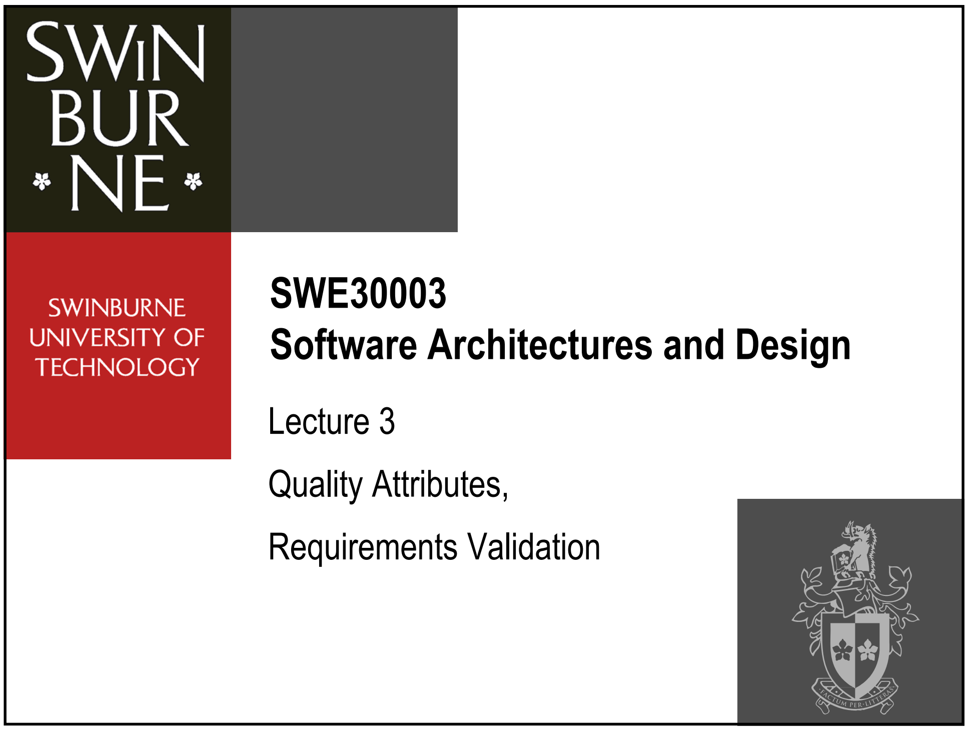


SWINBURNE UNIVERSITY OF TECHNOLOGY

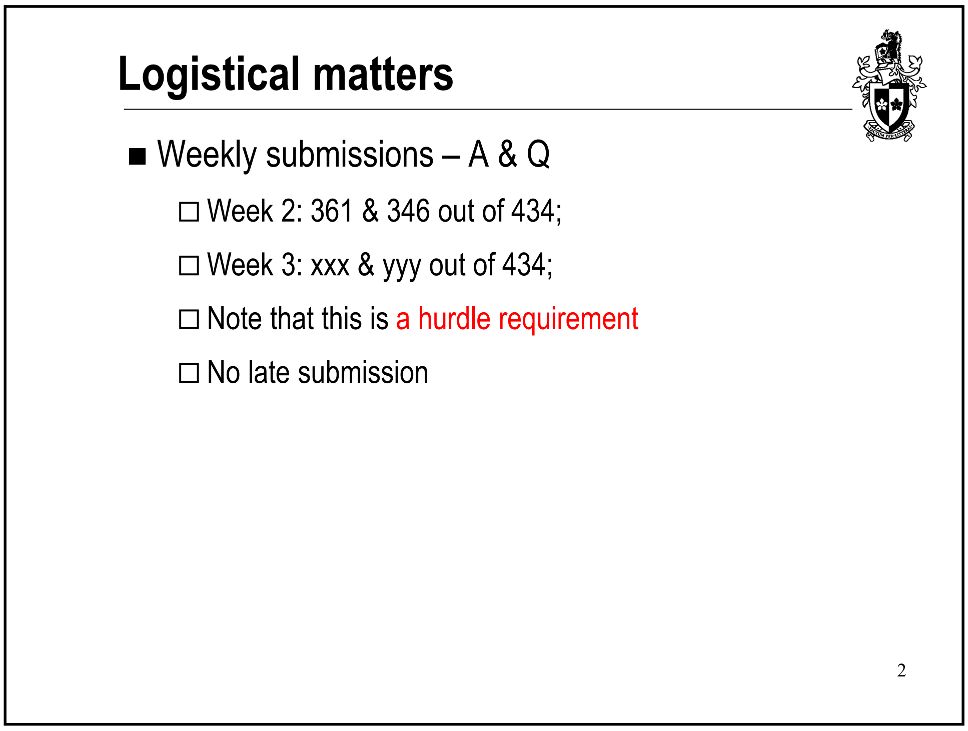
# SWE30003 Software Architectures and Design

Lecture 3

Quality Attributes,

Requirements Validation

1



## Logistical matters

- Weekly submissions – A & Q
  - ☐ Week 2: 361 & 346 out of 434;
  - ☐ Week 3: xxx & yyy out of 434;
  - ☐ Note that this is a hurdle requirement
  - ☐ No late submission

2

2

## Assignment 1 SRS

- A business software/information system
- Things the client wants to achieve … “not clear enough” – deliberate
- Requirements elicitation and specification: role play, analysis, reasonable assumptions
- Domain/data model: high level ER + explanation
- Functional requirements: Tasks&Support + explanation
- Non-functional (quality) requirements: next week
- Requirements validation: next week
- True collaboration …
- Use discussion board …

Questions? *** start early ***

3

3

## Question to Answer - Week 2

Describe the difference between Use Case diagrams and Task Descriptions. What are the advantages of using Task Descriptions as opposed to User Cases? Provide a situation where Task Descriptions are more suitable for use than Use Case Diagrams.

------

4

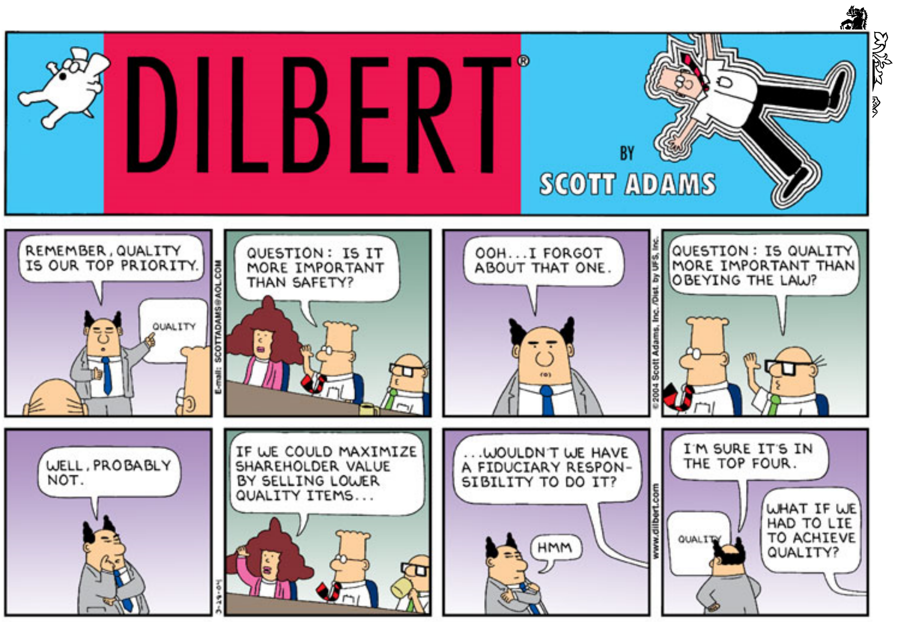
4

## Principal References

- Len Bass, Paul Clements, and Rick Kazman, *Software Architecture in Practice (4th Edition)*, Addison-Wesley, 2021, Chapters 3 and 4. (chapters in previous editions of the same topics are reasonable replacements).
- Ian Gorton, *Essential Software Architecture*, Springer, 2006, Chapter 3 (available from Canvas).
- Soren Lauesen, *Software Requirements - Styles and Techniques*, Addison-Wesley, 2002, Chapters 6 and 9.
- Allan R. Tucker (Ed.), *Computer Science Handbook (2nd Edition)*, Chapter 101 - *Software Qualities and Principles* (available from Canvas).

## Outline

- **Software Quality**
  - ☐ Quality Models
- Quality Attributes
  - ☐ Quality Attribute Scenarios
- Requirements V&V


DILBERT
BY SCOTT ADAMS
REMEMBER, QUALITY IS OUR TOP PRIORITY.
QUALITY
QUESTION: IS IT MORE IMPORTANT THAN SAFETY?
OOH...I FORGOT ABOUT THAT ONE.
QUESTION: IS QUALITY MORE IMPORTANT THAN OBEYING THE LAW?
WELL, PROBABLY NOT.
IF WE COULD MAXIMIZE SHAREHOLDER VALUE BY SELLING LOWER QUALITY ITEMS...
...WOULDN'T WE HAVE A FIDUCIARY RESPONSIBILITY TO DO IT?
HMM
I'M SURE IT'S IN THE TOP FOUR.
QUALITY
WHAT IF WE HAD TO LIE TO ACHIEVE QUALITY?
E-mail: SCOTTADAMS@AOL.COM
© 2004 Scott Adams, Inc./Dist. by UFS, Inc.
www.dilbert.com

http://en.wikipedia.org/wiki/Fair_use

7

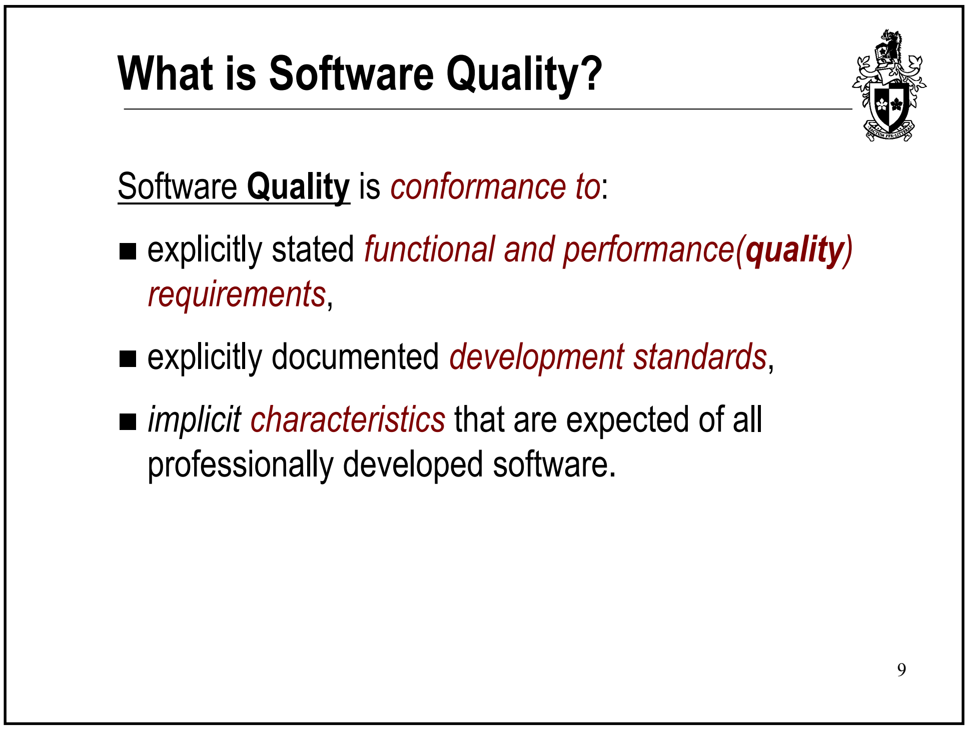
7

## What is Quality?

- ISO defines **quality** as "*the totality of characteristics of an entity that bear on its ability to satisfy stated or implied needs*."
- Some Related terms:
  - **"Conformance to requirements"** means that both the process and product of the project must meet the written specifications.
  - **"Fitness for use"** means that a product can be used as it was intended.
  - **"Meeting customer needs"** means that the product meets the (explicit and implicit) expectations of stakeholders.

8

8



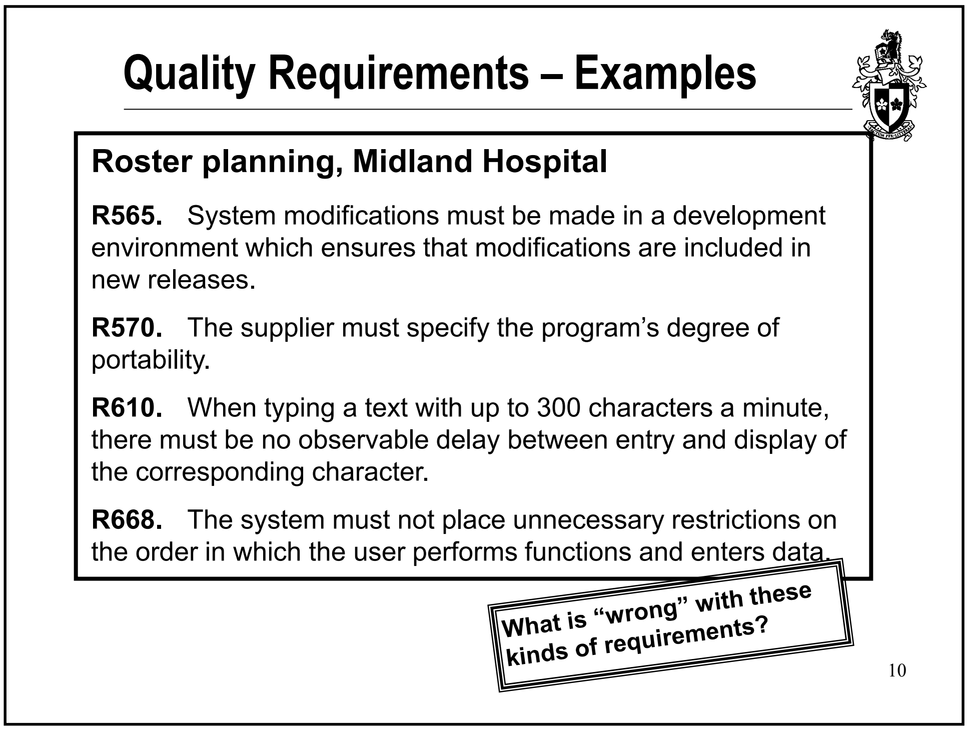
## What is Software Quality?

Software **Quality** is *conformance to*:

- explicitly stated *functional and performance(**quality**) requirements*,
- explicitly documented *development standards*,
- *implicit characteristics* that are expected of all professionally developed software.

9

9



## Quality Requirements – Examples

**Roster planning, Midland Hospital**

**R565.** System modifications must be made in a development environment which ensures that modifications are included in new releases.

**R570.** The supplier must specify the program's degree of portability.

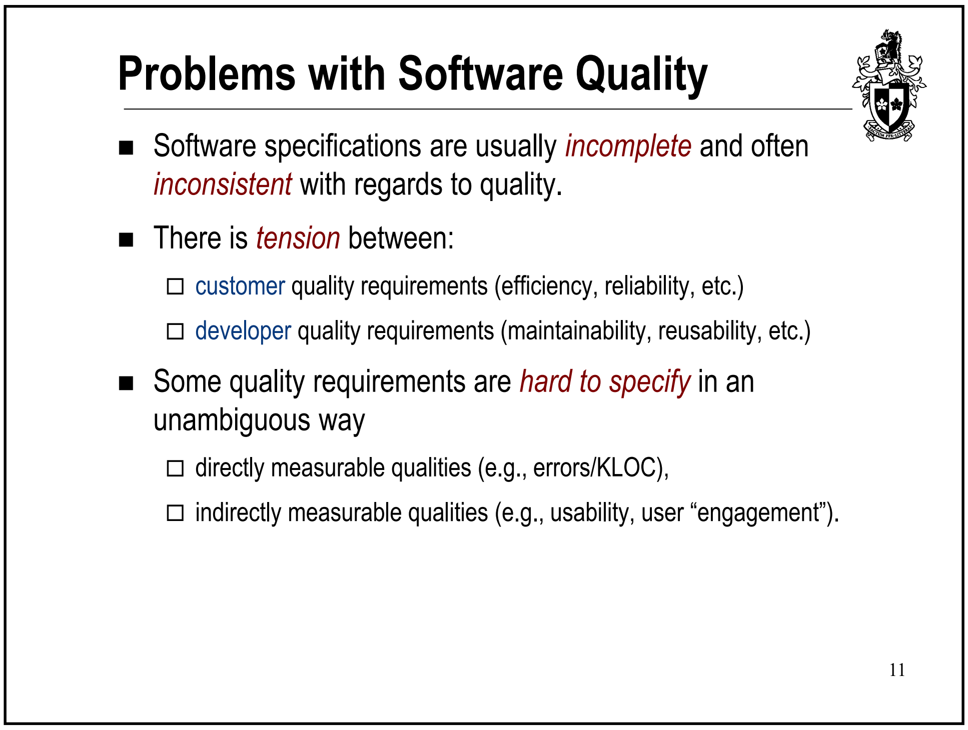
**R610.** When typing a text with up to 300 characters a minute, there must be no observable delay between entry and display of the corresponding character.

**R668.** The system must not place unnecessary restrictions on the order in which the user performs functions and enters data.

**What is "wrong" with these kinds of requirements?**

10

10

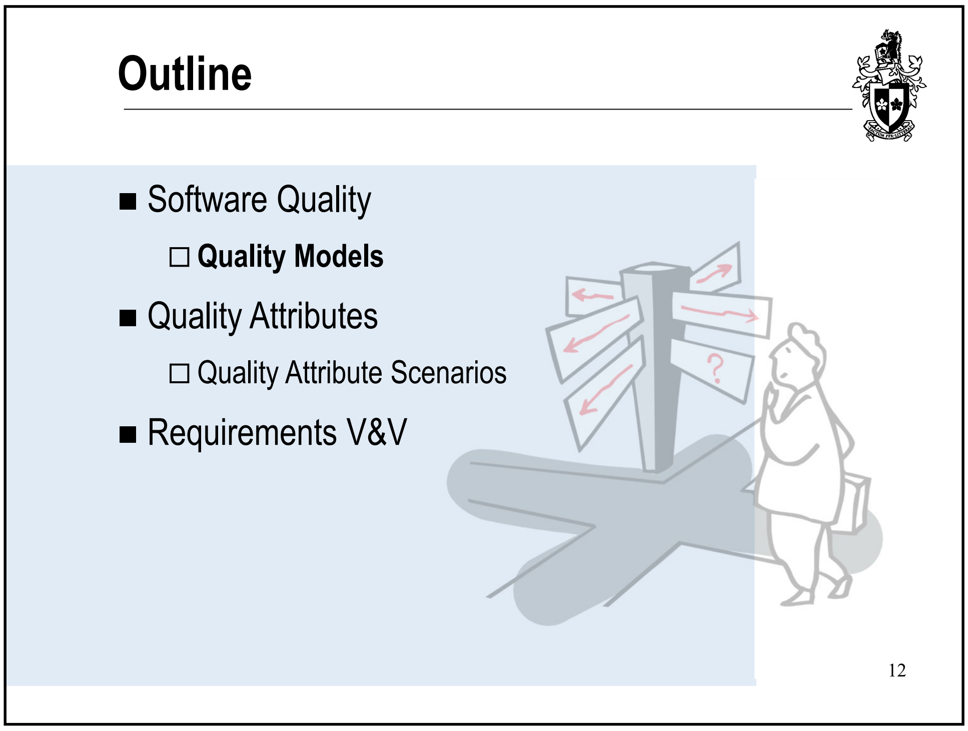


## Problems with Software Quality

- Software specifications are usually *incomplete* and often *inconsistent* with regards to quality.
- There is *tension* between:
  - ☐ customer quality requirements (efficiency, reliability, etc.)
  - ☐ developer quality requirements (maintainability, reusability, etc.)
- Some quality requirements are *hard to specify* in an unambiguous way
  - ☐ directly measurable qualities (e.g., errors/KLOC),
  - ☐ indirectly measurable qualities (e.g., usability, user “engagement”).

11

11

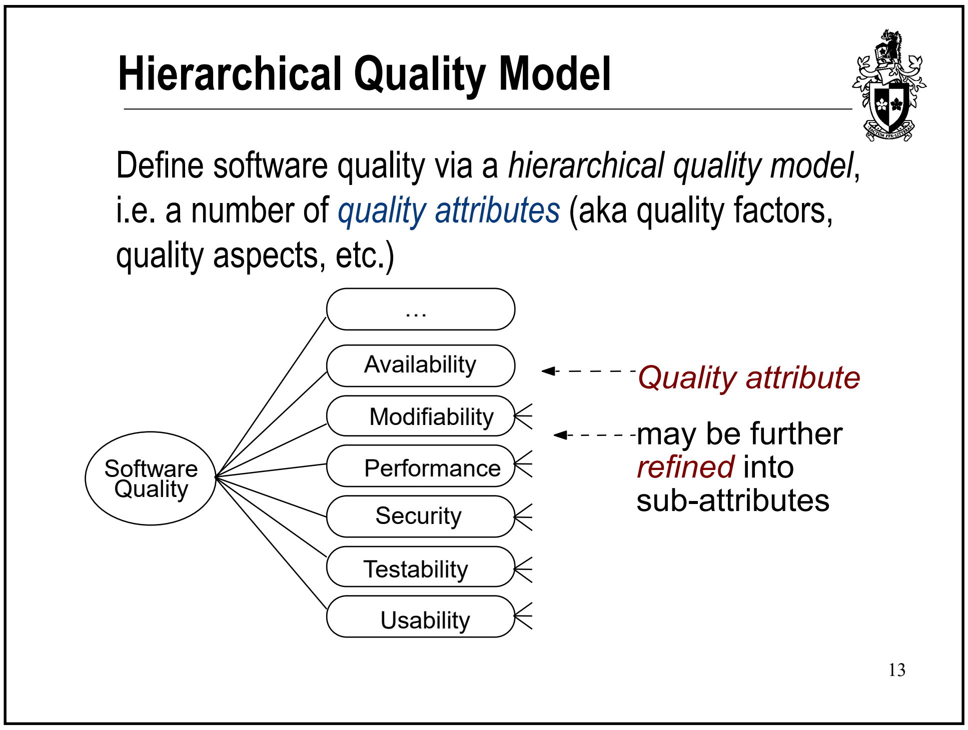


## Outline

- Software Quality
  - ☐ **Quality Models**
- Quality Attributes
  - ☐ Quality Attribute Scenarios
- Requirements V&V

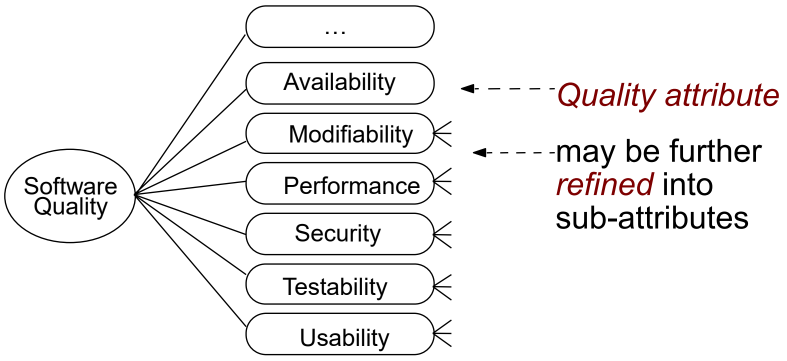
12

12



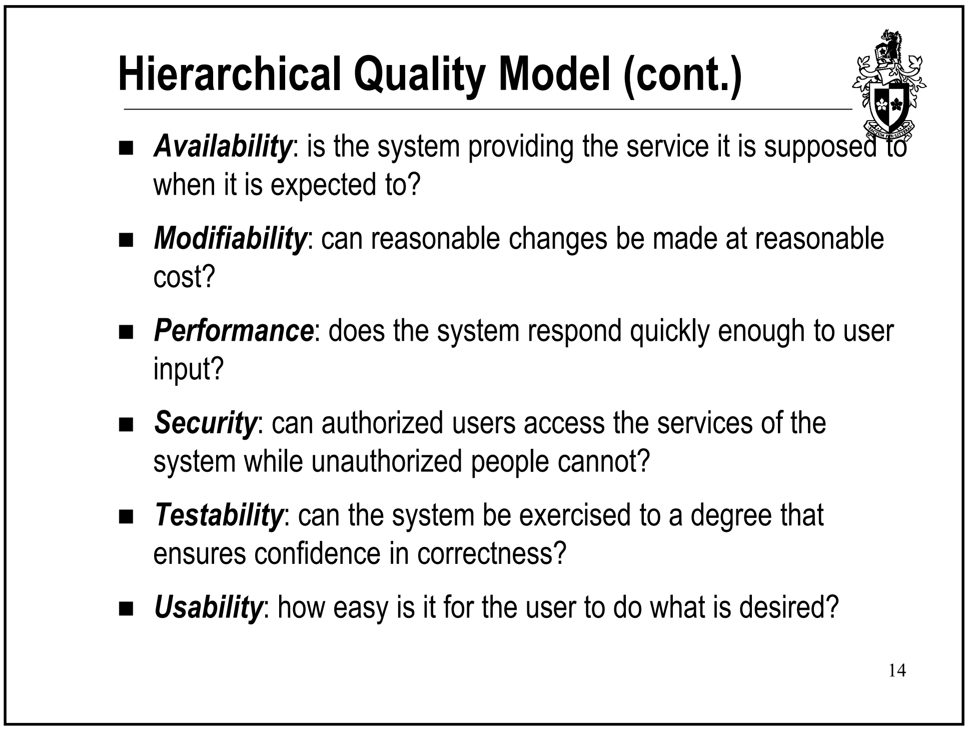
## Hierarchical Quality Model

Define software quality via a *hierarchical quality model*, i.e. a number of *quality attributes* (aka quality factors, quality aspects, etc.)



13

13

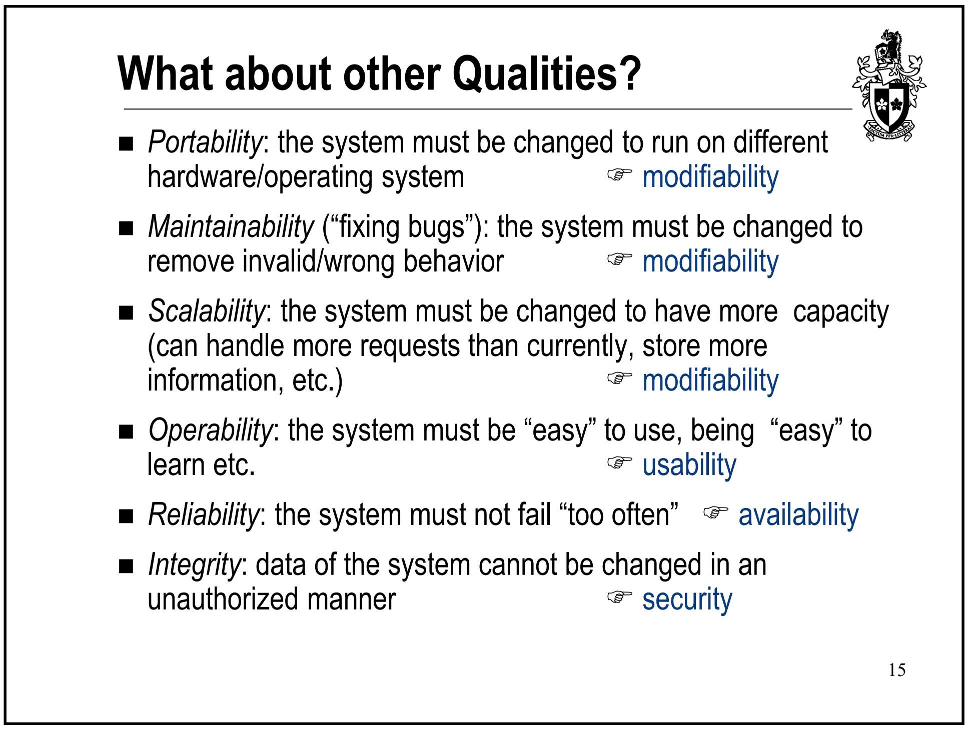


## Hierarchical Quality Model (cont.)

- ***Availability***: is the system providing the service it is supposed to when it is expected to?
- ***Modifiability***: can reasonable changes be made at reasonable cost?
- ***Performance***: does the system respond quickly enough to user input?
- ***Security***: can authorized users access the services of the system while unauthorized people cannot?
- ***Testability***: can the system be exercised to a degree that ensures confidence in correctness?
- ***Usability***: how easy is it for the user to do what is desired?

14

14

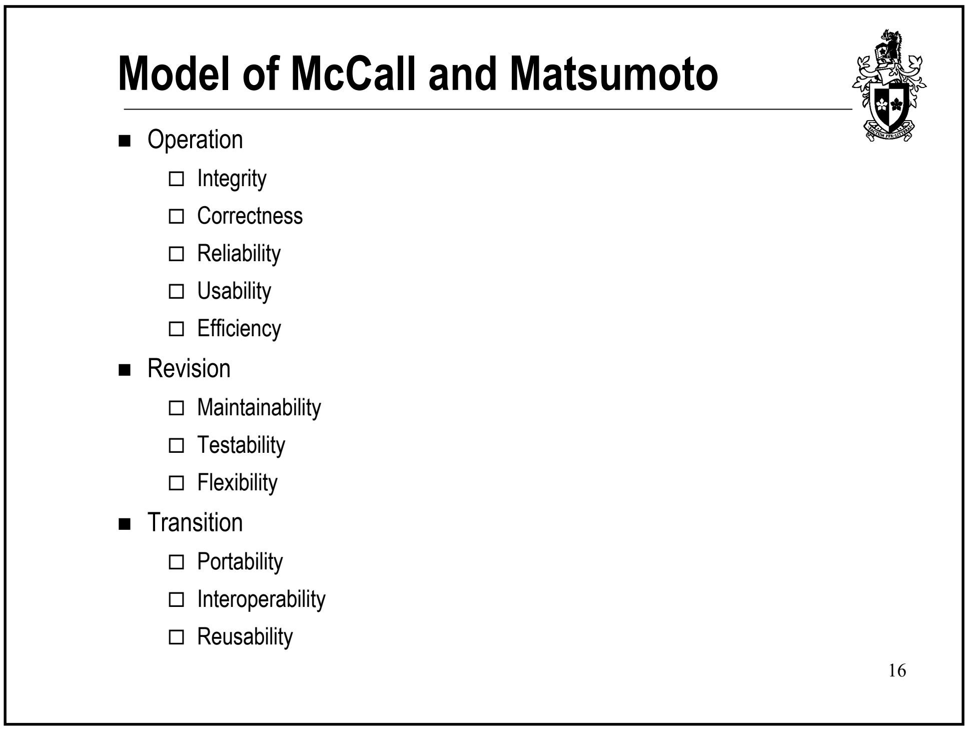


## What about other Qualities?

- *Portability*: the system must be changed to run on different hardware/operating system ☞ modifiability
- *Maintainability* ("fixing bugs"): the system must be changed to remove invalid/wrong behavior ☞ modifiability
- *Scalability*: the system must be changed to have more capacity (can handle more requests than currently, store more information, etc.) ☞ modifiability
- *Operability*: the system must be "easy" to use, being "easy" to learn etc. ☞ usability
- *Reliability*: the system must not fail "too often" ☞ availability
- *Integrity*: data of the system cannot be changed in an unauthorized manner ☞ security

15

15

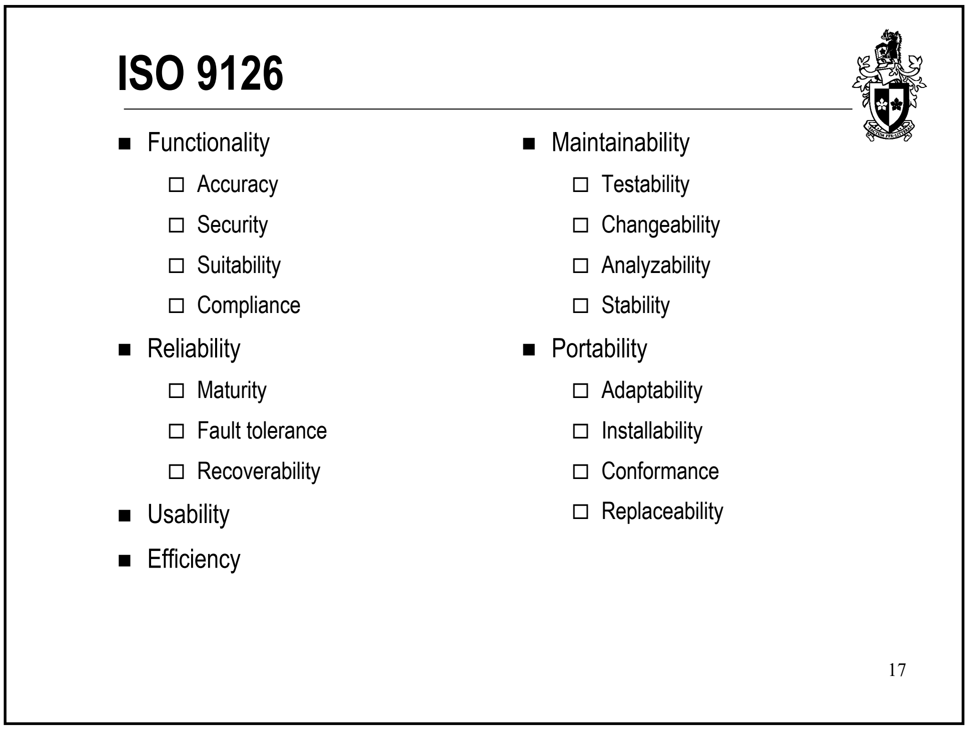


## Model of McCall and Matsumoto

- Operation
  - Integrity
  - Correctness
  - Reliability
  - Usability
  - Efficiency
- Revision
  - Maintainability
  - Testability
  - Flexibility
- Transition
  - Portability
  - Interoperability
  - Reusability

16

16

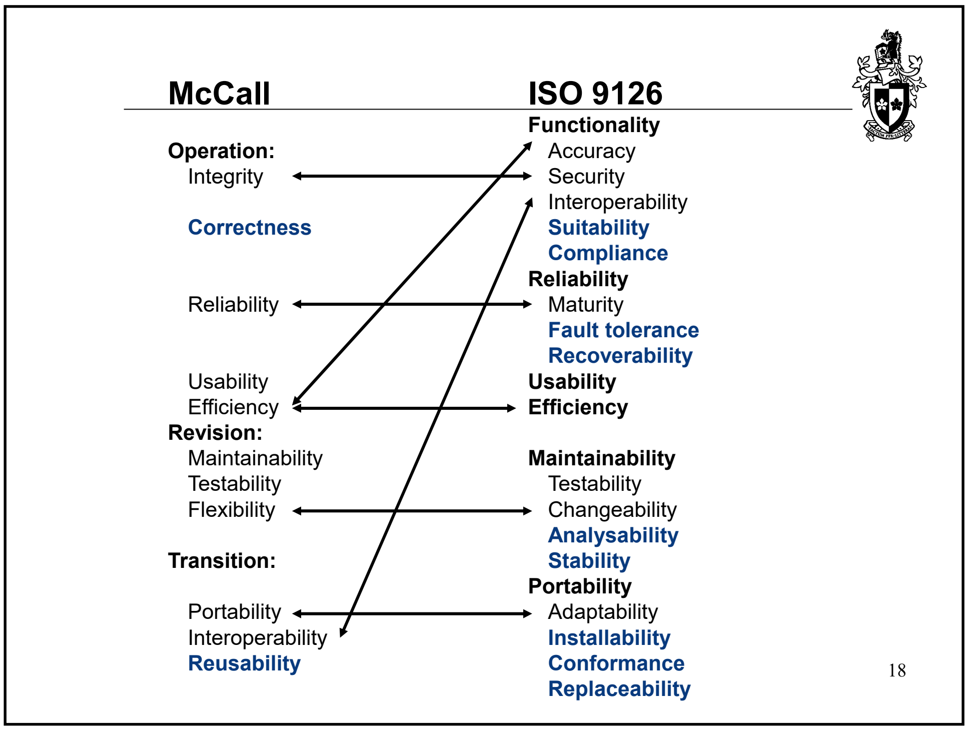


## ISO 9126

- Functionality
  - Accuracy
  - Security
  - Suitability
  - Compliance
- Reliability
  - Maturity
  - Fault tolerance
  - Recoverability
- Usability
- Efficiency
- Maintainability
  - Testability
  - Changeability
  - Analyzability
  - Stability
- Portability
  - Adaptability
  - Installability
  - Conformance
  - Replaceability

17

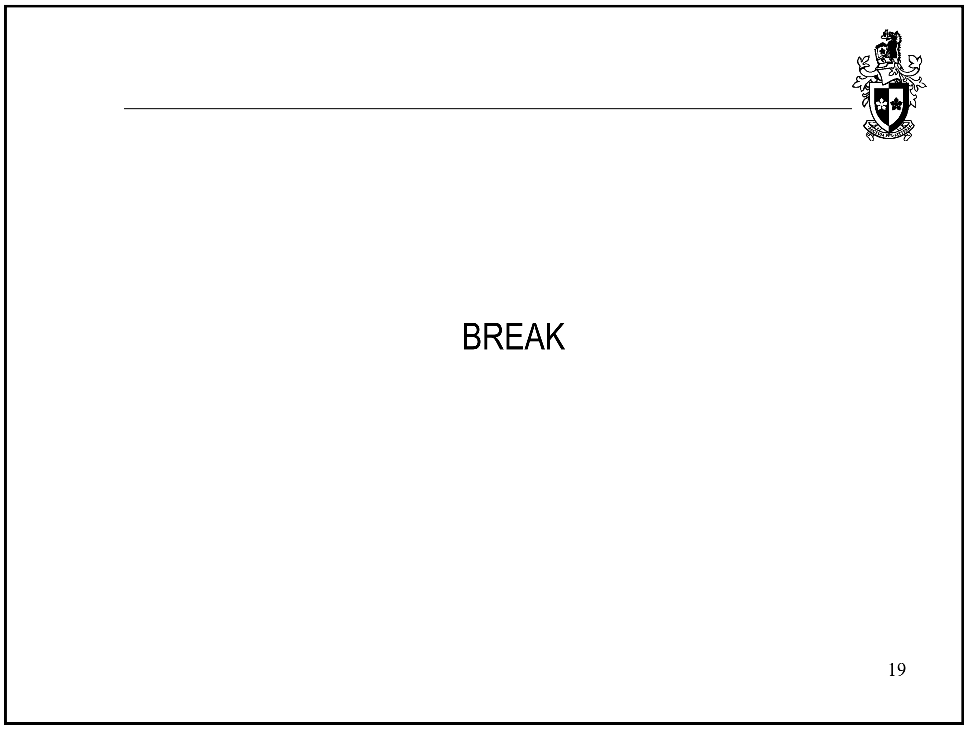
17



| McCall | ISO 9126 |
|---|---|
| | **Functionality** |
| **Operation:** | Accuracy |
| Integrity | Security |
| | Interoperability |
| **Correctness** | **Suitability** |
| | **Compliance** |
| | **Reliability** |
| Reliability | Maturity |
| | **Fault tolerance** |
| | **Recoverability** |
| Usability | **Usability** |
| Efficiency | **Efficiency** |
| **Revision:** | |
| Maintainability | **Maintainability** |
| Testability | Testability |
| Flexibility | Changeability |
| | **Analysability** |
| **Transition:** | **Stability** |
| | **Portability** |
| Portability | Adaptability |
| Interoperability | **Installability** |
| **Reusability** | **Conformance** |
| | **Replaceability** |

18

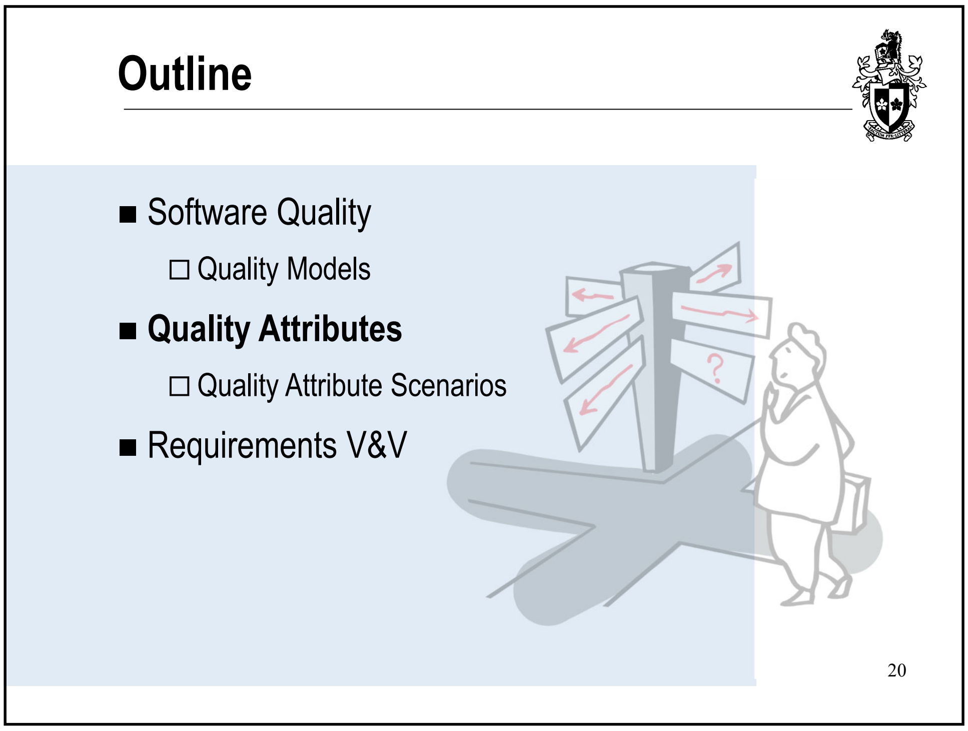
18



BREAK

19

19

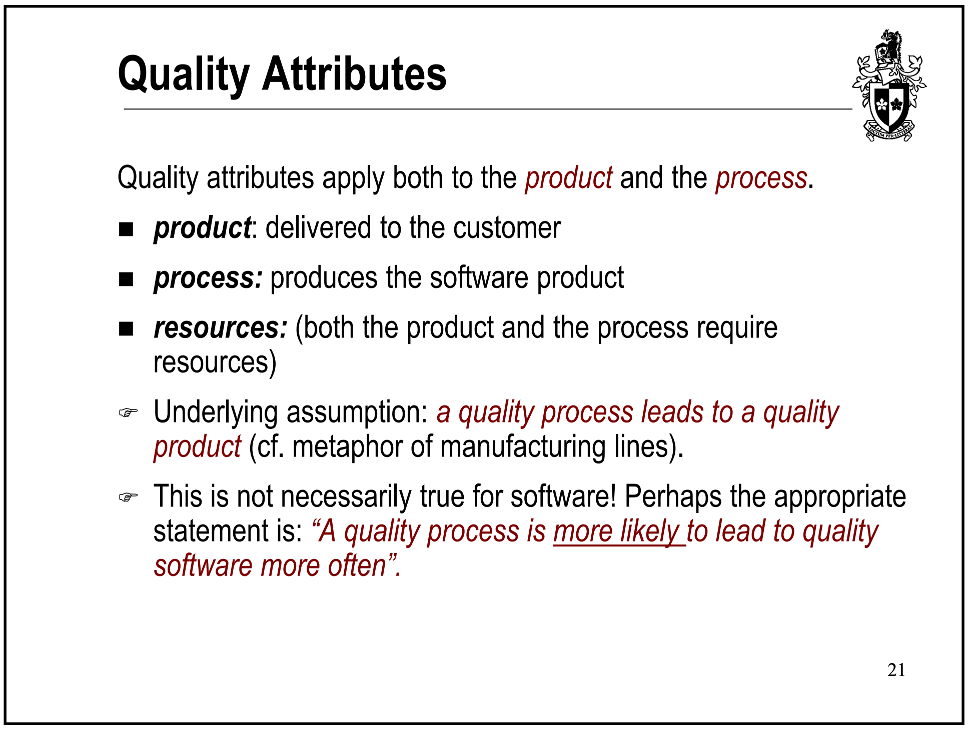


## Outline

- Software Quality
  - Quality Models
- **Quality Attributes**
  - Quality Attribute Scenarios
- Requirements V&V

20

20



## Quality Attributes

Quality attributes apply both to the *product* and the *process*.

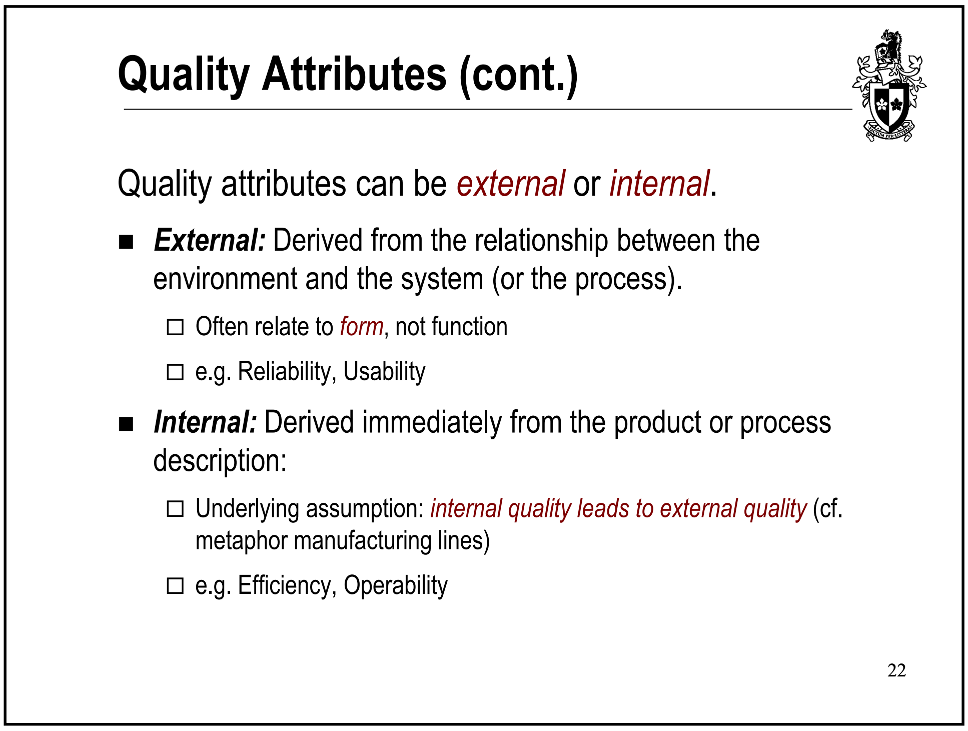
- ***product***: delivered to the customer
- ***process:*** produces the software product
- ***resources:*** (both the product and the process require resources)

☞ Underlying assumption: *a quality process leads to a quality product* (cf. metaphor of manufacturing lines).

☞ This is not necessarily true for software! Perhaps the appropriate statement is: *"A quality process is more likely to lead to quality software more often".*

21

21



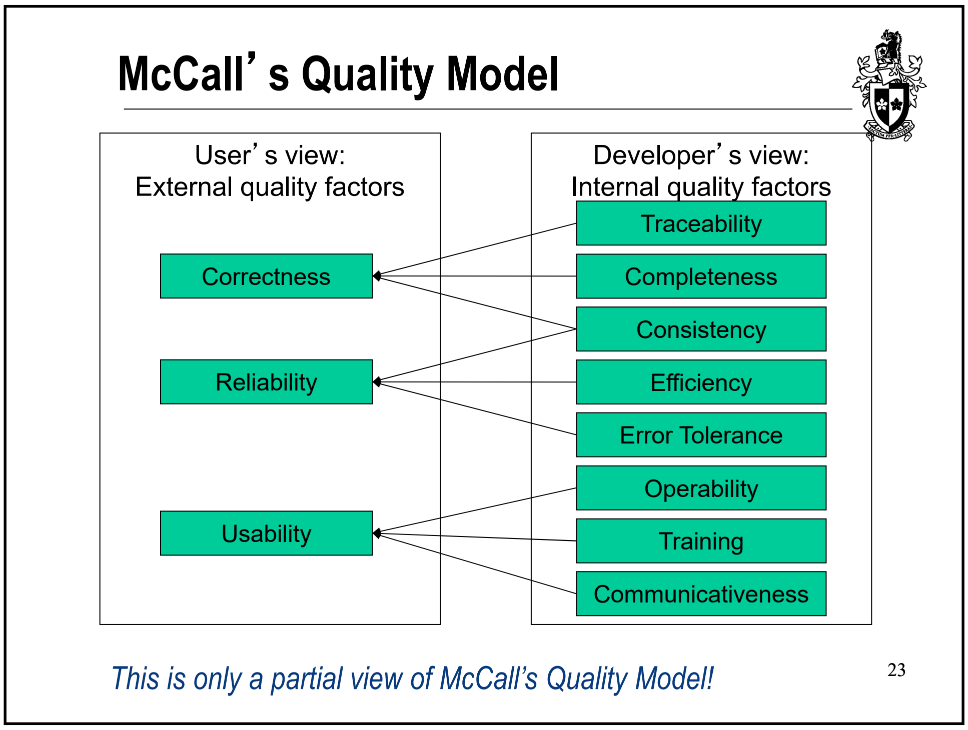
## Quality Attributes (cont.)

Quality attributes can be *external* or *internal*.

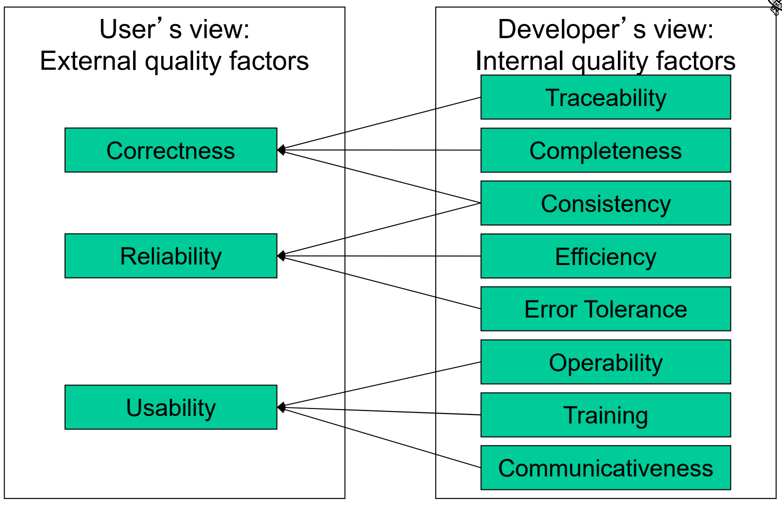
- ***External:*** Derived from the relationship between the environment and the system (or the process).
  - ☐ Often relate to *form*, not function
  - ☐ e.g. Reliability, Usability
- ***Internal:*** Derived immediately from the product or process description:
  - ☐ Underlying assumption: *internal quality leads to external quality* (cf. metaphor manufacturing lines)
  - ☐ e.g. Efficiency, Operability

22

22



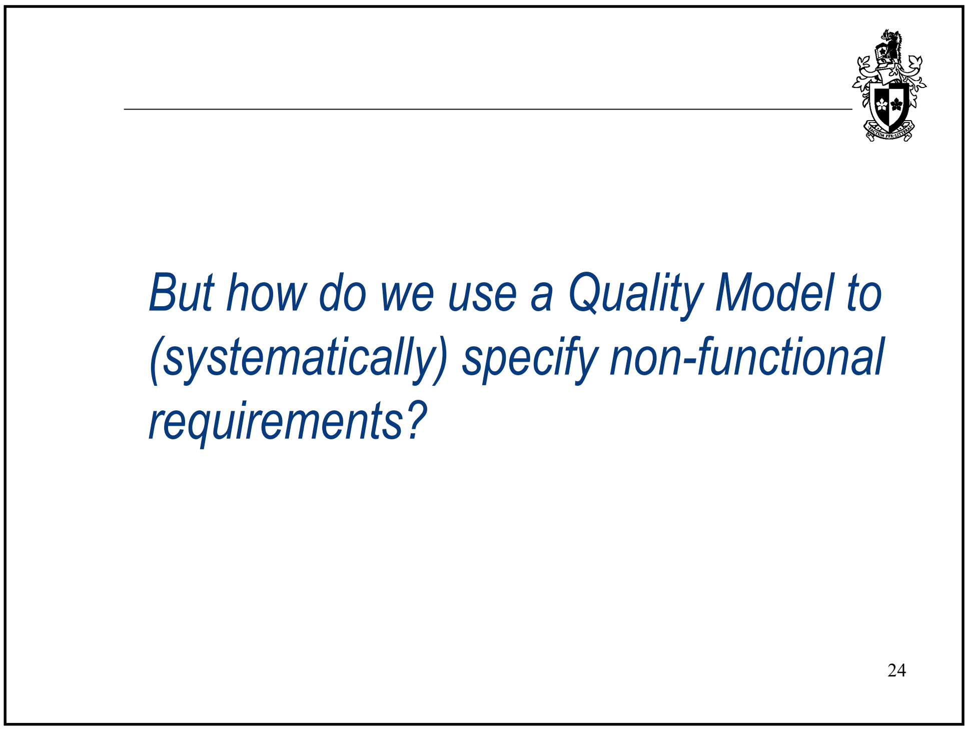
## McCall’s Quality Model



*This is only a partial view of McCall’s Quality Model!*

23

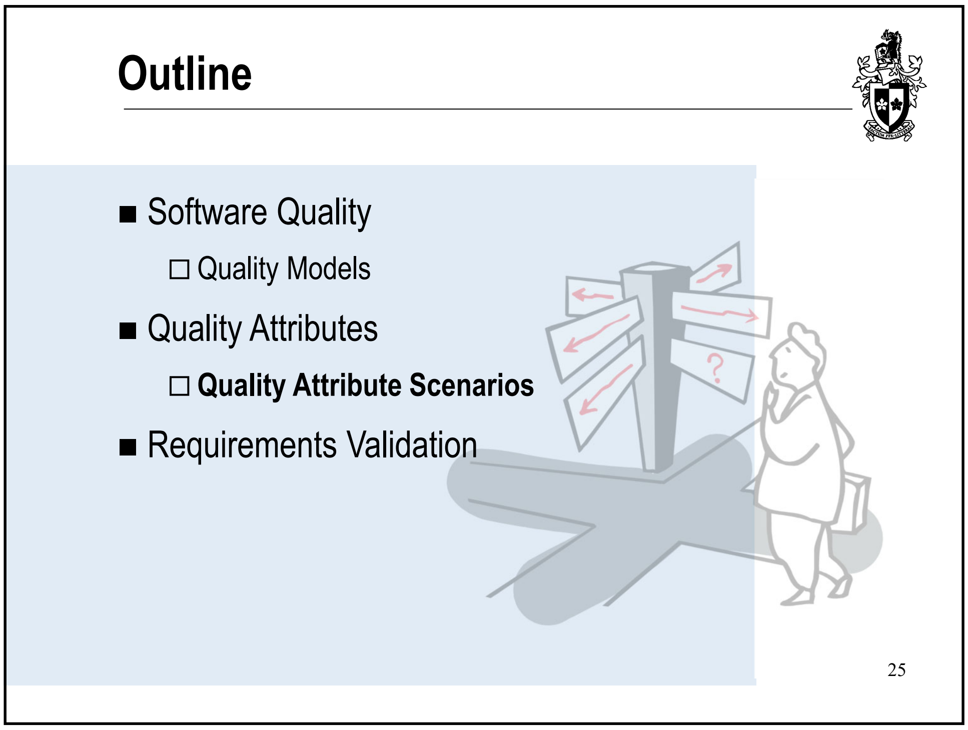
23



*But how do we use a Quality Model to (systematically) specify non-functional requirements?*

24

24

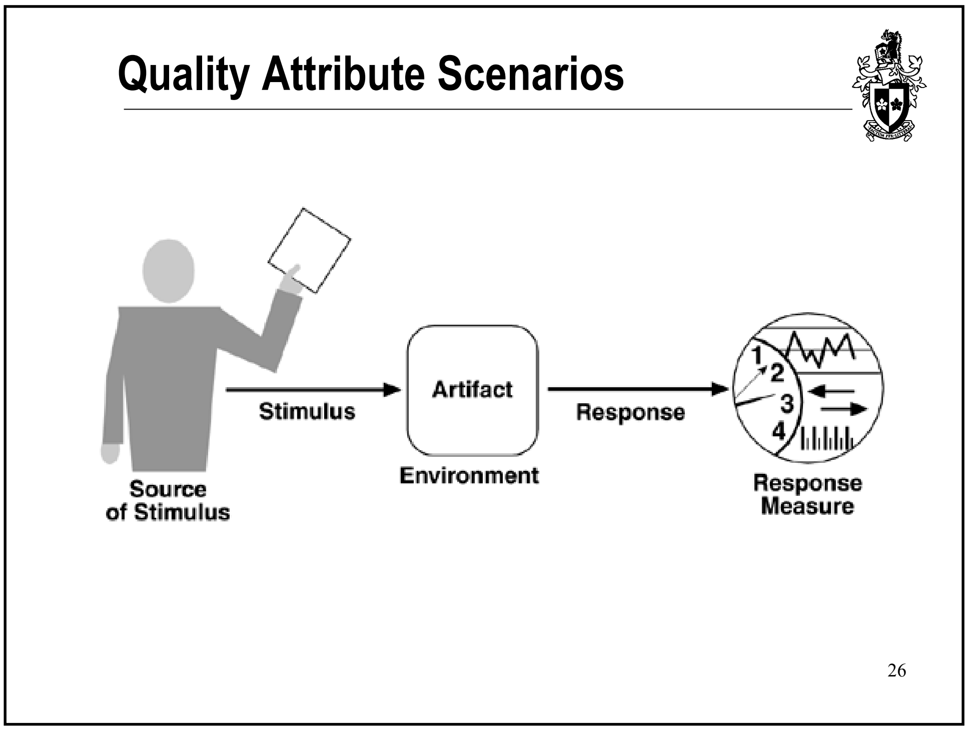


## Outline

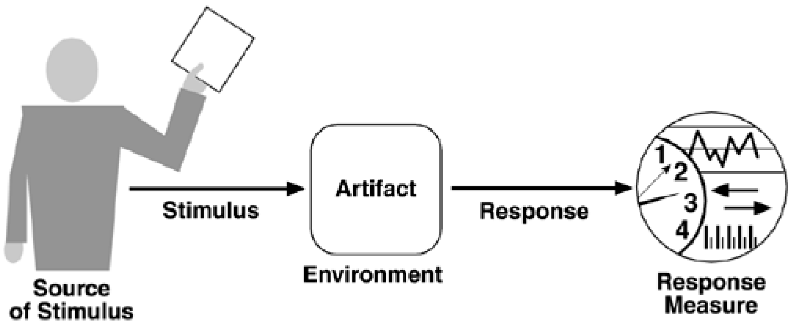
- Software Quality
  - Quality Models
- Quality Attributes
  - **Quality Attribute Scenarios**
- Requirements Validation

25

25

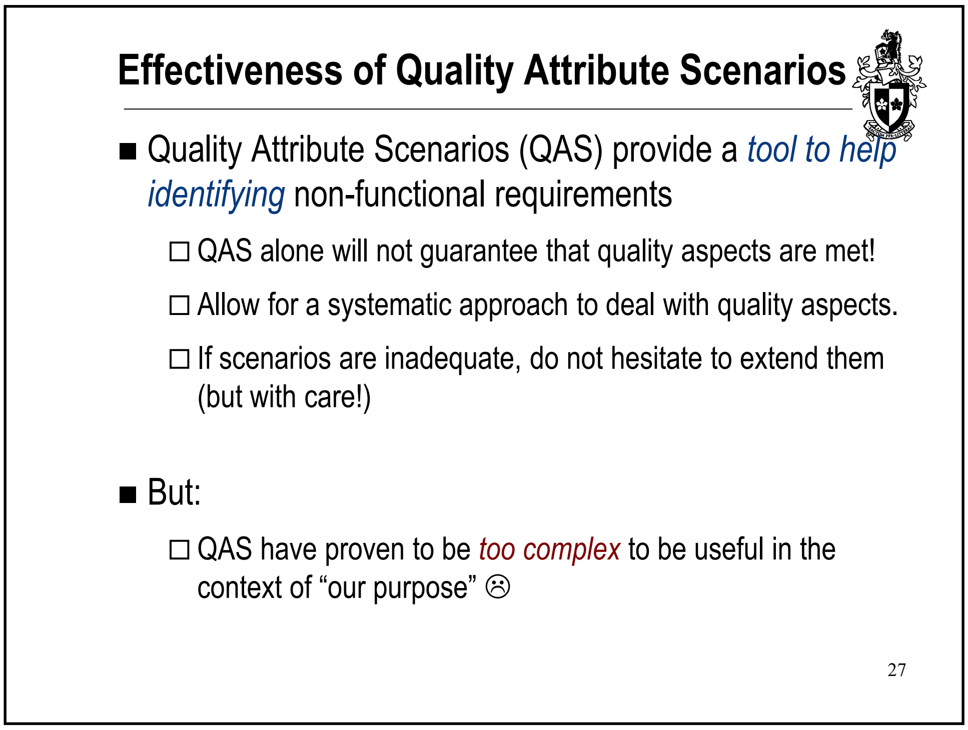


## Quality Attribute Scenarios



26

26

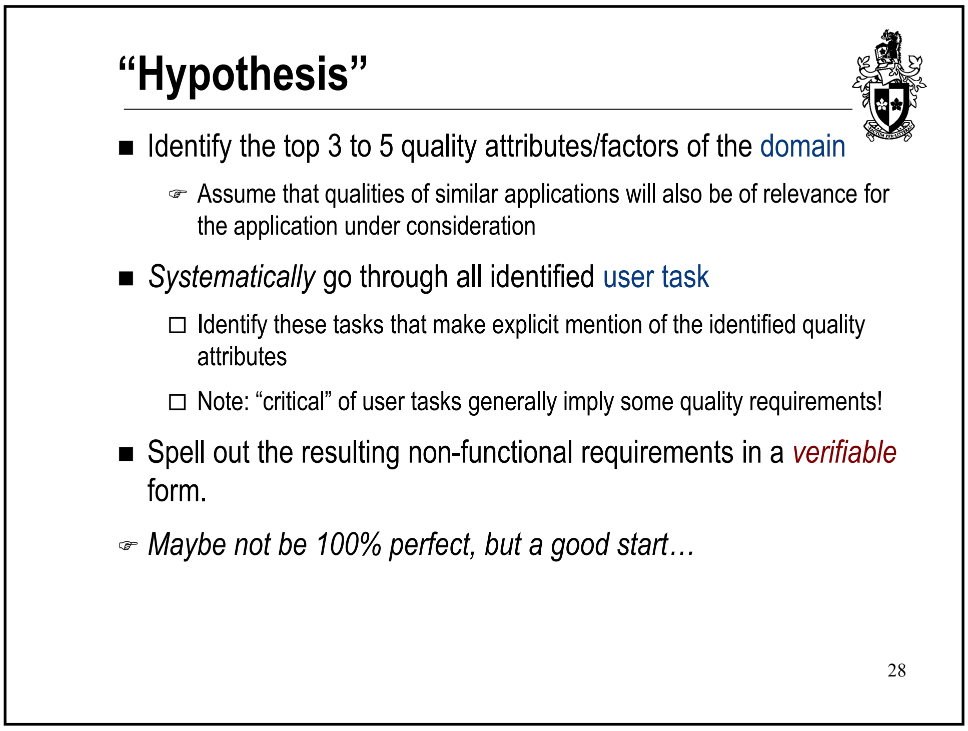


## Effectiveness of Quality Attribute Scenarios

- Quality Attribute Scenarios (QAS) provide a *tool to help identifying* non-functional requirements
  - ☐ QAS alone will not guarantee that quality aspects are met!
  - ☐ Allow for a systematic approach to deal with quality aspects.
  - ☐ If scenarios are inadequate, do not hesitate to extend them (but with care!)
- But:
  - ☐ QAS have proven to be *too complex* to be useful in the context of "our purpose" ☹

27

27



## "Hypothesis"

- Identify the top 3 to 5 quality attributes/factors of the domain
  - ☞ Assume that qualities of similar applications will also be of relevance for the application under consideration
- *Systematically* go through all identified user task
  - ☐ Identify these tasks that make explicit mention of the identified quality attributes
  - ☐ Note: "critical" of user tasks generally imply some quality requirements!
- Spell out the resulting non-functional requirements in a *verifiable* form.

☞ *Maybe not be 100% perfect, but a good start…*

28

28

## Examples

For the Hotel Information System:

- Identify the top 3 to 5 quality attributes/factors …
- *Systematically* go through all identified user task …
  - User …
  - “critical” parts …
- Spell out the resulting non-functional requirements in a *verifiable* form …

… see tutorial

29

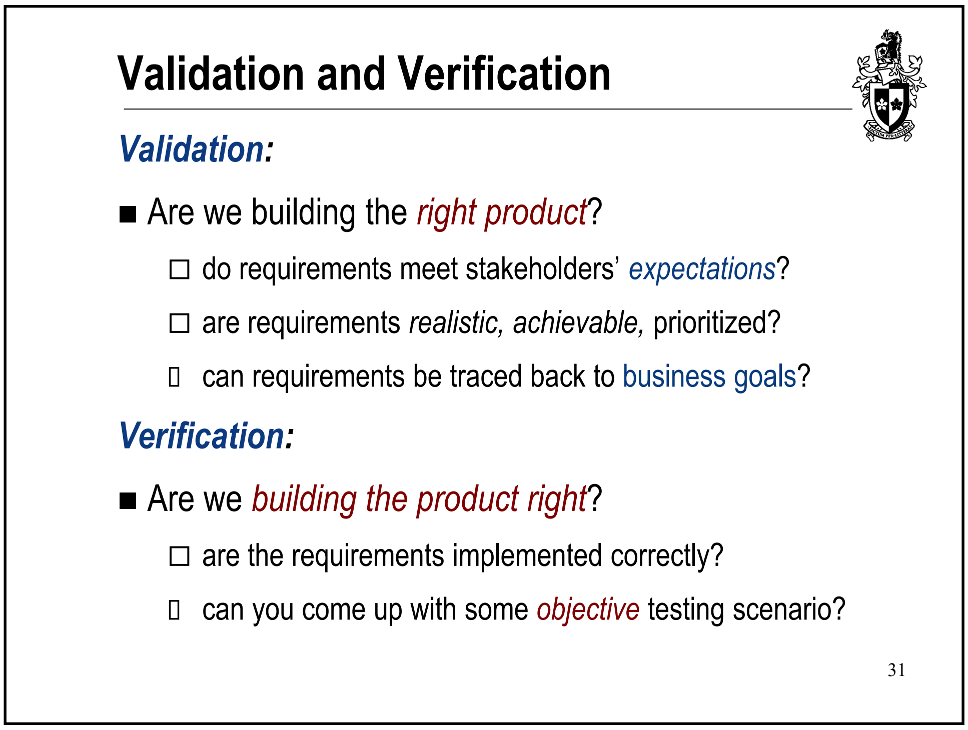
29

## Outline

- Software Quality
  - Quality Models
- Quality Attributes
  - Quality Attribute Scenarios
- **Requirements V&V**

30

30



## Validation and Verification

***Validation:***

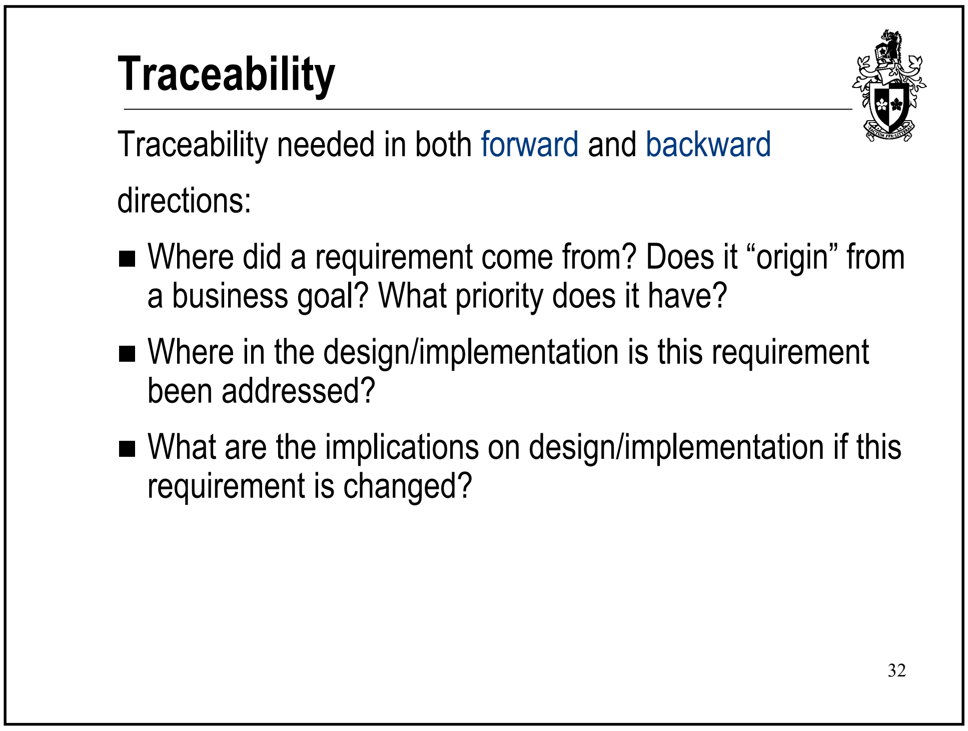
- Are we building the *right product*?
  - do requirements meet stakeholders' *expectations*?
  - are requirements *realistic, achievable,* prioritized?
  - can requirements be traced back to business goals?

***Verification:***

- Are we *building the product right*?
  - are the requirements implemented correctly?
  - can you come up with some *objective* testing scenario?

31

31



## Traceability

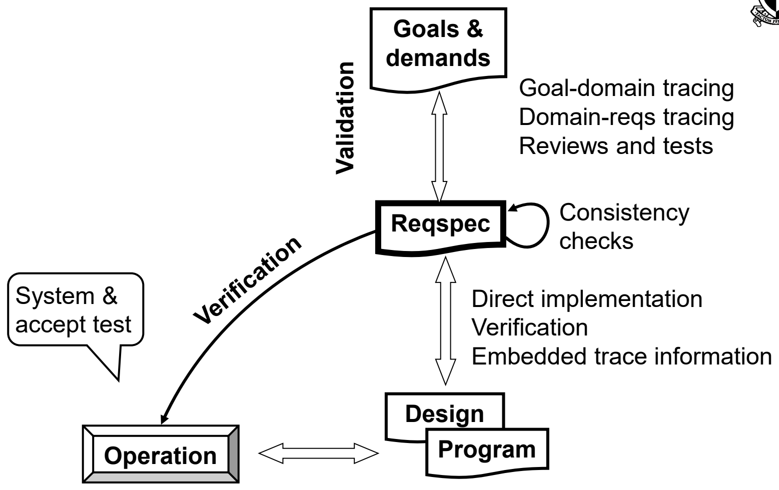
Traceability needed in both forward and backward directions:

- Where did a requirement come from? Does it "origin" from a business goal? What priority does it have?
- Where in the design/implementation is this requirement been addressed?
- What are the implications on design/implementation if this requirement is changed?

32

32

## Traceability (cont.)



From: Soren Lauesen: Software Requirements
© Pearson / Addison-Wesley 2002

33

33

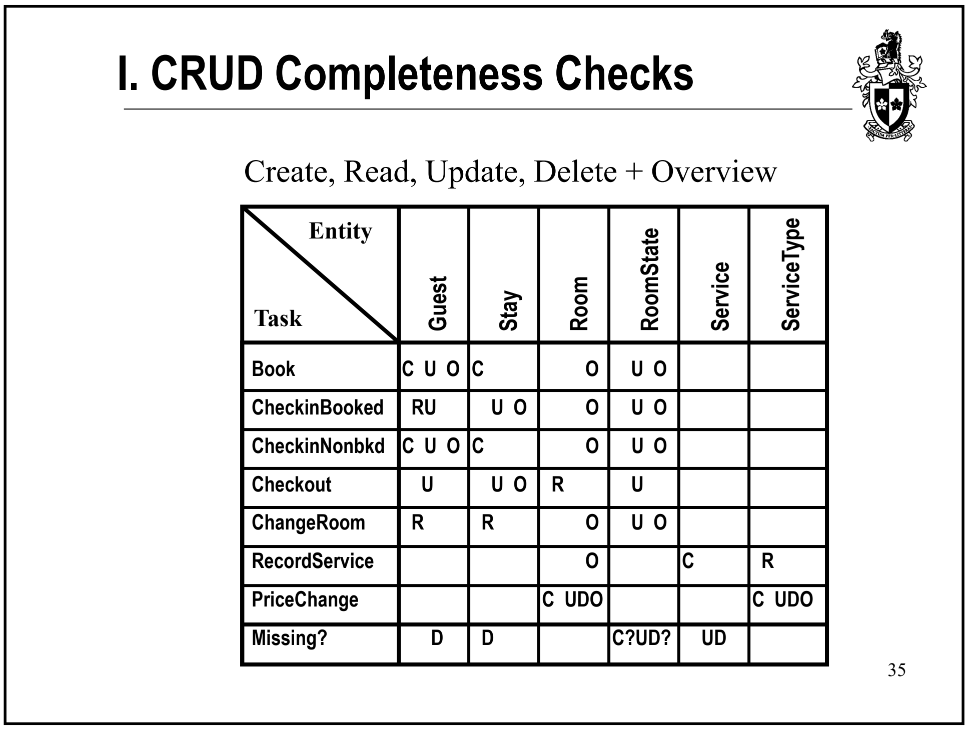
## Quality Criteria for a Requirements Spec.

- *Correct*: each requirement reflects a need
- *Traceable*: to goals/purpose (also to design/implementation…)
- *Complete*: all necessary requirements are included
- *Consistent*: all parts of SRS match; there are no inconsistencies
- *Verifiable*: possible to see whether it can be met/tested

- *Unambiguous*: all stakeholders agree on meaning
- *Ranked for importance/stability*: priorities; expected changes
- *Modifiable*: easy to change; maintain consistency
- *Understandable by customer and developers*

34

34



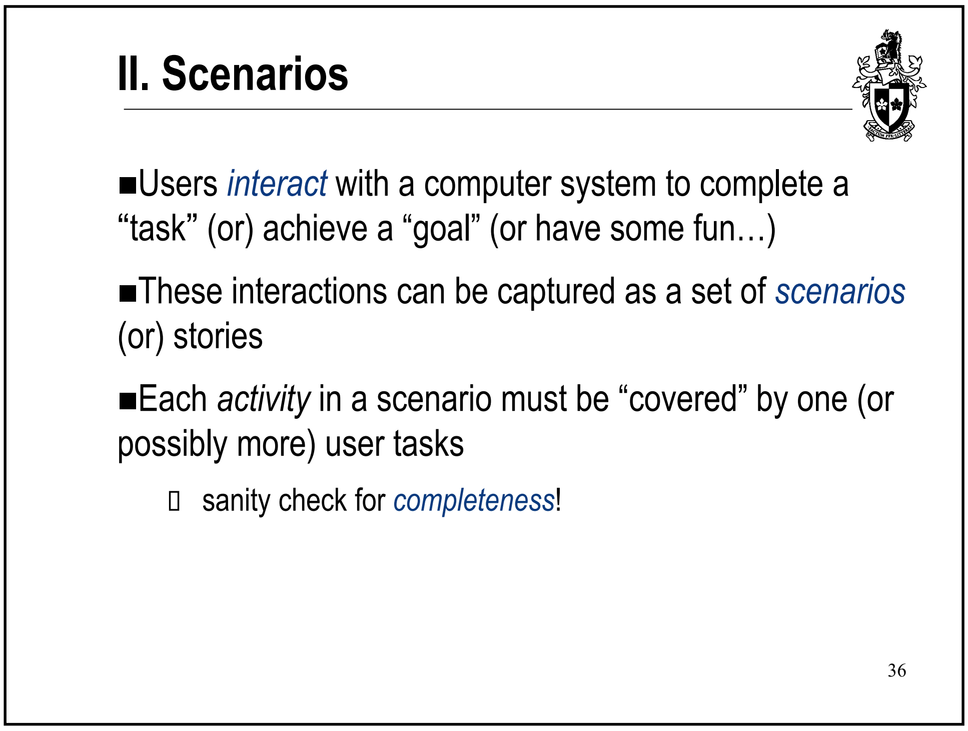
## I. CRUD Completeness Checks

Create, Read, Update, Delete + Overview

| Task \ Entity | Guest | Stay | Room | RoomState | Service | ServiceType |
|---|---|---|---|---|---|---|
| Book | C U O | C | O | U O | | |
| CheckinBooked | RU | U O | O | U O | | |
| CheckinNonbkd | C U O | C | O | U O | | |
| Checkout | U | U O | R | U | | |
| ChangeRoom | R | R | O | U O | | |
| RecordService | | | O | | C | R |
| PriceChange | | | C UDO | | | C UDO |
| Missing? | D | D | | C?UD? | UD | |

35

35

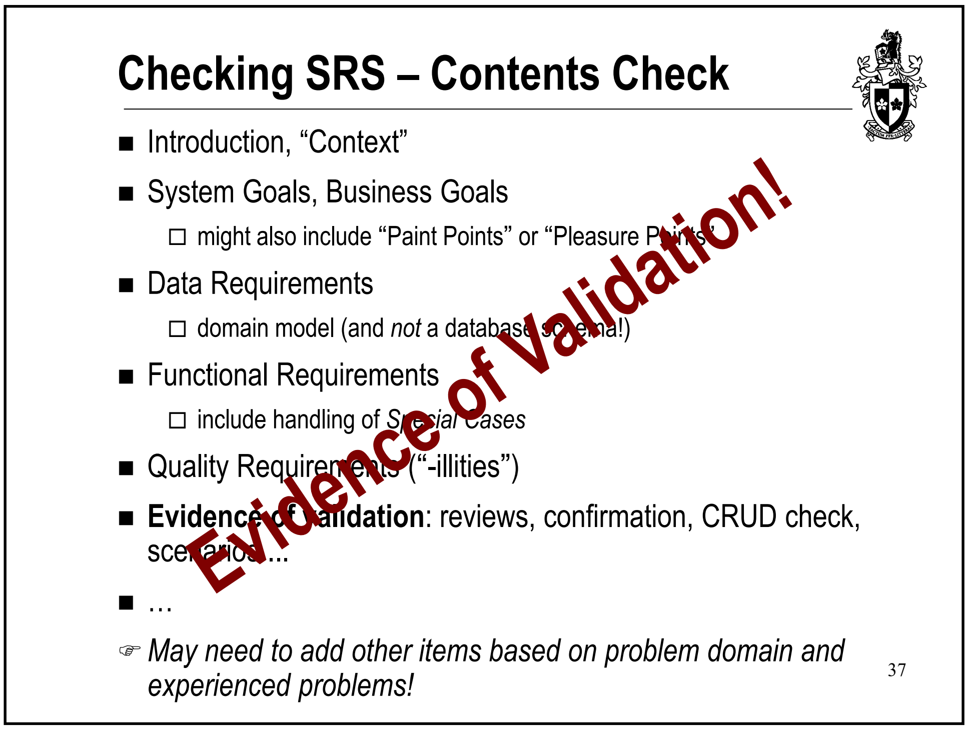


## II. Scenarios

- ■Users *interact* with a computer system to complete a “task” (or) achieve a “goal” (or have some fun…)
- ■These interactions can be captured as a set of *scenarios* (or) stories
- ■Each *activity* in a scenario must be “covered” by one (or possibly more) user tasks
  - sanity check for *completeness*!

36

36



## Checking SRS – Contents Check

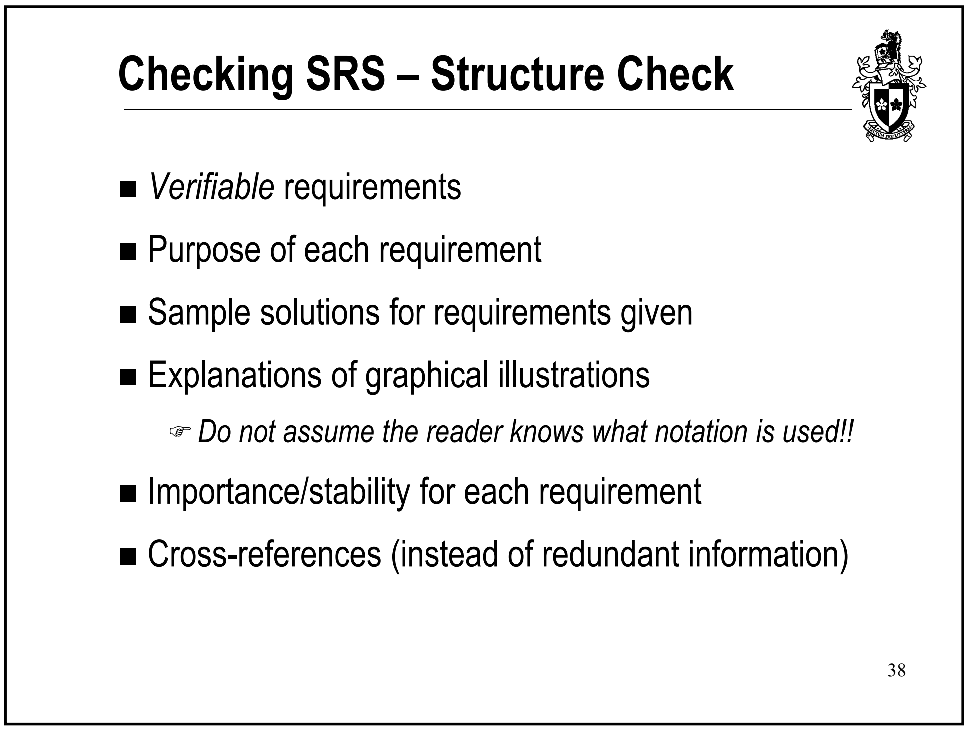
- Introduction, "Context"
- System Goals, Business Goals
  - might also include "Paint Points" or "Pleasure Points"
- Data Requirements
  - domain model (and *not* a database schema!)
- Functional Requirements
  - include handling of *Special Cases*
- Quality Requirements ("-illities")
- **Evidence of Validation**: reviews, confirmation, CRUD check, scenarios...
- …

☞ *May need to add other items based on problem domain and experienced problems!*

Evidence of Validation!

37

37



## Checking SRS – Structure Check

- *Verifiable* requirements
- Purpose of each requirement
- Sample solutions for requirements given
- Explanations of graphical illustrations
  - ☞ *Do not assume the reader knows what notation is used!!*
- Importance/stability for each requirement
- Cross-references (instead of redundant information)

38

38

## Questions for Consideration

1. What is throughput? How is it measured? Why is it important to make a distinction between *average* throughput and *peak* throughput?
2. What is scalability? Explain the differences between scaling up and scaling out.
3. In regards to Security, what is the difference between Authentication and Authorization and how do they provide security to an application?
4. If one wishes to specify Testability as a quality requirement for a product, how can this requirement possibly be expressed?
5. What are the most important quality attributes for an online banking system? Explain your rationale.

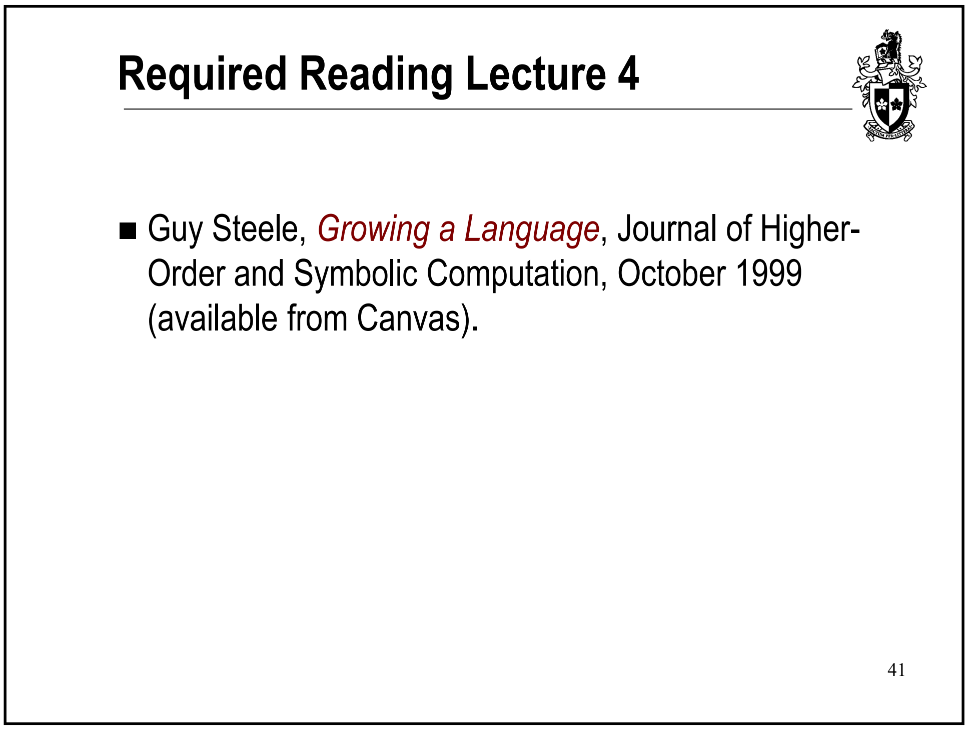
39

39

## Question to Answer Week 3

40

40



## Required Reading Lecture 4

- Guy Steele, *Growing a Language*, Journal of Higher-Order and Symbolic Computation, October 1999 (available from Canvas).

41

41