This image cannot currently be displayed.

Chapter 9

Software Reliability

Declaration

- These slides are made for UIT, BU students only. I am not holding any copy write of it as I had collected these study materials from different books and websites etc. I have not mentioned those to avoid complexity.

Basic Concepts

- There are three phases in the life of any hardware component i.e., burn-in, useful life & wear-out.
- In **burn-in phase,** failure rate is quite high initially, and it starts decreasing gradually as the time progresses.
- During **useful life period**, failure rate is approximately constant.
- Failure rate increase in **wear-out phase** due to wearing out/aging of components. The best period is useful life period
- The shape of this curve is like a "bath tub" and that is why it is known as bath tub curve. The "bath tub curve" is given in Figure7.1.

Basic Concepts: Curve

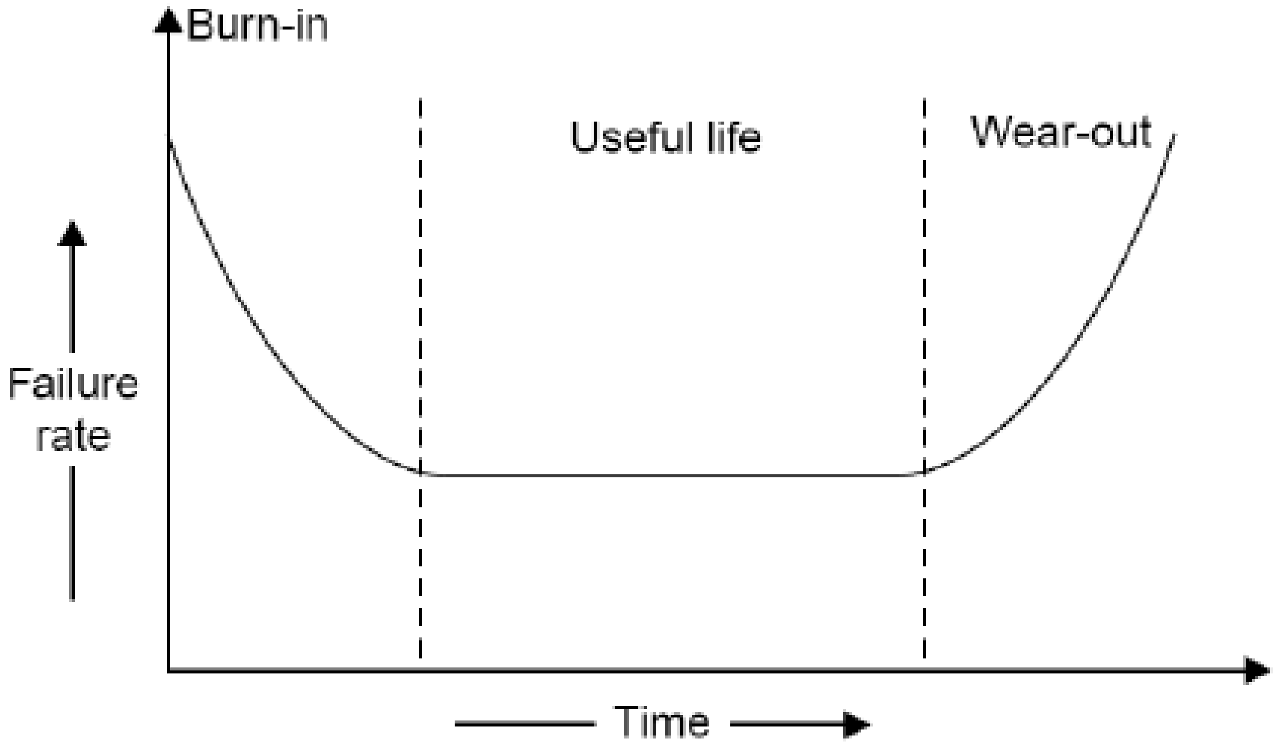


Fig. 7.1: Bath tub curve of hardware reliability.

Basic Concepts: Curve

We do not have wear out phase in software. The expected curve for software is given in fig. 7.2.

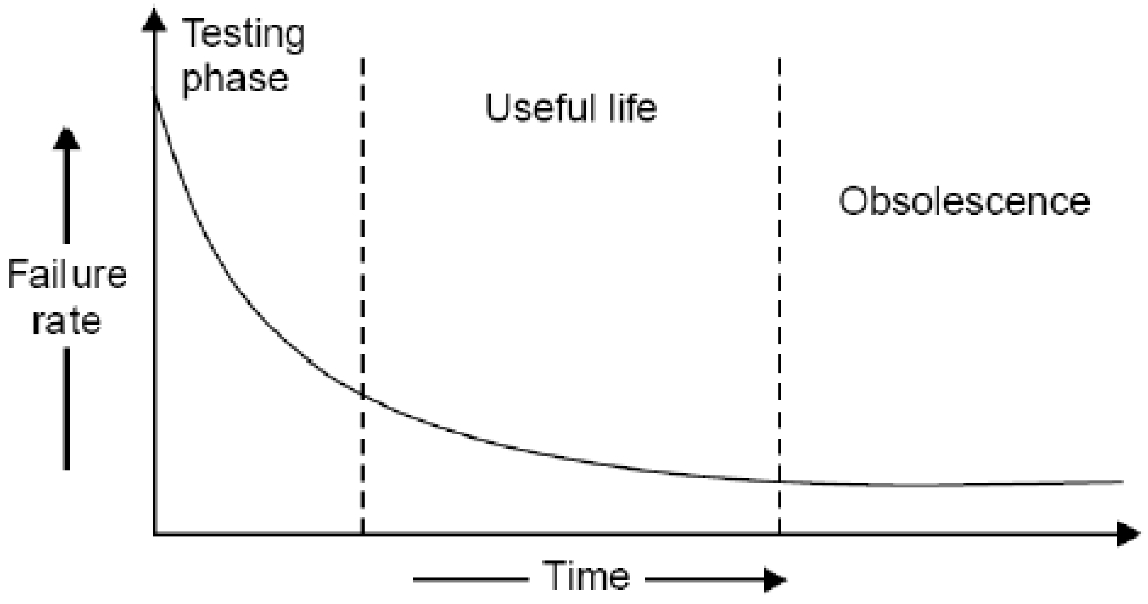


Fig. 7.2: Software reliability curve (failure rate versus time)

Factors for retirement of any software

- Software may be retired only if it becomes obsolete. Some of contributing factors are given below:
  - change in environment
  - change in infrastructure/technology
  - major change in requirements
  - increase in complexity
  - extremely difficult to maintain
  - Software Reliability
  - deterioration in structure of the code
  - slow execution speed
  - poor graphical user interfaces

Software Reliability: Definition

- As per IEEE standard: “Software reliability is defined as the ability of a system or component to perform its required functions under stated conditions for a specified period of time”.

Failures and Faults

- A fault is the defect in the program that, when executed under particular conditions, causes a failure.
- The execution time for a program is the time that is actually spent by a processor in executing the instructions of that program.
- The second kind of time is calendar time. It is the familiar time that we normally experience.

Some Terminologies

- There are
  1. time of failure,
  2. time interval between failures,
  3. cumulative failure experienced up to a given time,
  4. failures experienced in a time interval.

Time based failure

Failure Number	Failure Time (sec)	Failure Interval (sec)
1	8	8
2	18	10
3	25	7
4	36	11
5	45	9
6	57	12
7	71	14
8	86	15
9	104	18
10	124	20
11	143	19
12	169	26
13	197	28
14	222	25
15	250	28

Table 7.1: Time based failure specification

Time based failure

Time (sec)	Cumulative Failures	Failure in interval (30 sec)
30	3	3
60	6	3
90	8	2
120	9	1
150	11	2
180	12	1
210	13	1
240	14	1

Table 7.2: Failure based failure specification

Time based failure

Value of random variable (failures in time period)	Probability	
	Elapsed time t_A = 1 hr	Elapsed time t_B = 5 hr
0	0.10	0.01
1	0.18	0.02
2	0.22	0.03
3	0.16	0.04
4	0.11	0.05
5	0.08	0.07
6	0.05	0.09
7	0.04	0.12
8	0.03	0.16
9	0.02	0.13

Table 7.3: Probability distribution at times t_A and t_B

Time based failure

Value of random variable (failures in time period)	Probability	
	Elapsed time t_A = 1 hr	Elapsed time t_B = 5 hr
10	0.01	0.10
11	0	0.07
12	0	0.05
13	0	0.03
14	0	0.02
15	0	0.01
Mean failures	3.04	7.77

Table 7.3: Probability distribution at times t_A and t_B

Time based failure

- This table shows the **probability distributions** for a random variable (number of failures) during two different time intervals: t_A=1 hour and t_B=5 hours. Each row corresponds to the probability of observing a specific number of failures (from 0 to 15) within the respective time periods.
- **Random Variable (Failures):**
  - Represents the number of system failures or events within a specified time period.

What is risk ?

- Tomorrow's problems are today's risks.
- *"Risk is a problem that may cause some loss or threaten the success of the project, but which has not happened yet".*

Risk Management

- We Software developers are extremely optimists.
- We assume, everything will go exactly as planned.
- Risk management is required to reduce this surprise factor
- Dealing with concern before it becomes a crisis.
- Quantify probability of failure & consequences of failure.
- Risk management is the process of identifying addressing and eliminating these problems before they can damage the project.

Typical Software Risk

- Capers Jones has identified the top five risk factors that threaten projects in different applications.

1. Dependencies on outside agencies or factors.
2. Requirement issues
3. Management issues
4. Lack of knowledge
5. Other risk categories

Dependencies on outside agencies or factors

- Availability of trained, experienced persons
- Inter group dependencies
- Customer-Furnished items or information
- Internal & external subcontractor relationships

Requirement issues

$$\begin{tikzcd}
\text{Uncertain requirements} \arrow[d] \\
\text{Wrong product}
\end{tikzcd}$$

or

Right product badly

Either situation results in unpleasant surprises and unhappy customers.

Requirement issues

- Lack of clear product vision
- Unprioritized requirements
- Lack of agreement on product requirements
- New market with uncertain needs
- Rapidly changing requirements
- Inadequate impact analysis of requirements changes

Management Issues

- Project managers usually write the risk management plans, and most people do not wish to air their weaknesses in public.
  - Inadequate planning
  - Inadequate visibility into actual project status
  - Unclear project ownership and decision making
  - Staff personality conflicts
  - Unrealistic expectation
  - Poor communication

Lack of knowledge

- Inadequate training
- Poor understanding of methods, tools, and techniques
- Inadequate application domain experience
- New technologies
- Ineffective, poorly documented or neglected processes

Other risk categories

- Unavailability of adequate testing facilities
- Turnover of essential personnel
- Unachievable performance requirements
- Technical approaches that may not work

Risk Management Activities

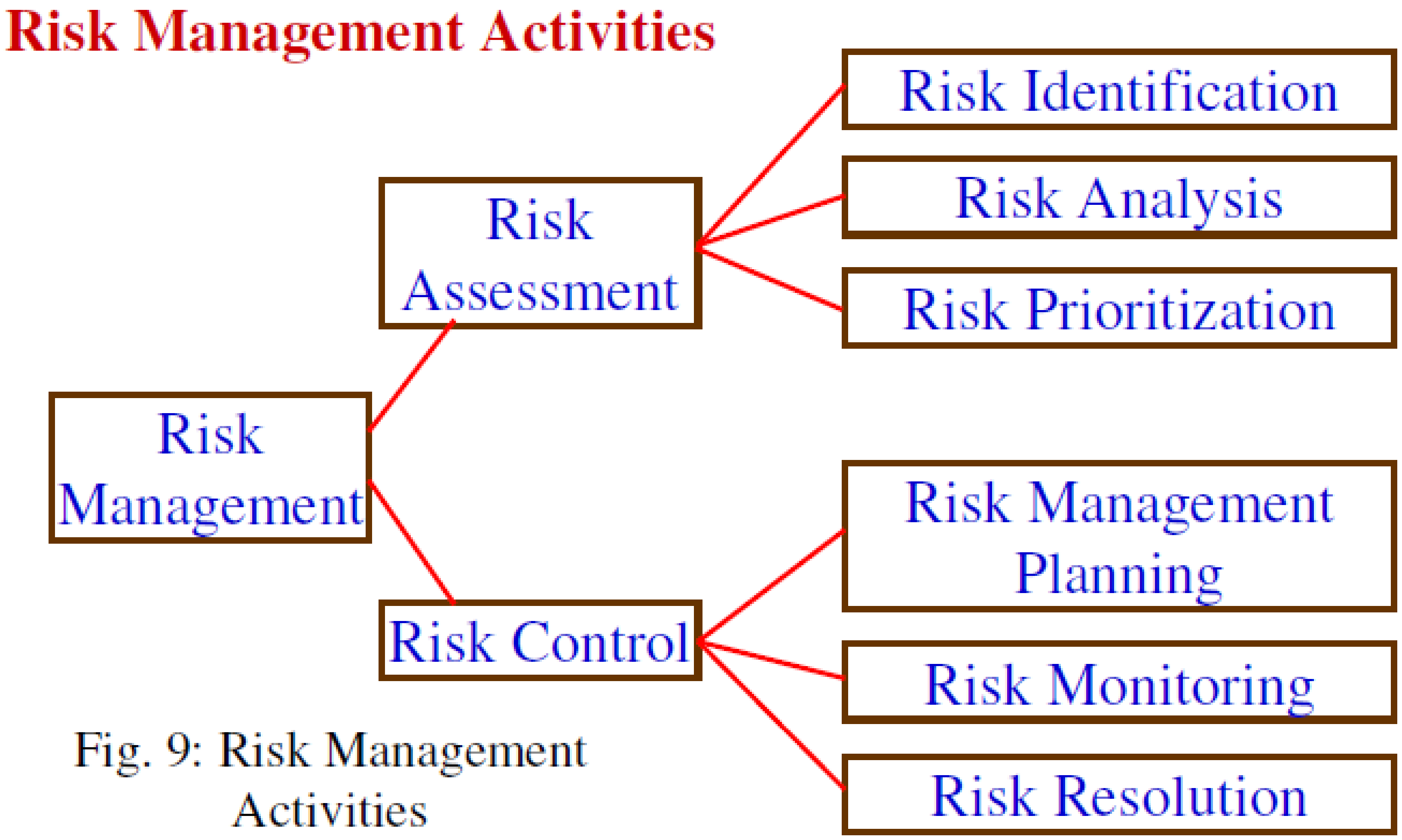


Fig. 9: Risk Management Activities

Risk Assessment

Risk analysis involves examining how project outcomes might change with modification of risk input variables.

Risk prioritization focus for severe risks.

Risk exposure: It is the product of the probability of incurring a loss due to the risk and the potential magnitude of that loss.

Risk avoidance

- Another way of handling risk is the risk avoidance. Do not do the risky things! We may avoid risks by not undertaking certain projects, or by relying on proven rather than cutting edge technologies.

Risk Control

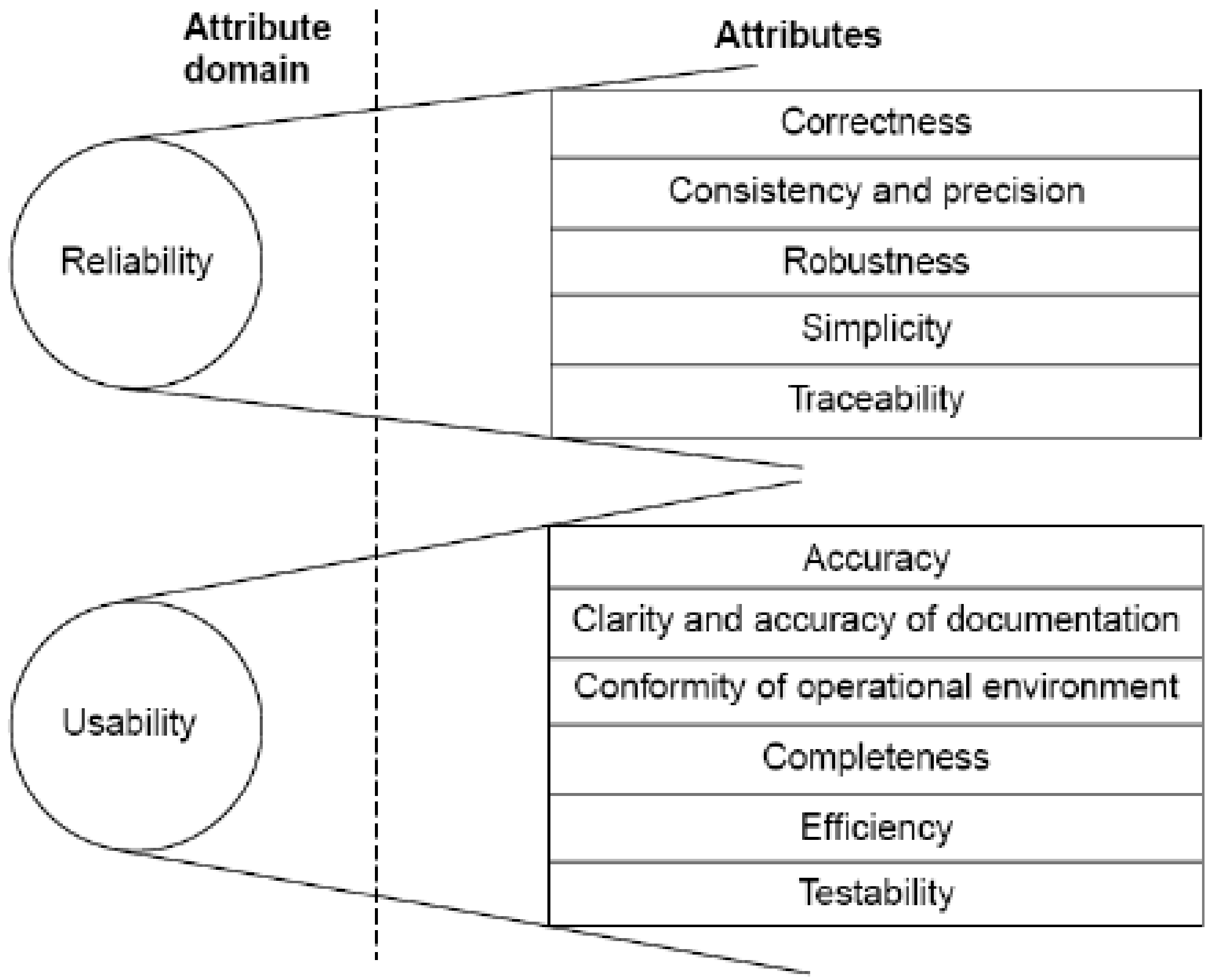
- Risk Management Planning produces a plan for dealing with each significant risks.
- Record decision in the plan.
- Risk resolution is the execution of the plans of dealing with each risk.

Software Quality

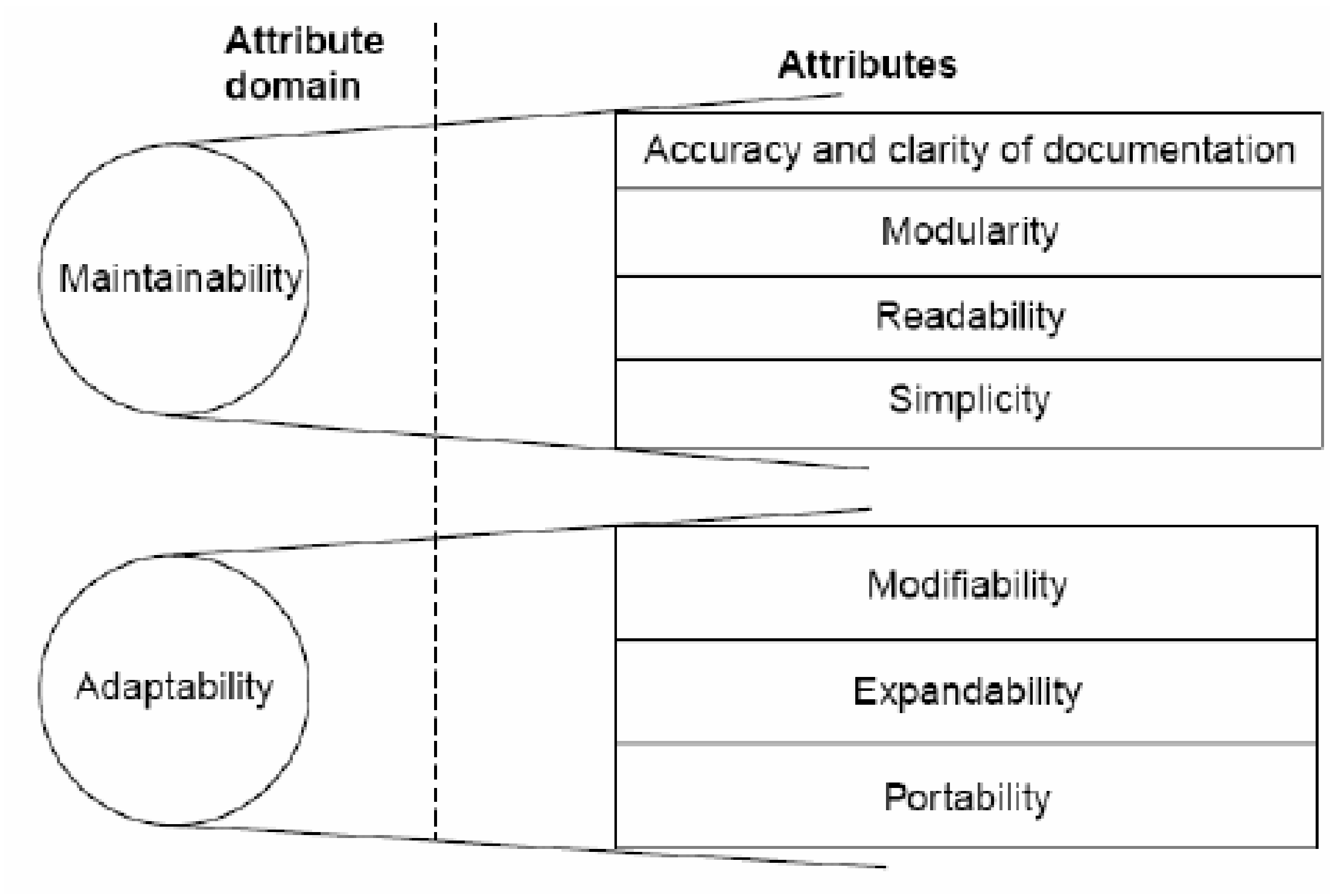
Different people understand different meanings of quality like:

- conformance to requirements
- fitness for the purpose
- level of satisfaction

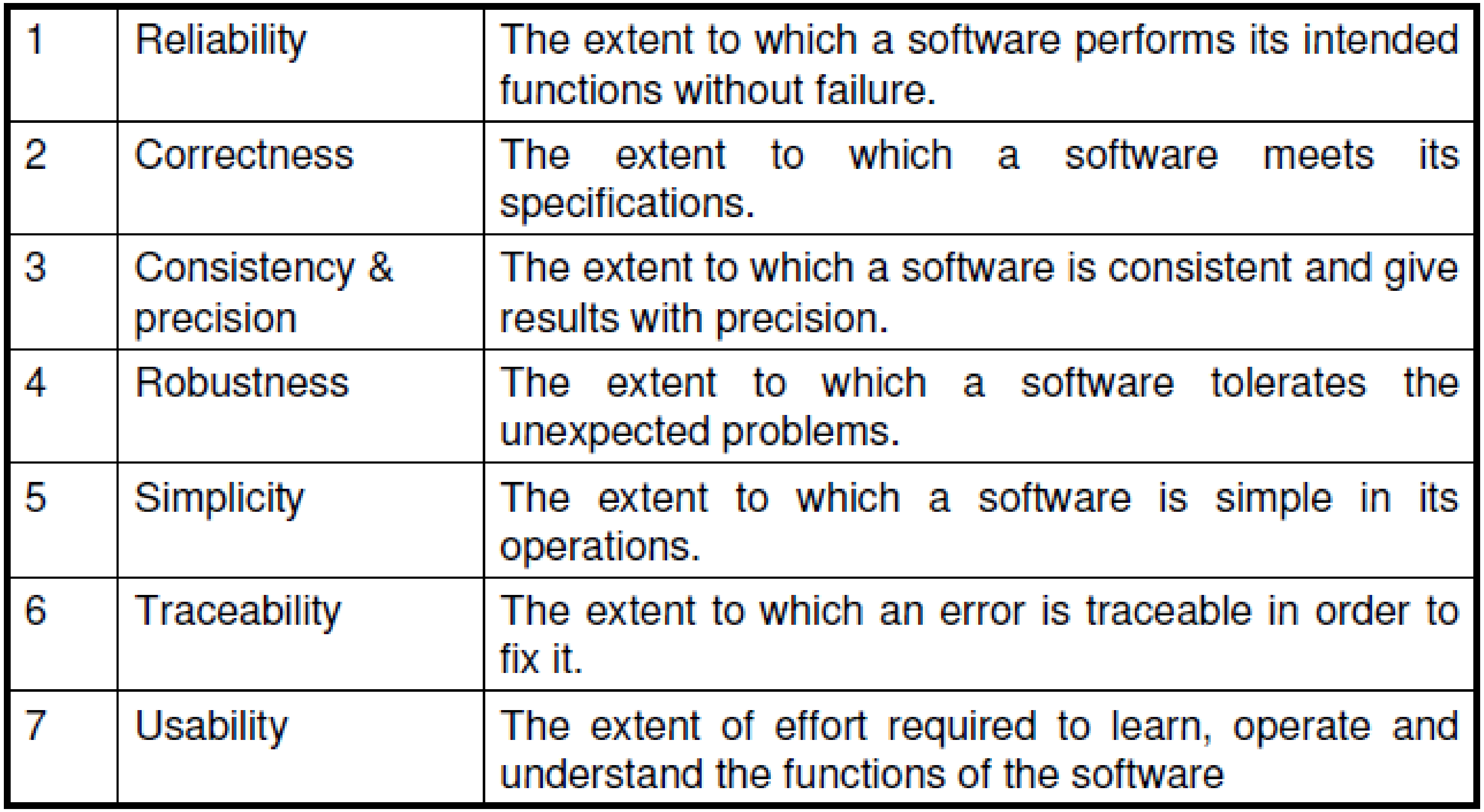
Software Quality



Software Quality



Software Quality



1	Reliability	The extent to which a software performs its intended functions without failure.
2	Correctness	The extent to which a software meets its specifications.
3	Consistency & precision	The extent to which a software is consistent and give results with precision.
4	Robustness	The extent to which a software tolerates the unexpected problems.
5	Simplicity	The extent to which a software is simple in its operations.
6	Traceability	The extent to which an error is traceable in order to fix it.
7	Usability	The extent of effort required to learn, operate and understand the functions of the software

Software Quality

8	Accuracy	Meeting specifications with precision.
9	Clarity & Accuracy of documentation	The extent to which documents are clearly & accurately written.
10	Conformity of operational environment	The extent to which a software is in conformity of operational environment.
11	Completeness	The extent to which a software has specified functions.
12	Efficiency	The amount of computing resources and code required by software to perform a function.
13	Testability	The effort required to test a software to ensure that it performs its intended functions.
14	Maintainability	The effort required to locate and fix an error during maintenance phase.

Software Quality

15	Modularity	It is the extent of ease to implement, test, debug and maintain the software.
16	Readability	The extent to which a software is readable in order to understand.
17	Adaptability	The extent to which a software is adaptable to new platforms & technologies.
18	Modifiability	The effort required to modify a software during maintenance phase.
19	Expandability	The extent to which a software is expandable without undesirable side effects.
20	Portability	The effort required to transfer a program from one platform to another platform.

McCall Software Quality Model

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

McCall Software Quality Model

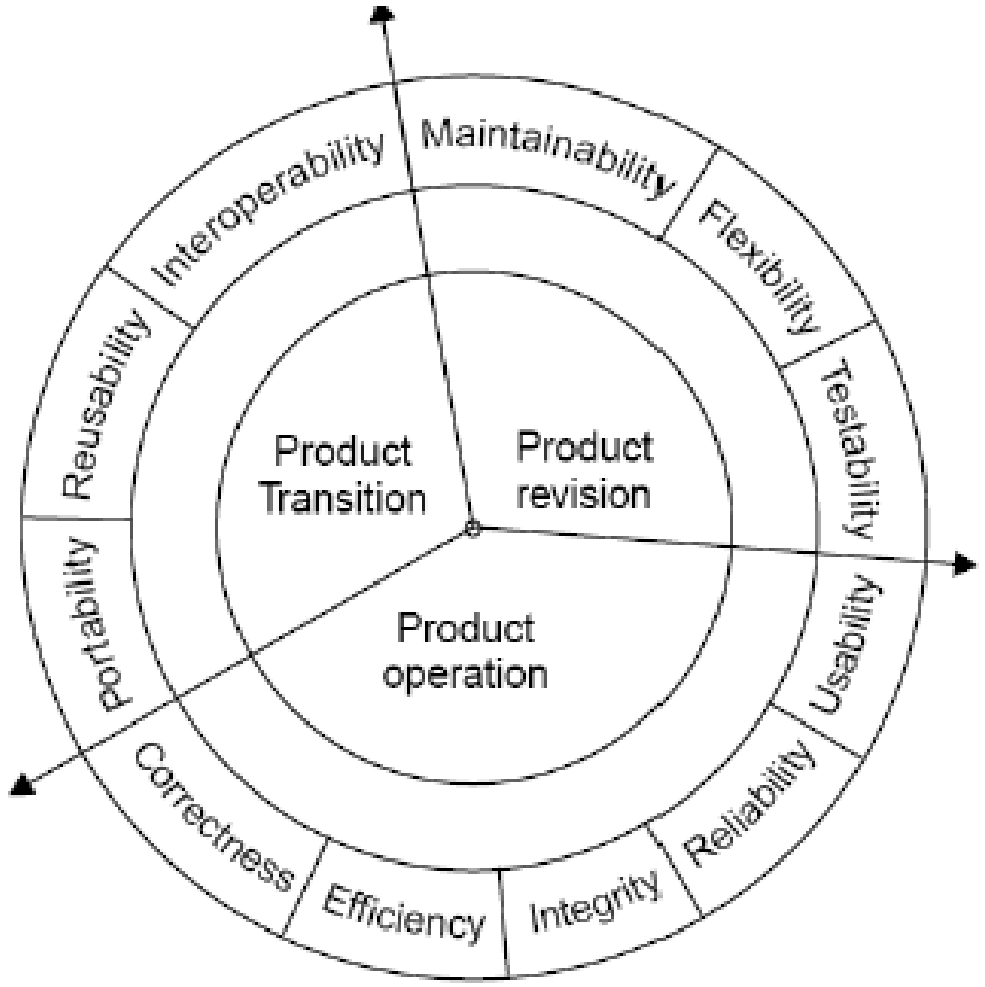


Fig 7.9: Software quality factors

McCall Software Quality Model

Most of the quality factors are explained in table 7.4. The remaining factors are given in table 7.5.

Sr.No.	Quality Factors	Purpose
1	Integrity	The extent to which access to software or data by the unauthorized persons can be controlled.
2	Flexibility	The effort required to modify an operational program.
3	Reusability	The extent to which a program can be reused in other applications.
4	Interoperability	The effort required to couple one system with another.

Table 7.5: Remaining quality factors (other are in table 7.4)

Boehm Software Quality model

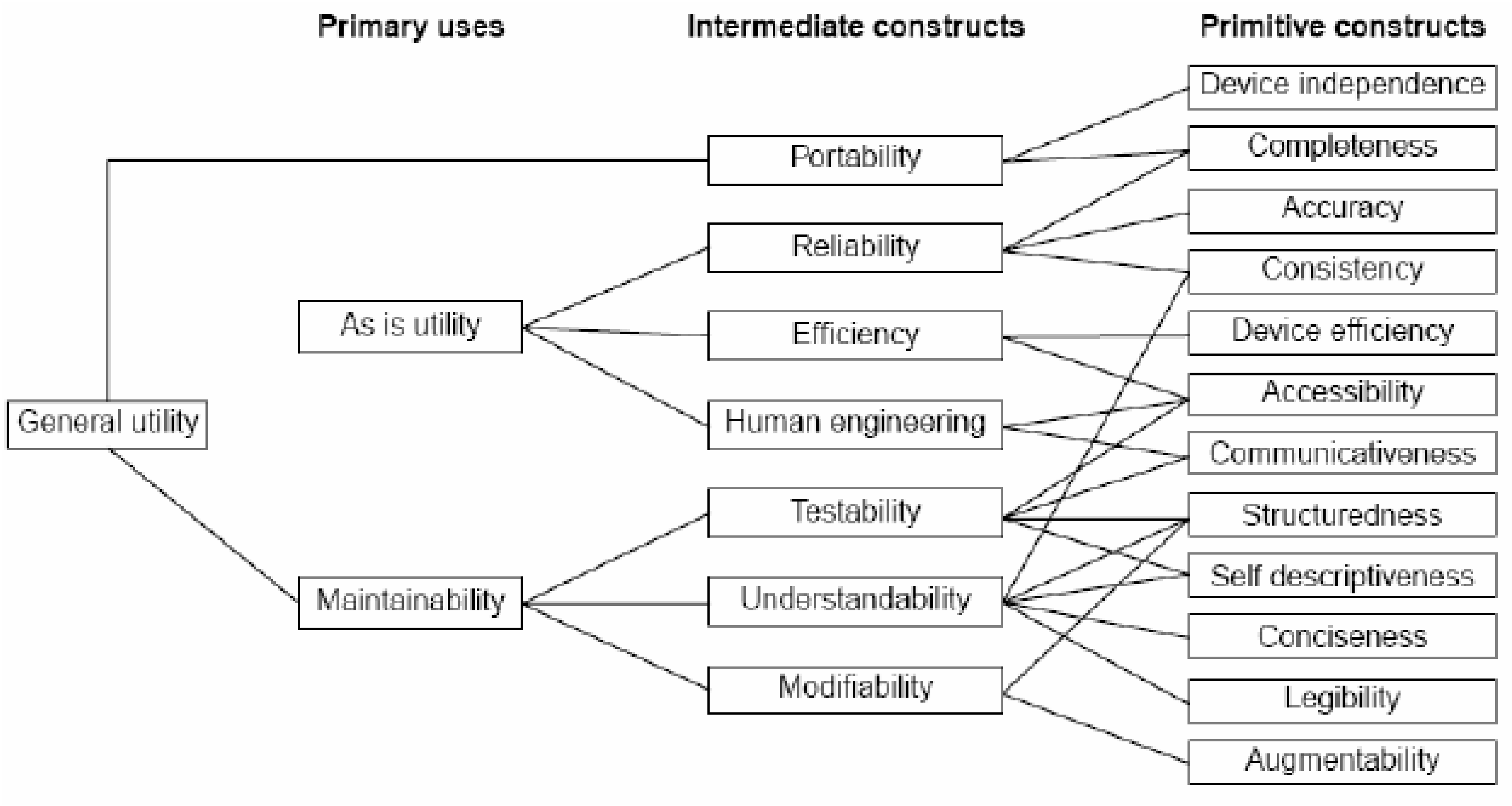


Fig.7.11: The Boehm software quality model

Boehm Software Quality model

- Though Boehm’s and McCall’s models might appear very similar, the difference is that McCall’s model primarily focuses on the precise measurement of the high-level characteristics “As-is utility”, whereas Boehm’s quality model is based on a wider range of characteristics with an extended and detailed focus on primarily maintainability.

ISO 9000

- ISO 9000 is defined as a set of international standards on quality management and quality assurance developed to help companies effectively document the quality system elements needed to maintain an efficient quality system. They are not specific to any one industry and can be applied to organizations of any size.
- ISO 9000 can help a company satisfy its customers, meet regulatory requirements, and achieve continual improvement. It should be considered to be a first step or the base level of a quality system.

ISO 9126

- ISO 9126 is an international standard for the evaluation of software.
- The ISO 9126-1 software quality model identifies **6 main quality characteristics**, namely:
  - Functionality
  - Reliability
  - Usability
  - Efficiency
  - Maintainability
  - Portability

ISO 9126

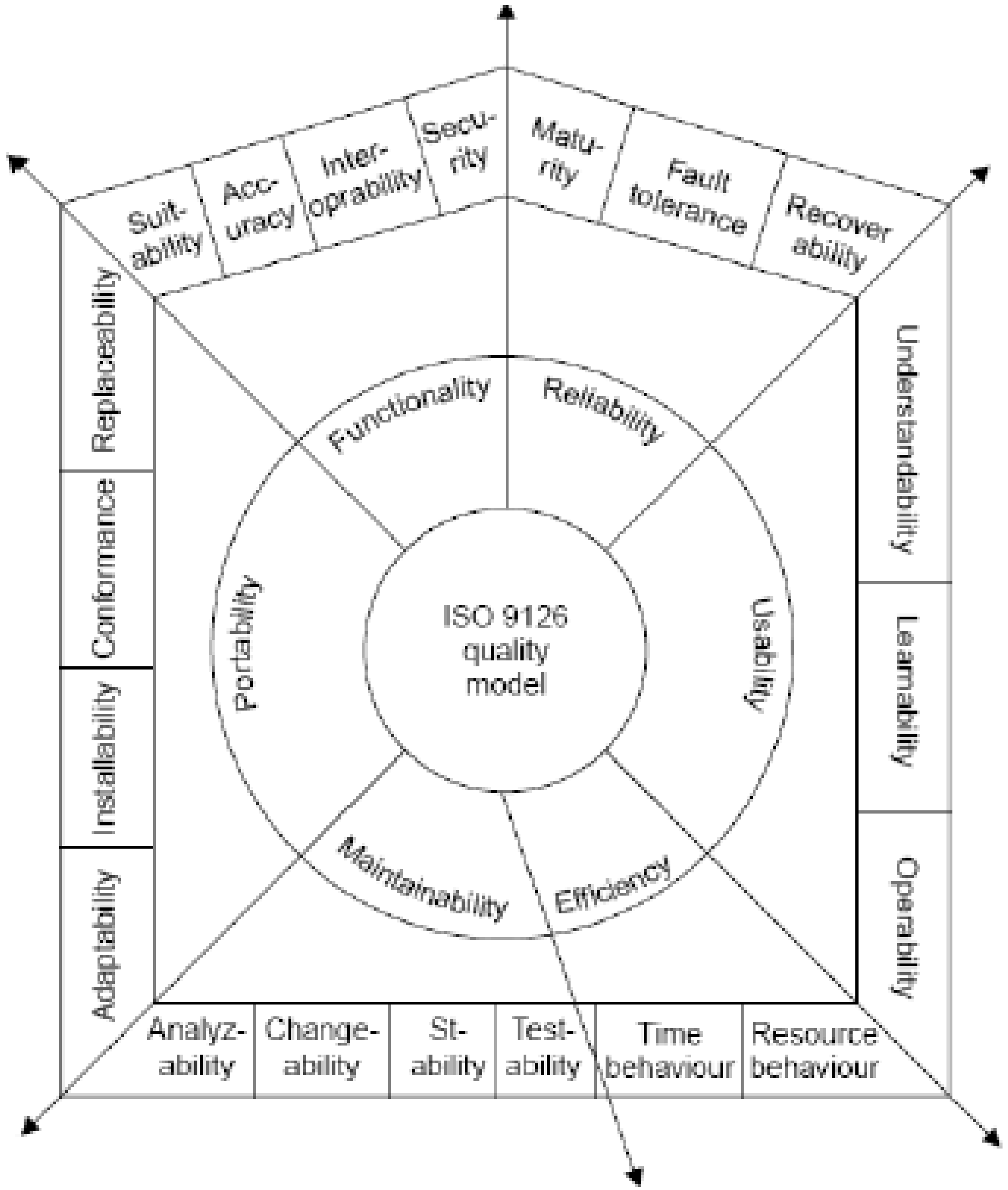


Fig.7.12: ISO 9126 quality model

This image cannot currently be displayed.

End of Chapter 9

Questions?