., Effective methods of EDP Quality Assurance, 1987

Capability Maturity Model

- Applied research and development center
- Development of a capability maturity model – 1987
  - Initial version – 1992 mainly for receipt of feedback from the software community.
  - First release for public use – 1993
- Objective
  - Continuously improve software intensive systems
  - Help organizations to improve their software engineering capabilities and to develop or acquire the right software, defect free, within budget and on time, every time.

Principles of CMMI

- More elaborate management methods based on quantitative approaches
- Evaluation of achievement and determining efforts necessary to reach next capability level by locating process areas requiring improvement
- Process areas (generic) define "**what**", not "how"
  - Use any life cycle model
  - Use any design methodology, software development tool and programming language
  - Use any documentation standard

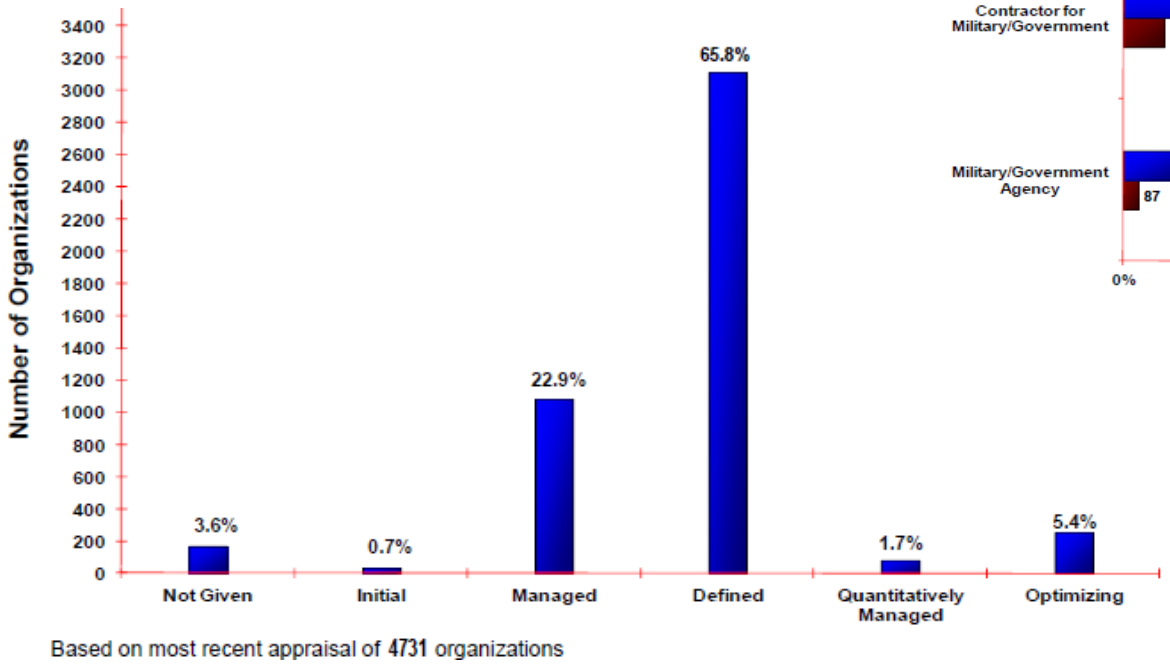

Optimizing (5)
Process change management
Technology change management
Defect prevention
Continuous process improvement
Change management
Managed (4)
Software quality management
Quantitative process management
Process control
Quantitative management
Defined (3)
Peer reviews
Intergroup coordination
Software product engineering
Integrated software management
Training program
Organization process definition
Organization process focus
Process definition
Engineering management
Repeatable (2)
Software configuration management
Software quality assurance
Software subcontract management
Software project tracking and oversight
Software project planning
Requirements management
Process discipline
Project management
Initial (1)
Image courtesy of
Software Engineering Institute (SEI)

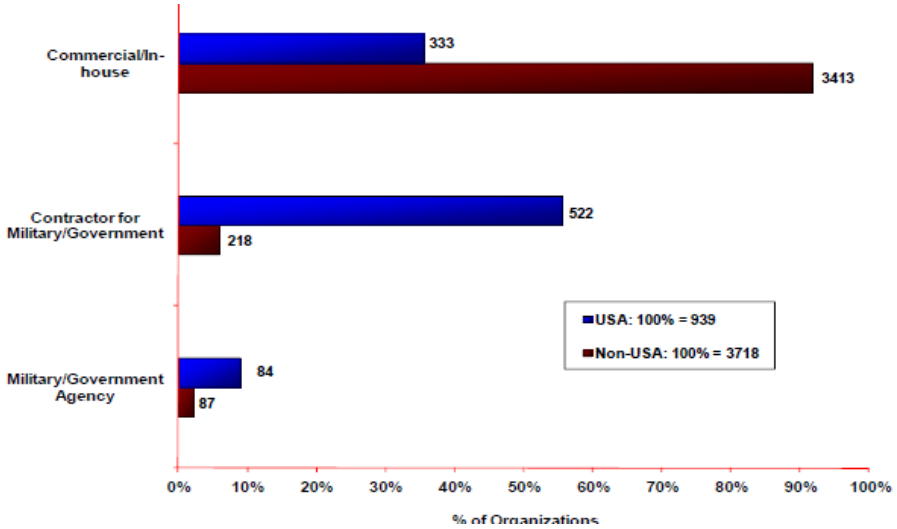
CMM/ CMMI

1. Initial – chaotic
   - unpredictable (cost, schedule, quality)
2. Repeatable - intuitive
   - cost/quality highly variable,
   - some control of schedule, informal/ad hoc procedures.
3. Defined - qualitative
   - reliable costs and schedules,
   - improving but unpredictable quality performance.
4. Managed - quantitative
   - reasonable statistical control over product quality.
5. Optimizing – quantitative basis for continuous improvement.

CMMI Customer Profiles

- 5000 businesses using CMMI over 70 countries, including the U.S., China, Germany, Italy, Chile, India, Australia, Egypt, Finland, Turkey and Russia





From CMMI-SCAMPI v1.2/1.3 Class A Appraisal Results, March 2012

References

- Galin D., *Software Quality Assurance: From theory to implementation*, Addison Wesley, 2004.
- Cote M-A., Suryn W., Georgiadou E., In search for a widely applicable and accepted software quality model for software quality engineering, Software Quality Journal, vol 15, no 4, 2007, pp. 401-416
- Deshernais, J.M., Software Measurement: Analysis of ISO/IEC 9126 and 25010, available at http://www.cmpe.boun.edu.tr/courses/cmpe58V/fall2009/06a-Analysis%20of%209126-2,3,4short.pdf
- Milicic D., Software Quality Models and Philosophies, Chapter 1 in book Lundberg L., Mattsson M., Wohlin C., Software Quality Attributes and Trade-Offs, 2005.