SWINBURNE UNIVERSITY OF TECHNOLOGY

SWE30010
Development Project 2: Design, Planning and Management

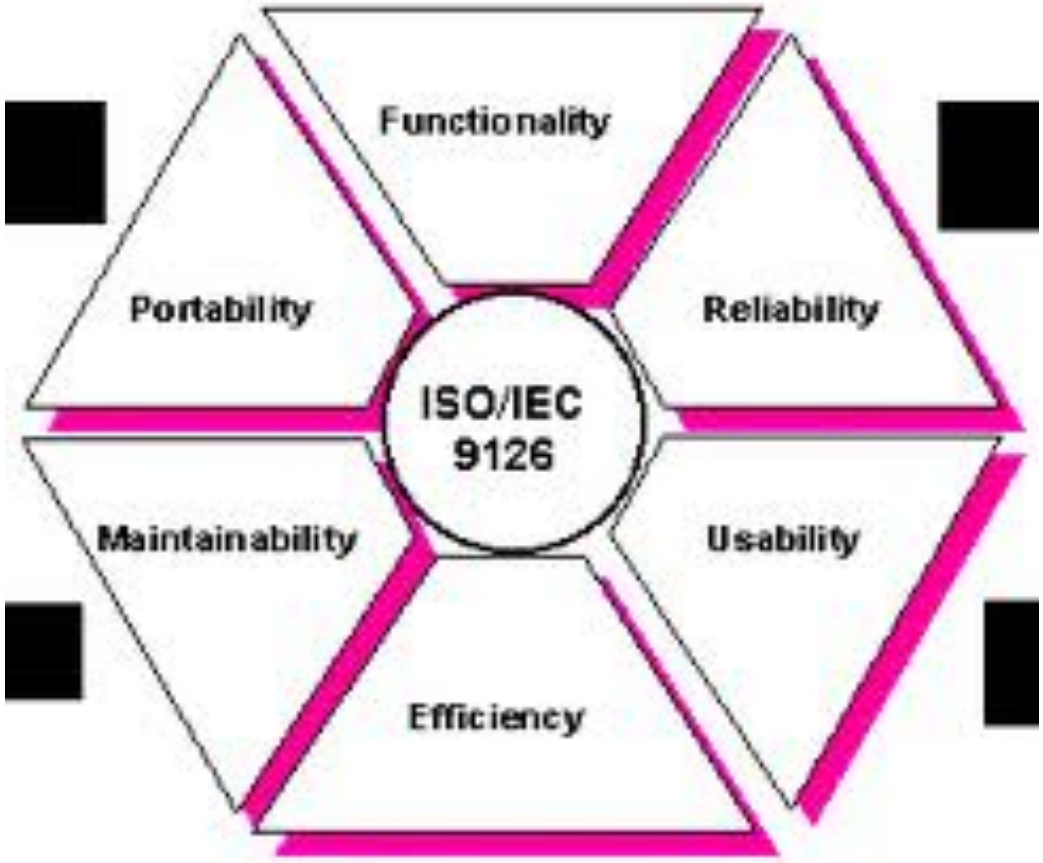
Lecture 3b

ISO Software Product Quality Model

Principal References

- Ian Sommerville, *Software Engineering* (8th Edition), Addison-Wesley, 2004, Chapter 27.
- Roger S. Pressman, *Software Engineering - A Practitioners Approach* (6th Edition), McGraw-Hill, 2005, Chapter 26.
- Bob Hughes and Mike Cotterell, *Software Project Management* (4th Edition), Wiley, 2006, Chapter 12.
- Carlo Ghezzi, Mehdi Jazayeri, Dino Mandroli, *Fundamentals of Software Engineering* (2nd Edition), Prentice-Hall 2003, Chapter 2.

ISO/IEC 9126-1 SE – Product Quality Model



For details see http://en.wikipedia.org/wiki/ISO/IEC_9126

ISO/IEC 9126 Model – Six Characteristics

- Functionality – 5 sub-characteristics
- Reliability – 4
- Usability – 5
- Efficiency – 3
- Maintainability – 5
- Portability – 5

- Total: 27 sub-characteristics

ISO/IEC 25010 Model – Eight Characteristics

- Functionality Suitability – 3 sub-characteristics [5 – 3 + 1]
- Performance Efficiency – 3 [3 – 1 + 1]
- Compatibility (new) – 2 [+ 2]
- Usability – 6 [5 – 1 + 2]
- Reliability – 4 [4 – 1 + 1]
- Security (new) – 5 [+ 5]
- Maintainability – 5 [5 – 3 + 3]
- Portability – 3 [5 – 2]
- Total: 31 sub-characteristics

Details in http://iso25000.com/index.php/en/iso-25000-standards/iso-25010

ISO 25010 vs ISO 9126

Functional Suitability

- Functional Appropriateness
- Functional Correctness
- Functional Completeness (new)

Functionality

- Suitability
- Accuracy
- Interoperability (to Compatibility)
- Security (to Security as a char.)
- Functional Compilance

ISO 25010 vs ISO 9126

Performance Efficiency

- Time Behaviour
- Resource Utilization
- Capacity (new)

Efficiency

- Time Behaviour
- Resource Utilization
- Efficiency Compilance

ISO 25010 vs ISO 9126

Compatibility (new)

- Co-existence (from Portability)
- Interoperability (from Functionality)

ISO 25010 vs ISO 9126

Usability

- Appropriateness recognizability
- Learnability
- Operability
- User interface aesthetics
- User error protection (new)
- Accessibility (new)
  - Use by people with a wide range of characteristics

Usability

- Understandability
- Learnability
- Operability
- Attractiveness

- Usability Compilance

ISO 25010 vs ISO 9126

Reliability

- Maturity
- Fault Tolerance
- Recoverability
- Availability (new)
  - when required for use

Reliability

- Maturity
- Fault Tolerance
- Recoverability

- Reliability Compilance

ISO 25010 vs ISO 9126

Security (new)

- Confidentiality (new)
  - Data accessible only by those authorized
- Integrity (new)
  - Protection from unauthorized modification
- Non-repudiation (new)
  - Actions can be proven to have taken place
- Accountability (new)
  - Actions can be traced to who did them
- Authenticity (new)
  - Identify can be proved to be the one claimed

ISO 25010 vs ISO 9126

Maintainability

- Analyzability
- Modifiability
  - Changeability and Stability combined
- Testability
- Modularity (new)
  - Changed in one component has a minimal impact on others
- Reusability (new)

Maintainability

- Analyzability
- Changeability
- Stability
- Testability
- Maintainability Compilance

ISO 25010 vs ISO 9126

Portability

- Adaptability
- Installability
- Replaceability

Portability

- Adaptability
- Installability
- Co-Existence (to Compatibility)
- Replaceability
- Portability Compilance

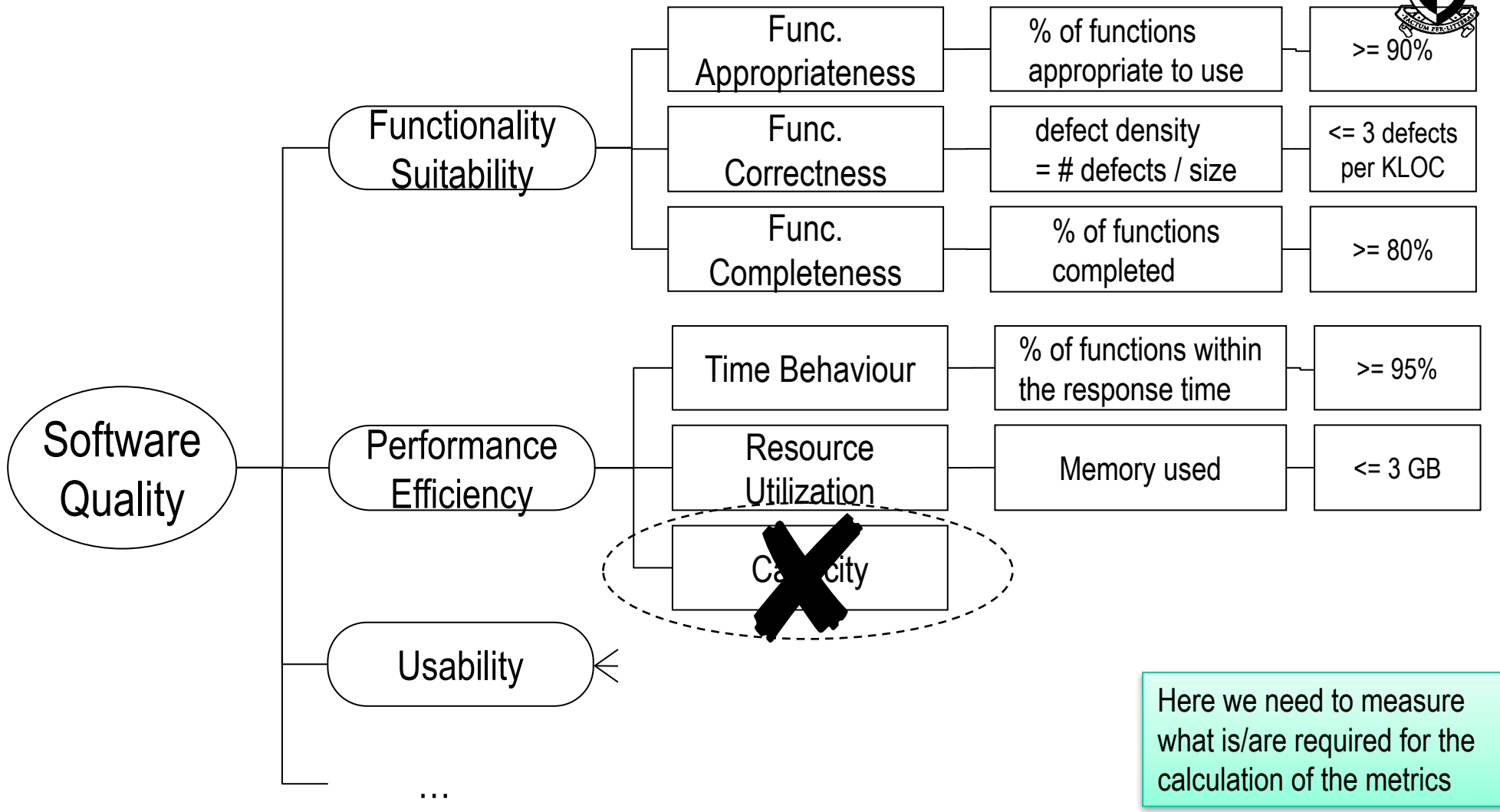
What do the Quality Models Mean?

- Various QMs
  - McCall's Model (not discussed in DP2), ISO 9126, ISO 25010, …
- It's all very well referring to all of the "ilities" in quality models, but what do they mean?
- How do we actually specify quality under these headings?
- What measurements should be taken?
- What are acceptable values for the measures?

What do the Quality Models Mean? (cont'd)

- How do we design and construct software so that the quality requirements will arise naturally, rather than being "forced" through fixing defects?
- How do we educate customers to engage with designers and developers in setting appropriate quality standards?
- These are the REAL questions that software engineers have to handle!
- And they are VERY HARD!!

Using QM – Traditional PM – An Example



Using QM – Scrum approach

- Specify in “Definition of Done”
- Specify what is required in the Product Backlog Item
  - During the initial product backlog development
  - When revising the product backlog item (e.g. after a sprint)
- Specify what is required in the Sprint Backlog Item
  - During sprint planning meeting
  - When revising the “sprint” backlog item (e.g. product owner cancelling a sprint)
- Be S.M.A.R.T. when specifying what is required

Using QM – Scrum – some generic examples

Definition of Done	How to check?
■ **All** code has been refactored	■ Code review
■ Performance testing	■ Test the performance of each item ☐ Prepare your performance test cases ☐ Do the performance testing ☐ Measure and verify the response time
■ Integration testing	■ Test the item after it being integrated to the main system ☐ Prepare your integration test cases ☐ Integrate the item into the main system ☐ Do the integration testing ☐ Verify the results after integration

Scrum – Peer Review System (Example)

S.M.A.R.T. enough?

Product / Sprint Backlog Items

- Allow a student to submit his/her peer review assessment of another student via the online peer review form
- A proper message will be display within 1s after the student click the submit button
- A student without any prior experience on peer review submission should be able to “submit” his / her assessment within 10 minutes after the form has been displayed
- The system should be able to allow 10 students submitting their peer review assessments at the same time

Scrum – Peer Review System (Example)

S.M.A.R.T. enough?

Product / Sprint Backlog Items

- Allow a student to modify his/her previous peer review assessment of another student via the online peer review form
- Once a student id is submitted, the system will display the original peer review assessment of this student previously entered by the user within 2s
- …

Conclusion

- Hopefully we have identified issues that need to be explored
- We have identified concerns that should occupy our thoughts when we design and build software
- There is quite a lot of research “out there”
- But there are few certainties, and not enough empirical evidence
- That said, if we focus on making software functional, reliable, robust, efficient, maintainable, usable (however we define these) – then we contribute to the production of high quality software
  - Even if we do not quite achieve this in a computed, quantitative, controlled way!