University of Tripoli – Faculty of Information Technology

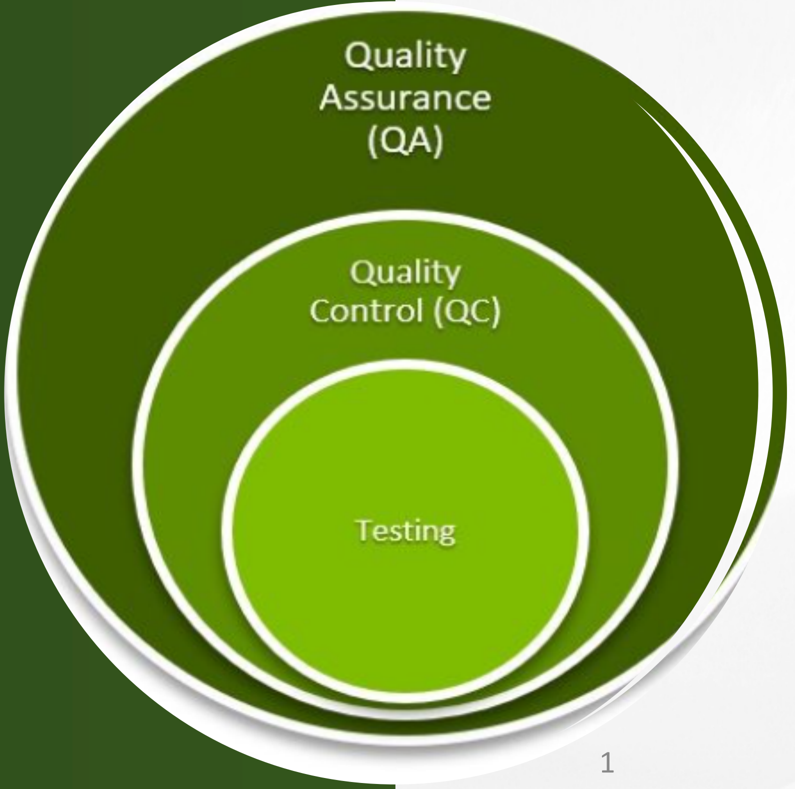
Software Engineering Department

Software Quality Assurance

ITSE421

Spring 2024

By Marwa Solla



Software Quality Assurance and Testing

Lecture 2 : Quality Factors

Tripoli University - Faculty of Information Technology
Marwa Solla
m.solla@uot.edu.ly

Spring 2024

What We Learn In This Lecture

Quality Factors

McCall's Model

Alternatives Models

What is a Software quality factor ?

- ***Definition***

A software quality factor is defined by "a non– functional requirement for a software program which is not called up by the customer's contract, but nevertheless is a desirable requirement which enhances the quality of the software program."

Classifications of software requirements into software quality factors

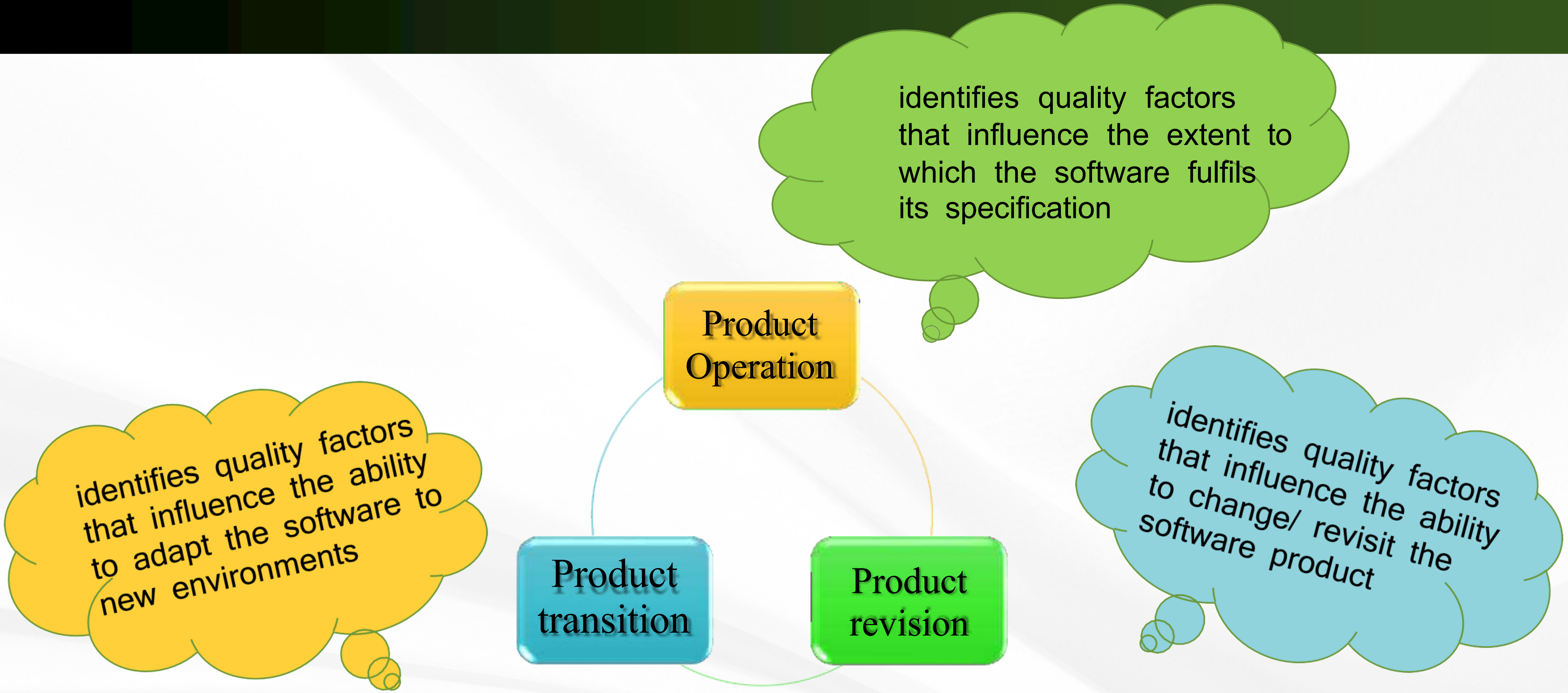
- Several models of software quality factors and their categorization in factor categories have been suggested over the years.
- The classic model of software quality factors, suggested by McCall, consists of 11 factors .
- The 11 factors are grouped into three categories : product operation, product revision and product transition factors.

McCall’s Quality Factors

❑Three categories:

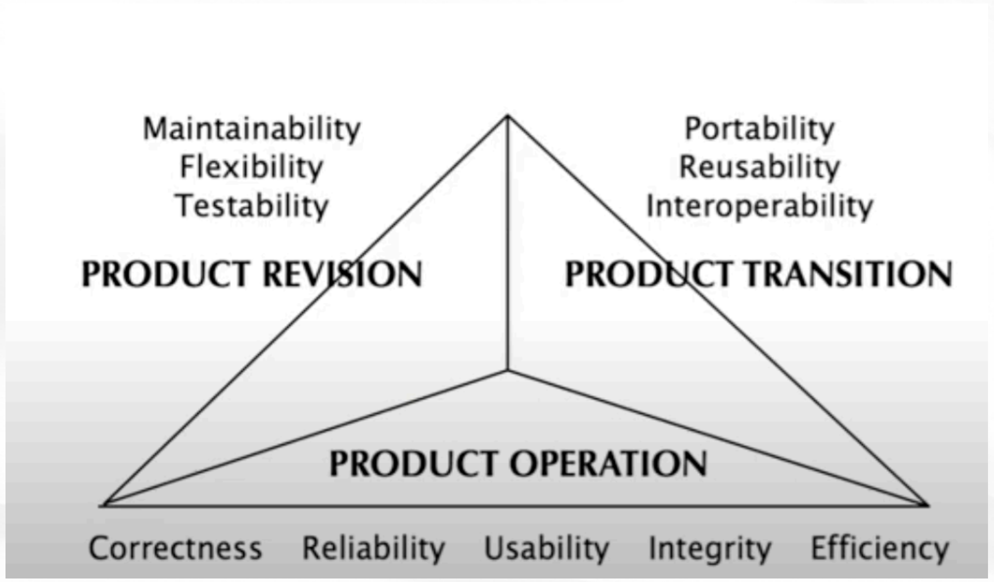
- Product Operation Factors
  - How well it runs….
  - Correctness, reliability, efficiency, integrity, and usability
- Product Revision Factors
  - How well it can be changed, tested, and redeployed.
  - Maintainability; flexibility; testability
- Product Transition Factors
  - How well it can be moved to different platforms and interface with other systems
  - Portability; Reusability; Interoperability

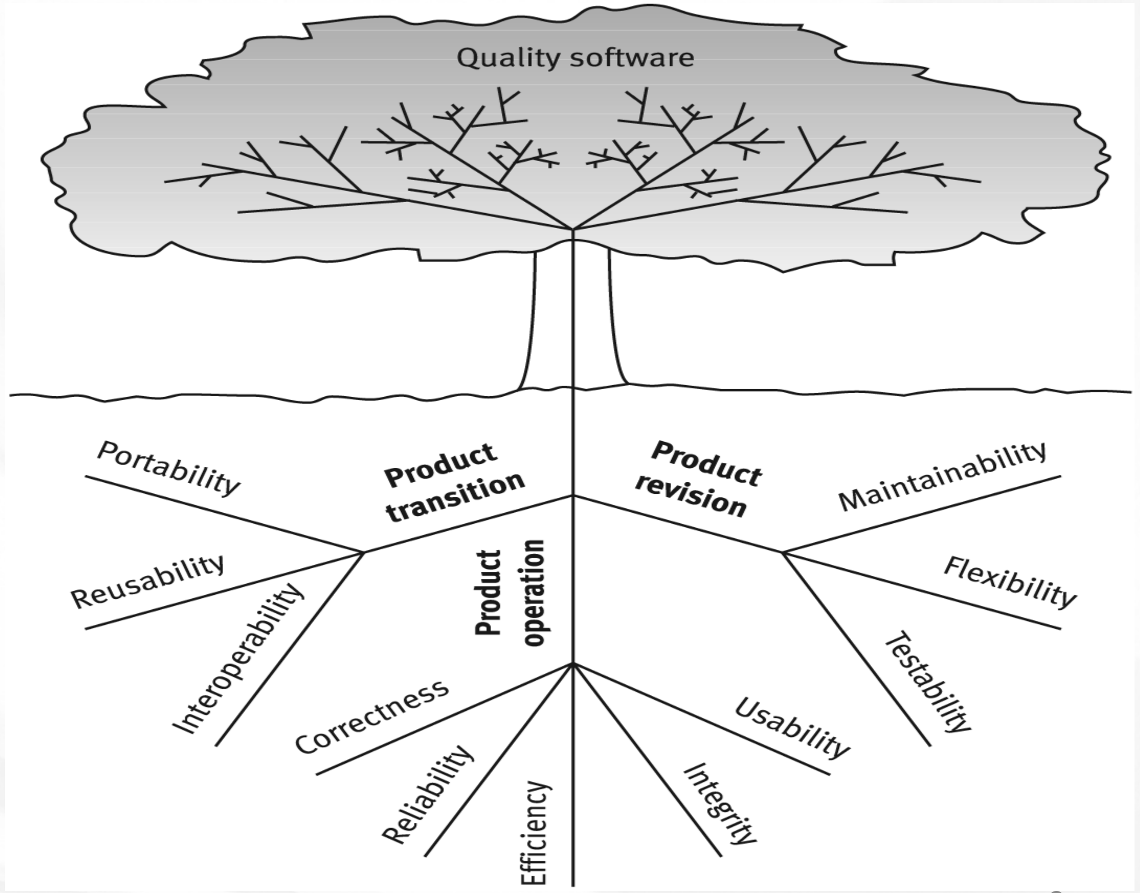
The McCall Model


identifies quality factors that influence the extent to which the software fulfils its specification
Product Operation
identifies quality factors that influence the ability to adapt the software to new environments
Product transition
Product revision
identifies quality factors that influence the ability to change/ revisit the software product

McCall's Quality Factors

McCall's Quality Factors Model tree





Product operation software quality factors

According to McCall’s model, five software quality factors are included in the product operation category, all of which deal with requirements that directly affect *the daily operation* of the software. These factors are as follows.

- Correctness
- Reliability
- Efficiency
- Integrity
- Usability

Product operation software quality factors

1. Correctness.

- **Correctness**– The functionality should match the specification
- The extent to which software meets its requirements specification.
- Correctness requirements are defined in a list of the software system's required outputs.
- '*correctness*' issues are arising from the requirements documentation and the specification of the outputs.
- Correctness is the prime quality. If a system does not do what it is supposed to do, everything else about it

Product operation software quality factors

Examples include:

- **Specifying accuracies for correct outputs** at, say, NLT <1% errors, that could be affected by inaccurate data or faulty calculations;
- **Specifying** the **completeness of the outputs** provided, which can be impacted by incomplete data
- **Specifying the standards** for coding and documenting the software system

Product operation software quality factors

- **2. Reliability Requirements.**
- (remember, this quality factor is specified in the specs!)
- Reliability requirements deal with the failure to provide service.
  - Address failure rates either overall or to required functions.
  - They determine the maximum allowed software system failure rate, and can refer to the entire system or to one or more of its separate functions

- *Example specs:*
  - A heart monitoring system must have a failure rate of less than one per million cases.

Product operation software quality factors

3. Efficiency Requirements.

- Deals with the hardware resources needed to perform the functions of the software.The number of hardware resources and code the software, needs to perform a function.
- The main hardware resources to be considered are the computer's processing capabilities (measured in MIPS — million instructions per second, MHz or megahertz — million cycles per second.
- Data storage capabilities measured in MB or TB; communication lines (usually measured in KBPS, MBPS, or GBPS).
  - *Example* spec: simply very slow communications…

Product operation software quality factors

4. **Integrity** — deal with system security that prevent unauthorized persons access.

- The extent to which the software can control an unauthorized person from accessing the data or software.

5. **Usability Requirements** — deals with the scope of staff resources needed to train new employees and to operate the software system.

The extent of effort required to learn, operate, and understand the functions of the software.

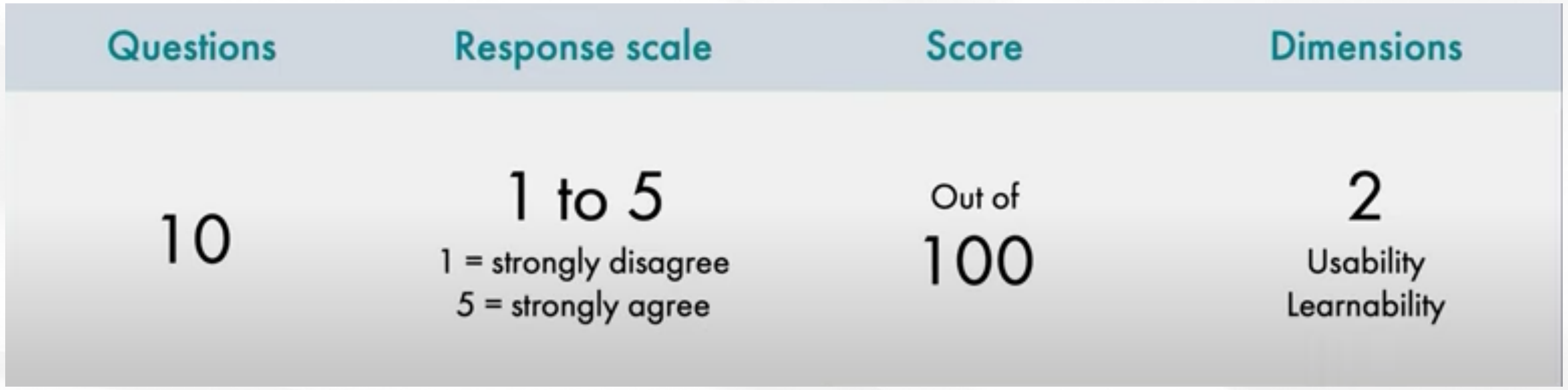
- Deals with learnability, utility, and more. (me)

- Example spec: A staff member should be able to process n transactions / unit time.

Usability measurement scale

► The System Usability Scale (SUS) is the most frequently used questionnaire to measure usability

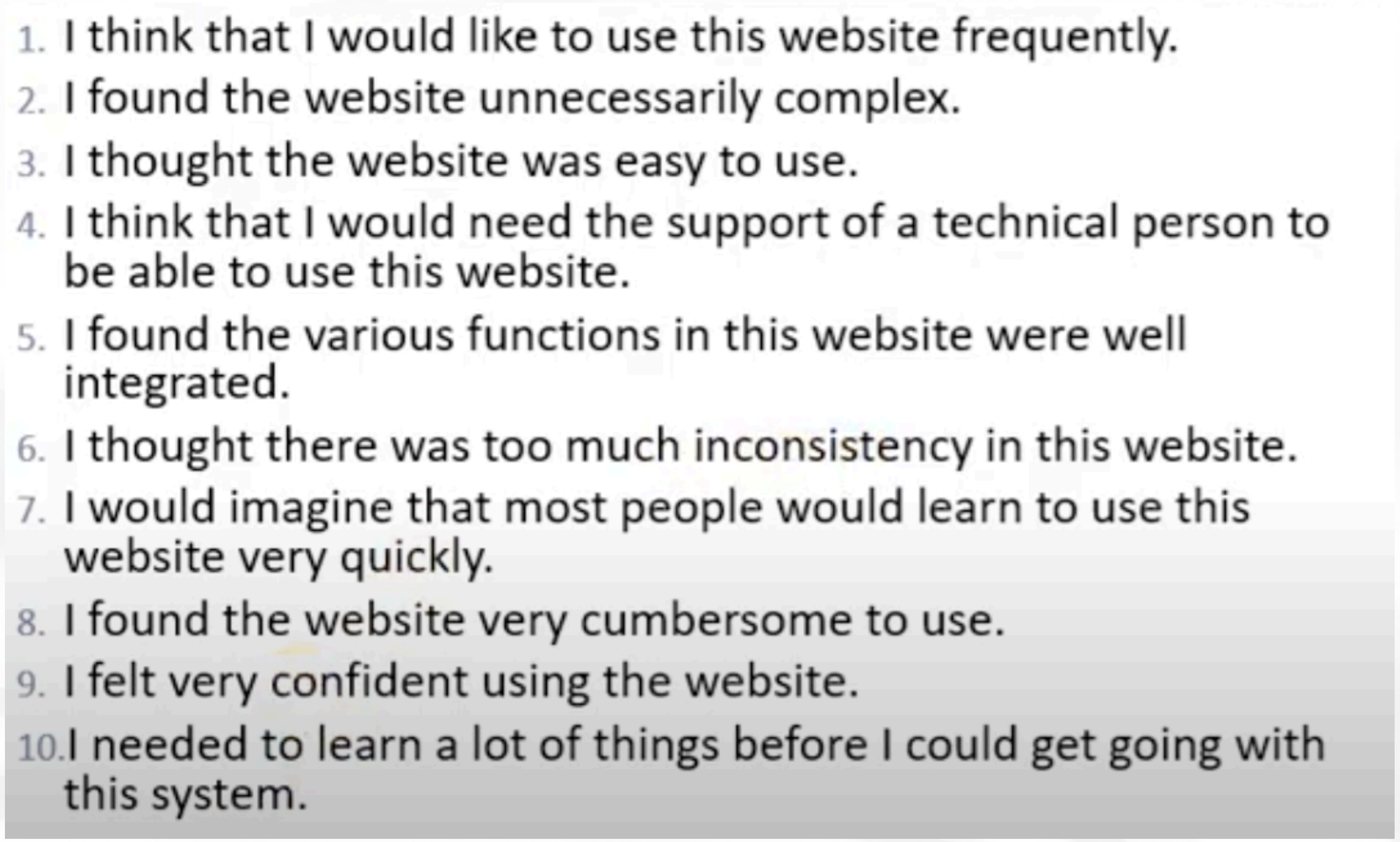
► The SUS consists of 10 questions with 5 options to choose from.



Questions	Response scale	Score	Dimensions
10	1 to 5 1 = strongly disagree 5 = strongly agree	Out of 100	2 Usability Learnability

Usability measurement scale

► The Template for SUS Questions are:



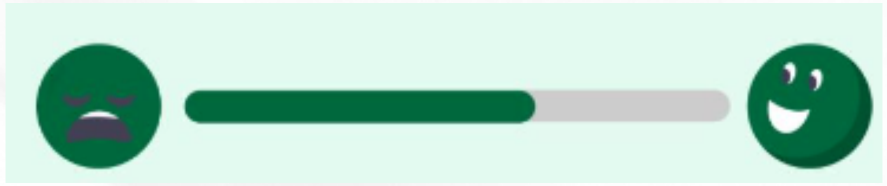
1. I think that I would like to use this website frequently.
2. I found the website unnecessarily complex.
3. I thought the website was easy to use.
4. I think that I would need the support of a technical person to be able to use this website.
5. I found the various functions in this website were well integrated.
6. I thought there was too much inconsistency in this website.
7. I would imagine that most people would learn to use this website very quickly.
8. I found the website very cumbersome to use.
9. I felt very confident using the website.
10. I needed to learn a lot of things before I could get going with this system.

Usability measurement scale

To compute SUS:

Step 1: Convert the scale into number for each of the 10 questions

- Strongly Disagree: 1 point
- Disagree: 2 points
- Neutral: 3 points
- Agree: 4 points
- Strongly Agree: 5 points



Usability measurement scale

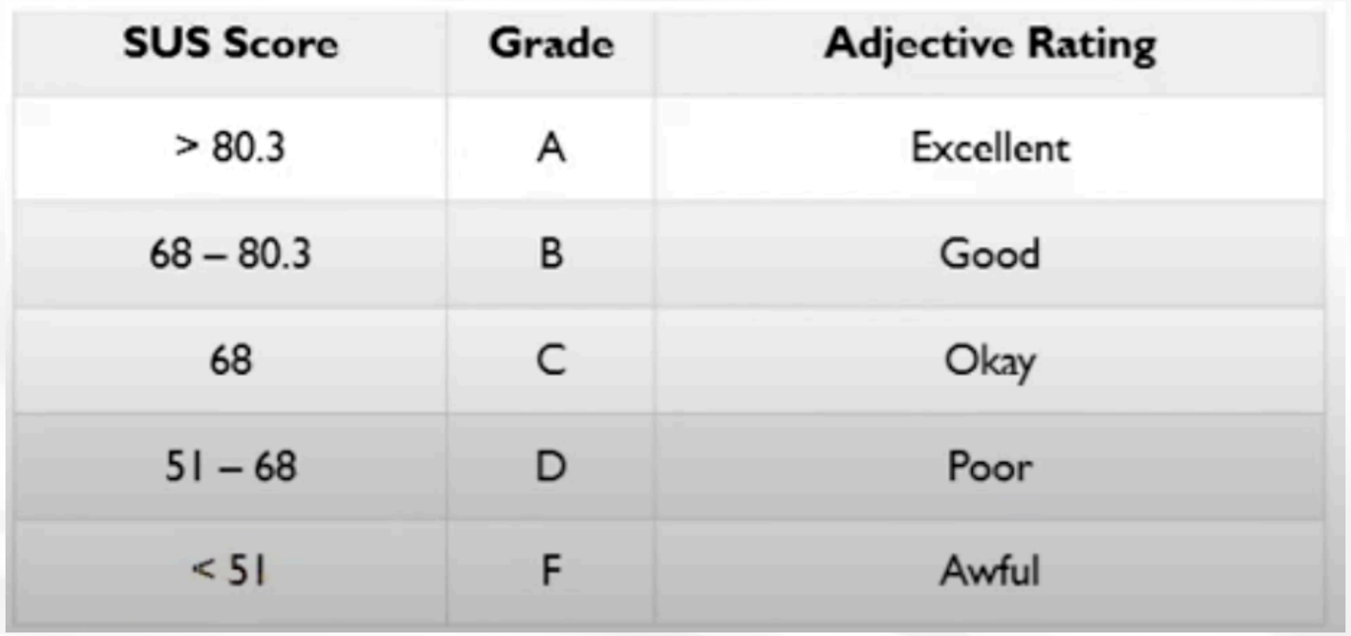
Step 2:

- For odd items: subtract one from the user response.
- For even–numbered items: subtract the user responses from 5
- This scales all values from 0 to 4 (with four being the most positive response).
- Add up the converted responses for each user and multiply that total by 2.5. This converts the range of possible values from 0 to 100 instead of from 0 to 40.

Usability measurement scale

Step 3: Evaluation

The average SUS score is 68. This simply means that a score of 68 will just put you at 50th percentile. Below is the general guideline on the interpretation of SUS score:



SUS Score	Grade	Adjective Rating
> 80.3	A	Excellent
68 – 80.3	B	Good
68	C	Okay
51 – 68	D	Poor
< 51	F	Awful

Product revision software quality factors

According to the McCall model of software quality factors, three quality fac–tors comprise the product revision category, These deal with requirements that affect the complete range of software maintenance activities:

- ❑ corrective maintenance,
- ❑ adaptive maintenance, and
- ❑ perfective maintenance

- Maintainability
- Flexibility
- Testability

1. Maintainability Requirements

The degree of effort needed to identify reasons (find the problem) for software failure and to correct failures and to verify the success of the corrections.

Deals with the modular structure of the software, internal program documentation, programmer manuals

Example specs: size of module <= 30 statements.

2. Flexibility Requirements –

- deals with resources to change (adopt) software to different types of customers that use the app perhaps a little differently;
- May also involve a little perfective maintenance to perhaps do a little better due to the customer's perhaps slightly more robust environment.

3. Testability Requirements –

- Are intermediate results of computations predefined to assist testing?
- Are log files created?
- Does the software diagnose itself prior to and perhaps during operations?

Product Transition software quality factors

According to McCall, three quality factors are included in the product transition category, a category that pertains to the adaptation of software to other environments and its interaction with other software systems.

- Portability
- Reusability
- Interoperability

Can I move the app to different hardware? Interface easily with different hardware / software systems; can I reuse major portions of the code with little modification to develop new apps?

Product Transition software quality factors

1. **Portability Requirements:** If the software must be ported to different environments (different hardware, operating systems, …) and still maintain an existing environment, then portability is a must.

2. **Reusability Requirements:** Are we able to reuse parts of the app for new applications?

   - Can save immense development costs due to errors found / tested.
   - Certainly higher quality software and development more quickly results.
   - Very big deal nowadays.

Product Transition software quality factors

3. **Interoperability Requirements:** Does the application need to interface with other existing systems

- Frequently these will be known ahead of time and plans can be made to provide for this requirement during design time.
  - Sometimes these systems can be quite different; different platforms, different databases, and more
- Also, industry or standard application structures in areas can be specified as requirements.

To Sum McCall Model

- **Product Operation**
  - Correctness : Does it do what customer wants ? (meeting specifications)
  - Efficiency : Does it quickly solve the intended problem ? (enough computing resources)
  - Integrity : Is it secure ? (access limitations to people)
  - Reliability : Does it do it accurately all of the time ? (successful performance)
  - Usability : Can I run it ? (efforts in learning/operating)
- **Product Revision**
  - Maintainability : Can it be fixed ? (fixing bugs and errors)
  - Flexibility : Can it be changed ? (modifying an operational program)
  - Testability : Can it be tested ? (ensuring performance)
- **Product Transition**
  - Portability : Can it be used on another machine ? (platform dependence)
  - Reusability : Can parts of it be reused ? (generic coding)
  - Interoperability : Can it interface with other system ? (coupling system)

Alternatives

- Some other SQA professionals have offered essentially renamed quality factors.
- One has offered 12 factors; another 15 factors.

- Totally five new factors were suggested
- **Evans** and Marciniak offer two ‘new’ ones:
  - Verifiability and Expandability
- **Willis** and Deutsch offer three ‘new’ ones.
  - Safety
  - Manageability, and
  - Survivability

Alternatives

Evans and Marciniak offer Verifiability and Expandability:

1. **Verifiability Requirements** addresses design and programming features that allow for efficient verification of design and programming;
   - This does not refer to outputs; rather, structure of code; design elements and their dependencies, coupling, cohesion; patterns...
   - Apply to modularity, simplicity, adherence to documentation and programming guidelines, etc.

Alternatives

Evans and Marciniak offer Verifiability and Expandability.

2. **Expandability Requirements** really refers to scalability and extensibility to provide more usability.
    - Essentially this is McCall's **flexibility**

Alternatives

Deutsch and Willis offer Safety, Manageability, and Survivability:

1. **Safety Requirements** address conditions that could bring the equipment or application down especially for controlling software, as in setting alarms or sounding warnings.
   - Especially important to process control / real time software such as that running conveyor belts or instrumentation for ordinance…

Alternatives

Deutsch and Willis offer Safety, Manageability, and Survivability

2. **Manageability Requirements** refer to tools primarily administrative to control versions, configurations and change management / tracking.
   - We must have tools to manage versions and various configurations that may vary from customer to customer.
3. **Survivability Requirements** refer to MTBF, or continuity of service, as well as MTTR (mean time to recover).
   - Appears to be quite similar to Reliability in McCall's model

McCall's factor model and Alternative models

No.	Software quality factor	McCall's classic model	Alternative factor models	
			Evans and Marciniak model	Deutsch and Willis model
1	Correctness	+	+	+
2	Reliability	+	+	+
3	Efficiency	+	+	+
4	Integrity	+	+	+
5	Usability	+	+	+
6	Maintainability	+	+	+
7	Flexibility	+	+	+
8	Testability	+		
9	Portability	+	+	+
10	Reusability	+	+	+
11	Interoperability	+	+	+
12	Verifiability		+	+
13	Expandability		+	+
14	Safety			+
15	Manageability			+
16	Survivability			+

Quality Criteria

Quality Criteria: The lower or second–level quality attributes that can be accessed either subjectively or objectively are called Quality Criteria. These attributes are internal. Each quality factor has many second–level quality attributes or quality criteria.

Software compliance with quality factors

- sub–factors have been defined for those quality factors that represent a wide range of attributes and aspects of software use.
- Software quality metrics are suggested for each of these sub–factors.

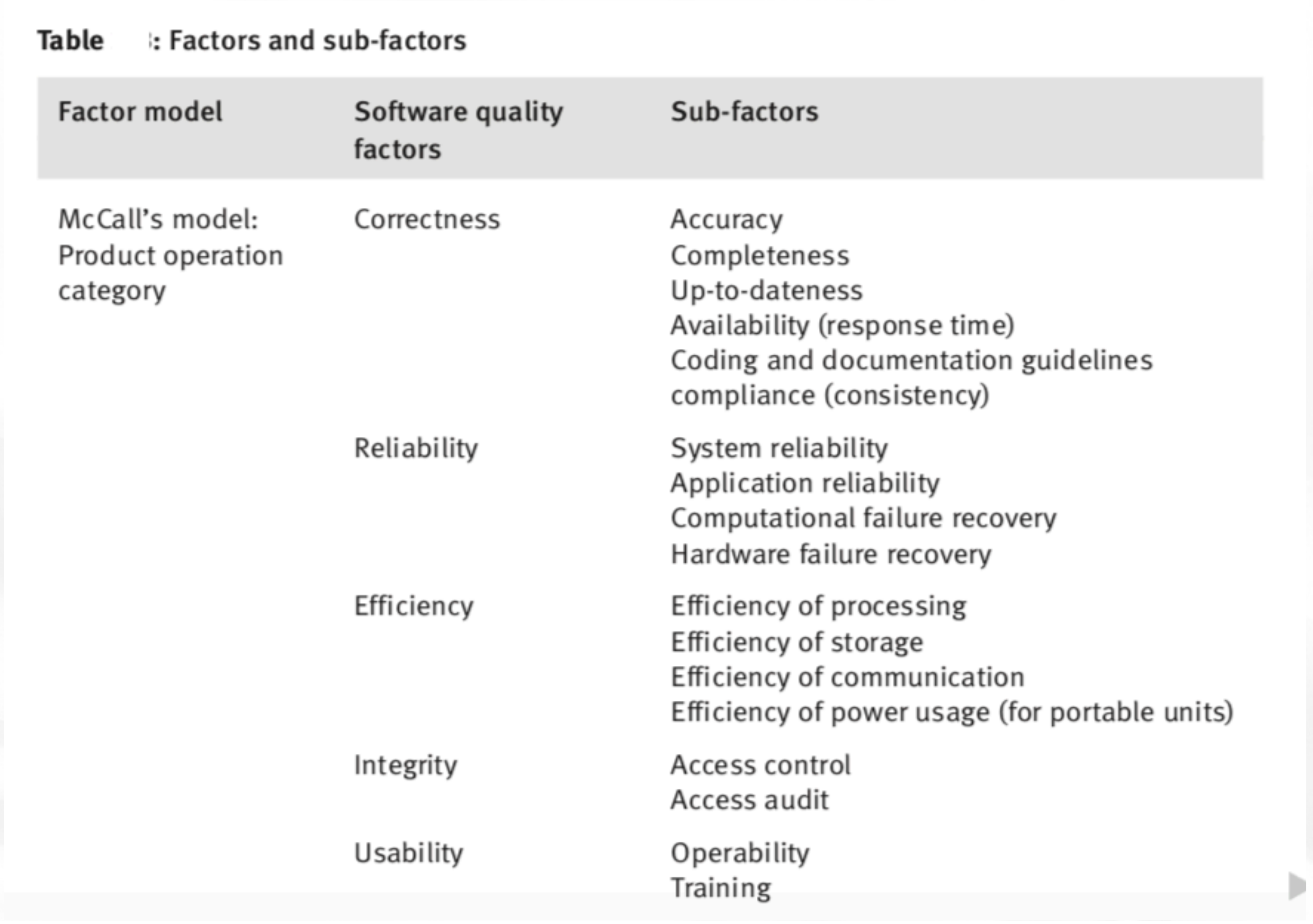
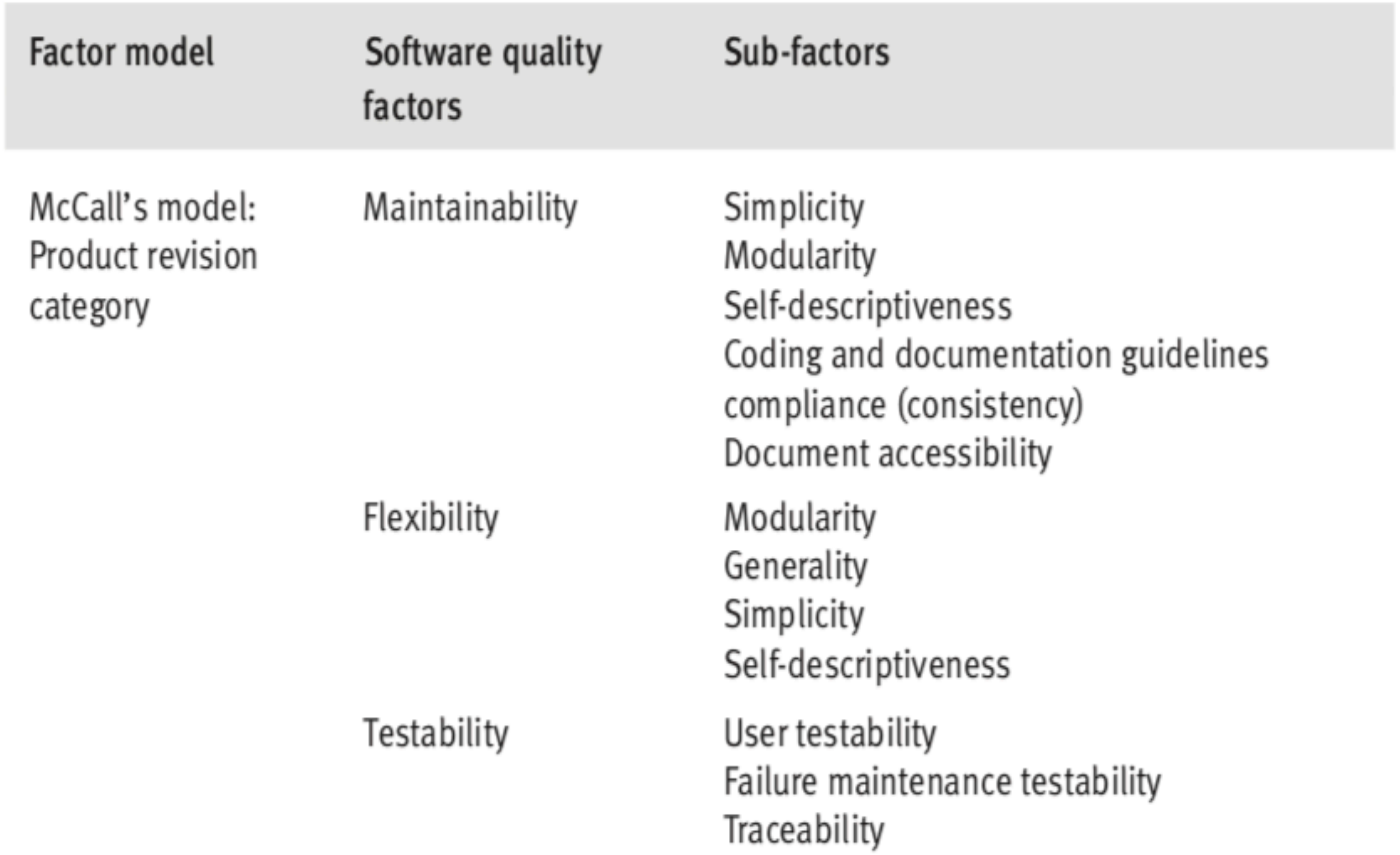


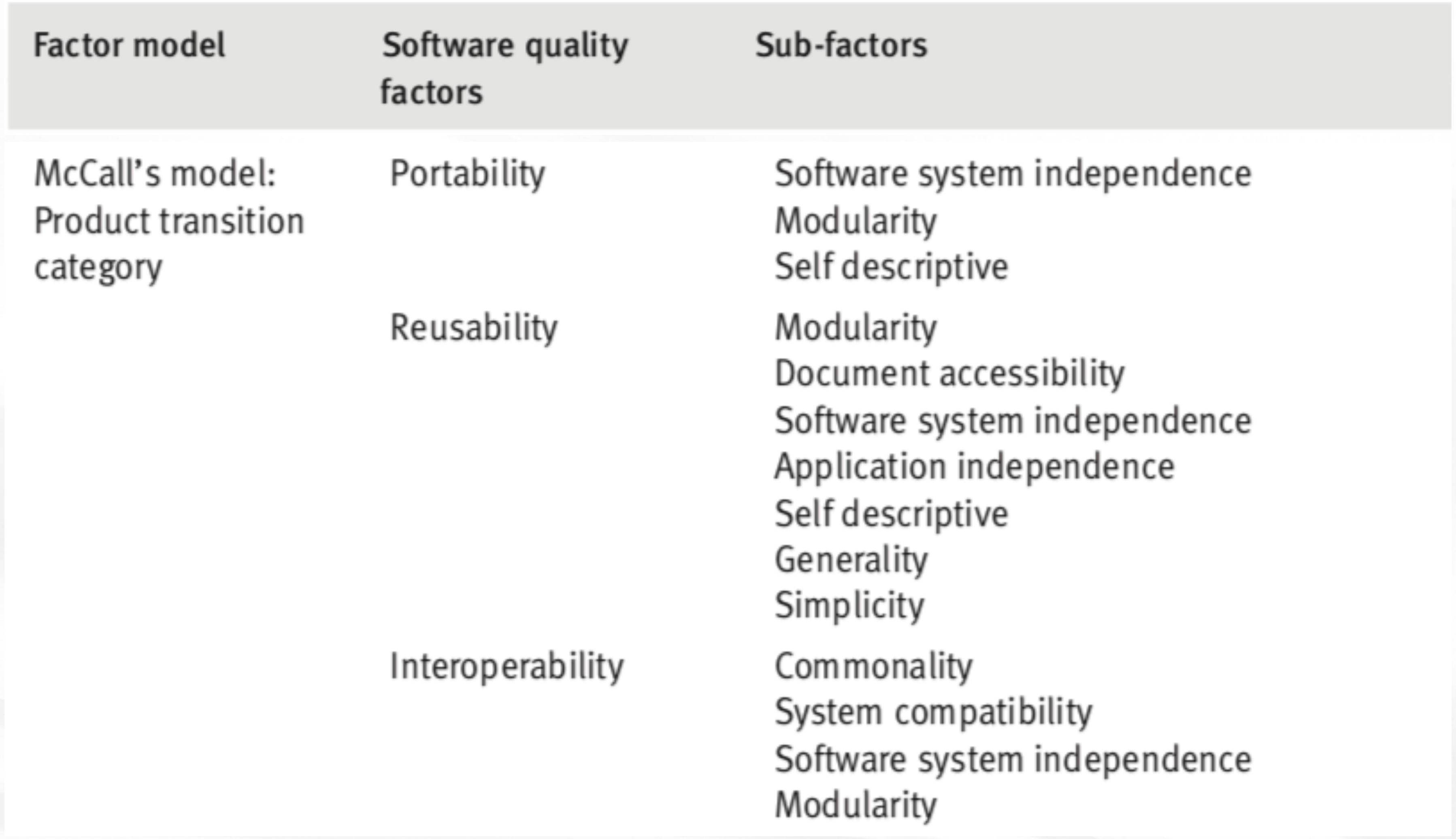
Table : Factors and sub-factors

Factor model	Software quality factors	Sub-factors
McCall's model: Product operation category	Correctness	Accuracy Completeness Up-to-dateness Availability (response time) Coding and documentation guidelines compliance (consistency)
	Reliability	System reliability Application reliability Computational failure recovery Hardware failure recovery
	Efficiency	Efficiency of processing Efficiency of storage Efficiency of communication Efficiency of power usage (for portable units)
	Integrity	Access control Access audit
	Usability	Operability Training

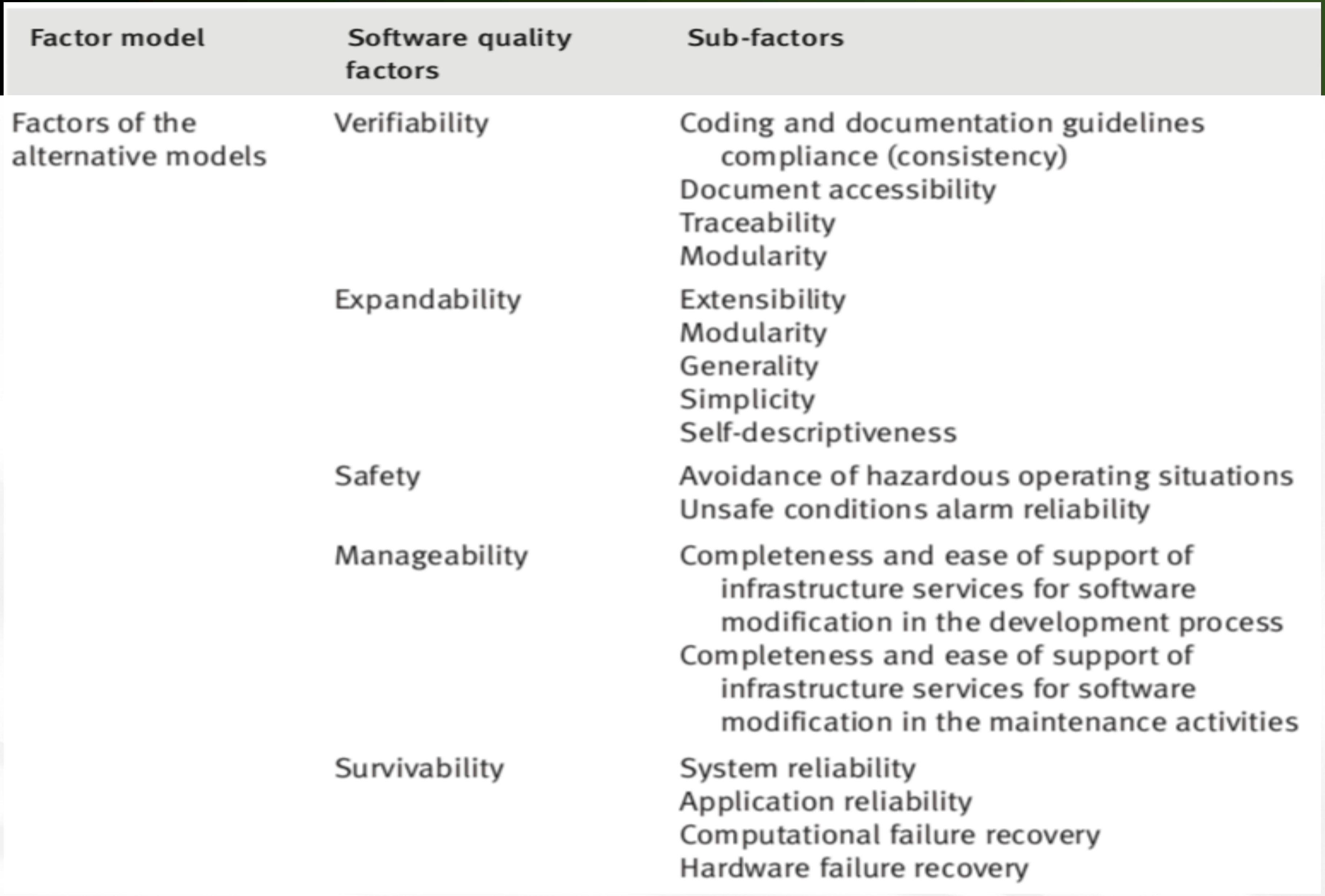
Software compliance with quality factors



Factor model	Software quality factors	Sub-factors
McCall’s model: Product revision category	Maintainability	Simplicity Modularity Self-descriptiveness Coding and documentation guidelines compliance (consistency) Document accessibility
	Flexibility	Modularity Generality Simplicity Self-descriptiveness
	Testability	User testability Failure maintenance testability Traceability



Factor model	Software quality factors	Sub-factors
McCall's model: Product transition category	Portability	Software system independence Modularity Self descriptive
	Reusability	Modularity Document accessibility Software system independence Application independence Self descriptive Generality Simplicity
	Interoperability	Commonality System compatibility Software system independence Modularity

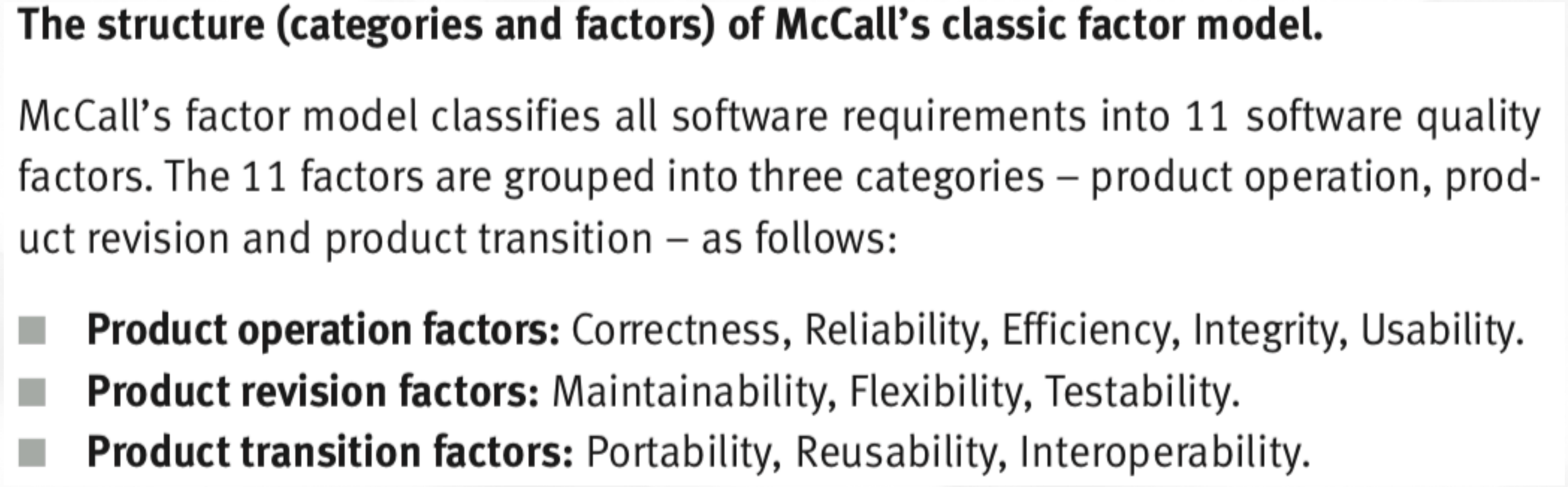


Factor model	Software quality factors	Sub-factors
Factors of the alternative models	Verifiability	Coding and documentation guidelines compliance (consistency) Document accessibility Traceability Modularity
	Expandability	Extensibility Modularity Generality Simplicity Self-descriptiveness
	Safety	Avoidance of hazardous operating situations Unsafe conditions alarm reliability
	Manageability	Completeness and ease of support of infrastructure services for software modification in the development process Completeness and ease of support of infrastructure services for software modification in the maintenance activities
	Survivability	System reliability Application reliability Computational failure recovery Hardware failure recovery

Relationship Between Quality Factors and Quality Criteria

- Each quality factor is positively influenced by a set of quality criteria, and the same quality criterion impacts a number of quality factors.
    - Example: Simplicity impacts reliability, usability, and testability.
- If an effort is made to improve one quality factor, another quality factor may be degraded.
    - Portable code may be less efficient.
- Some quality factors positively impact others.
    - An effort to improve the correctness of a system will increase its reliability.

Summary

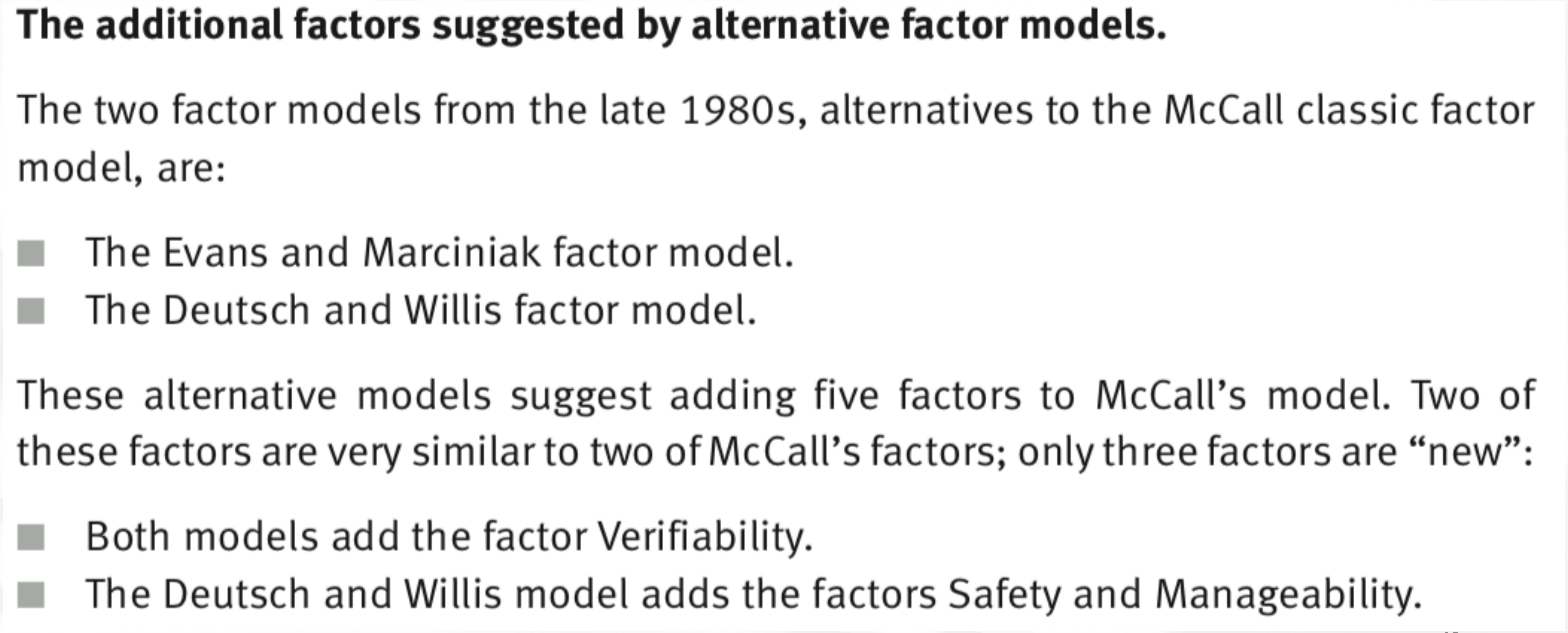


The structure (categories and factors) of McCall's classic factor model.

McCall's factor model classifies all software requirements into 11 software quality factors. The 11 factors are grouped into three categories – product operation, product revision and product transition – as follows:

- **Product operation factors:** Correctness, Reliability, Efficiency, Integrity, Usability.
- **Product revision factors:** Maintainability, Flexibility, Testability.
- **Product transition factors:** Portability, Reusability, Interoperability.

Summary



The additional factors suggested by alternative factor models.

The two factor models from the late 1980s, alternatives to the McCall classic factor model, are:

- The Evans and Marciniak factor model.
- The Deutsch and Willis factor model.

These alternative models suggest adding five factors to McCall’s model. Two of these factors are very similar to two of McCall’s factors; only three factors are “new”:

- Both models add the factor Verifiability.
- The Deutsch and Willis model adds the factors Safety and Manageability.