Measuring External Product Attributes

QUALITY

- The principal objective of software engineering is to improve the quality of software products
- Software quality is difficult to judge because it depends on the beholder.
- However, from a measurement perspective, we must be able to define quality in terms of specific software product attributes of interest to the user

QUALITY ATTRIBUTES

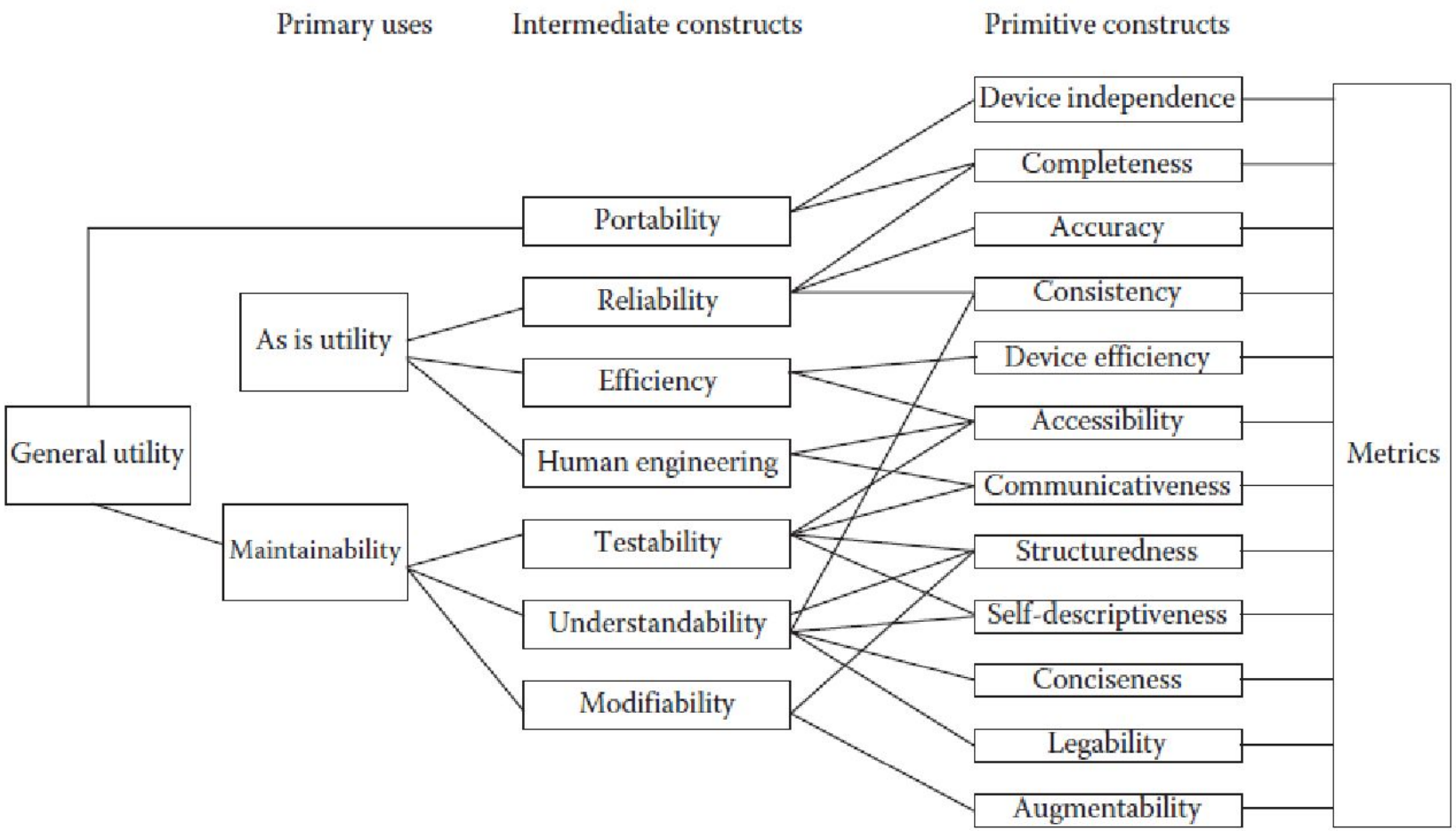
- Maintainability
- Usability
- Security
- *Reliability*

These key attributes, called *quality factors*, are normally high-level external attributes

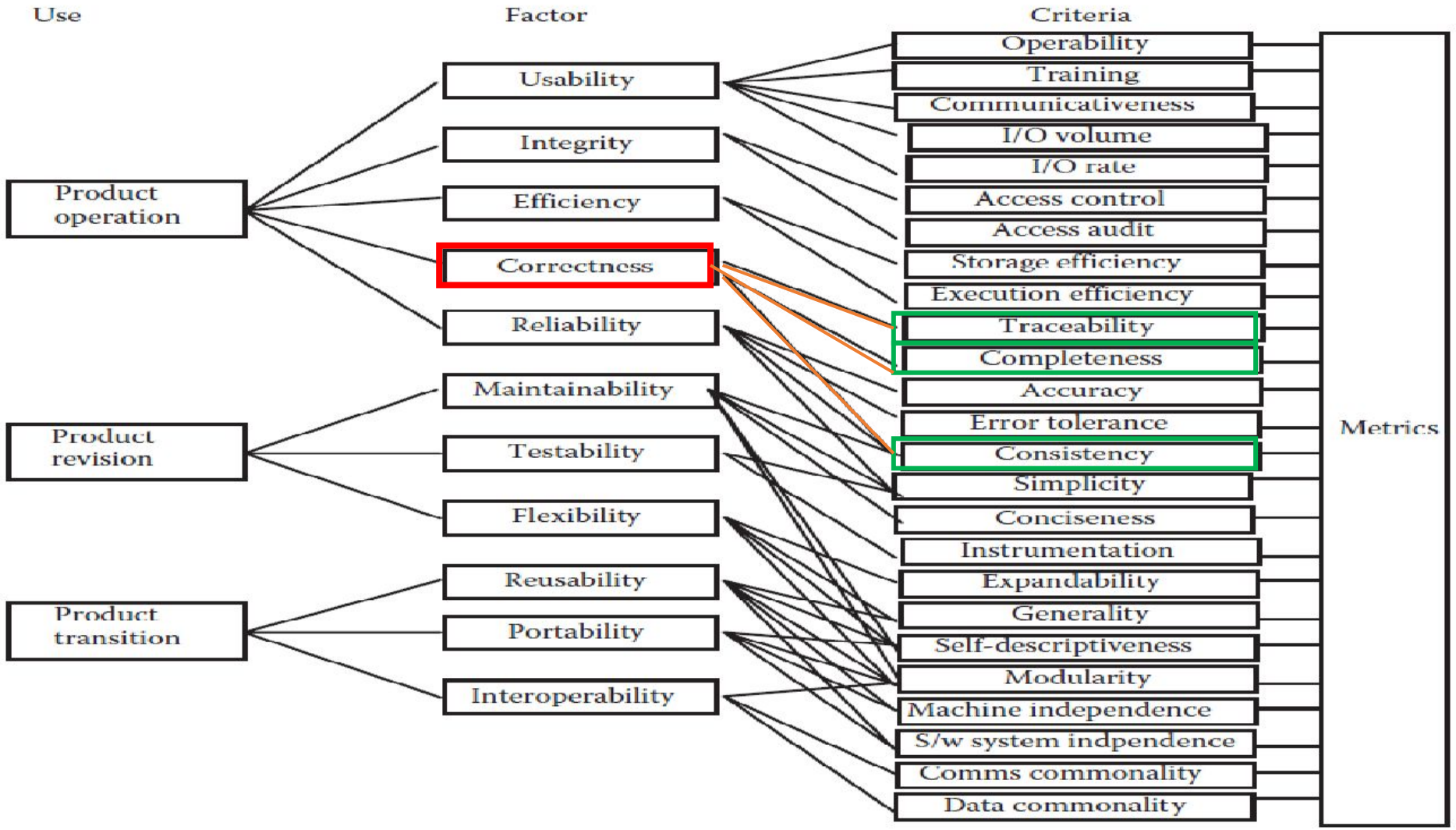
MODELING SOFTWARE QUALITY

- Early Models
  - Boehm software quality model
  - McCall software quality model
- These models focus on the final product and identify key attributes of quality from the user's perspective.
- Each of the models assumes that the quality factors are still at too high a level to be meaningful or to be measured directly. Hence, they are further decomposed into lower-level attributes called *quality criteria* or *quality subfactors*.

BOEHM SOFTWARE QUALITY MODEL



McCall SOFTWARE QUALITY MODEL



TWO DIFFERENT WAYS

- The *“fixed model”* approach
  - Assumes all the important quality factors needed to monitor a project are a subset of those published models (Boehm’s and McCall’s)
- The *“define your own quality model”* approach
  - Do not follow the models.
  - Meet with prospective users to reach a consensus on which quality attributes are important

Example of McCall Model (Correctness measure)

The McCall model, depicted in Figure 10.2, includes 41 metrics to measure the 23 quality criteria generated from the quality factors. Measuring any factor requires us first to consider a checklist of conditions that may apply to the requirements (R), the design (D), and the implementation (I). The condition is designated "yes" or "no," depending on whether or not it is met. To see how the metrics and checklists are used, consider measuring the criterion *completeness* for the factor *correctness*. The checklist for *completeness* is:

1. Unambiguous references (input, function, output) [R,D,I].
2. All data references defined, computed, or obtained from external source [R,D,I].
3. All defined functions used [R,D,I].
4. All referenced functions defined [R,D,I].
5. All conditions and processing defined for each decision point [R,D,I].
6. All defined and referenced calling sequence parameters agree [D,I].
7. All problem reports resolved [R,D,I].
8. Design agrees with requirements [D].
9. Code agrees with design [I].

Example of McCall Model (Correctness measure)

Notice that there are six conditions that apply to requirements, eight to design and eight to implementation. We can assign a 1 to a yes answer and 0 to a no, and we can compute the completeness metric in the following way to yield a measure that is a number between 0 and 1:

$$\frac{1}{3}\left(\frac{\text{Number of yes for R}}{6} + \frac{\text{Number of yes for D}}{8} + \frac{\text{Number of yes for I}}{8}\right)$$

Since the model tells us that *correctness* depends on *completeness*, *traceability*, and *consistency*, we can calculate analogous measures for the latter two. Then, the measure for *correctness* is the mean of their measures:

$$\text{Correctness} = \frac{x + y + z}{3}$$

That is, x, y, and z are the metrics for *completeness*, *traceability*, and *consistency*, respectively.*

ISO/IEC 9126-1 and ISO/IEC 25010 Standard Quality Models

- In 1992, a derivation of the McCall model was proposed as the basis for an international standard for software quality measurement and adopted.
- In 2011, ISO/IEC 25010 "Systems and Software Engineering—Systems and Software Quality Requirements and Evaluation (SQuaRE)" replaced it.
- In the ISO/IEC 25010 standard, software quality is defined to be the following: *"The degree to which a software product satisfies stated and implied needs when used under specified conditions."*

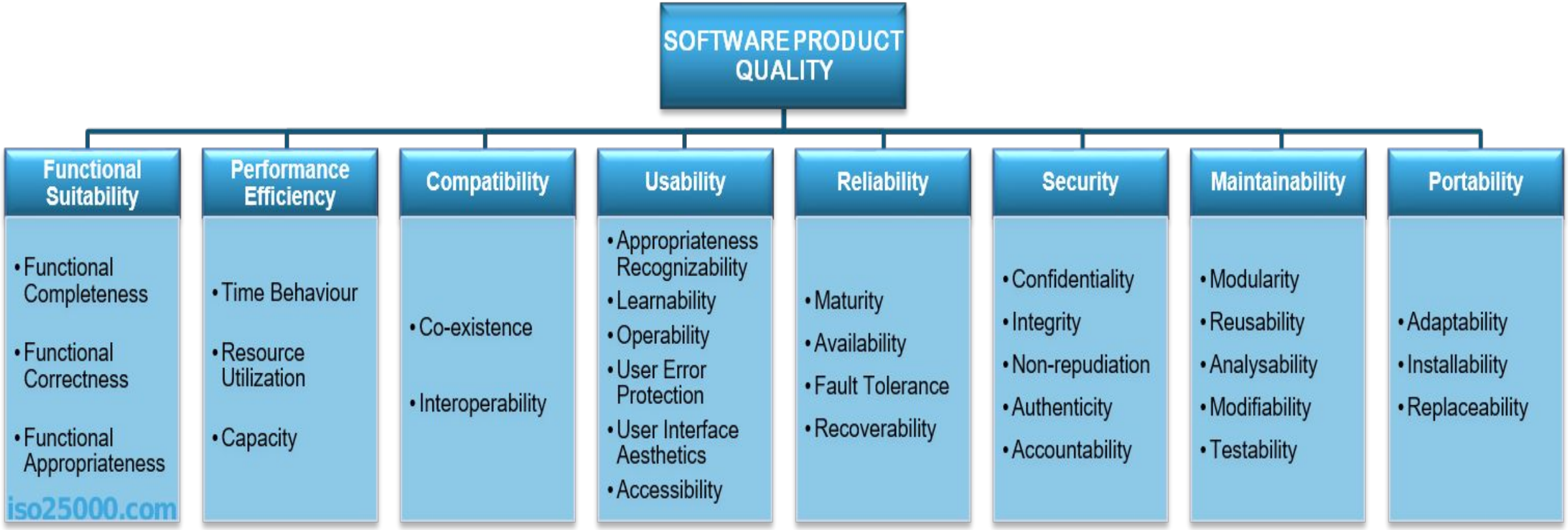
ISO= International Organization for Standardization
IEC= International Electrotechnical Commission

ISO/IEC 25010 Standard Quality Model

- The quality is decomposed into eight characteristics:
  1. Functional suitability
  2. Performance efficiency
  3. Compatibility
  4. Usability
  5. Reliability
  6. Security
  7. Maintainability
  8. Portability
- Each of the eight is defined in terms of other attributes of a relevant aspect of the software, and each can be refined through multiple levels of sub-characteristics.

https://iso25000.com/index.php/en/iso-25000-standards/iso-25010

ISO/IEC 25010 Standard Quality Model



Please check
https://iso25000.com/index.php/en/iso-25000-standards/iso-25010

Measuring aspect of quality (how to measure)

- Many software engineers base their quality assessments on measures defined for a specific purpose, separate from any formal quality model
- For example, certain systems have stringent requirements for *portability* (the ability to move an application from one host environment to another) and *integrity* (the assurance that modifications can be made only by authorized users). The user or practitioner may offer simple definitions of these terms, such as

$$\text{Portability} = 1 - ET/ER$$

- *ET* is a measure of the resources needed to move the system to the target environment, and *ER* is a measure of the resources needed to create the system for the resident environment

Measuring aspect of quality (how to measure)

- Defects-Based Quality Measures
  - $Defect\ density = \frac{Number\ of\ known\ defects}{Product\ size}$

- Defect density is not the only useful defect-based quality measure. For many years, many companies have defined quality in terms of *spoilage*. Specifically, they compute it as:

$$sytem\ spoilage = \frac{Time\ to\ fix\ post\ release\ defects}{Total\ system\ development\ time}$$

USABILITY MEASURES

- The ISO/IEC 25010 standard defines usability as follows:
  - *Usability* is the degree to which a product or system can be used by specified users to achieve specified goals with **effectiveness**, **efficiency**, and **satisfaction** in a specified context of use.

- *Effectiveness* measures indicate the degree to which users can correctly complete tasks. Thus, counts or percentages of completed tasks, as well as errors made, can measure the *effectiveness*
- *Efficiency* measures generally involve the time required to complete tasks. They may be concerned with the rate at which a user can input data via a keyboard, mouse, or other means. Studies have employed questionnaires, expert ratings, and users' ratings to measure the mental effort required for users to complete a task.
- *Satisfaction* measures indicate subjective notions of the quality of interactions with a software system. It can be measured using standard questionnaires about users' experiences. Preferences between alternatives can be determined by users' rankings or by observations of user behavior. Satisfaction can be evaluated during use using biological measurements such as pulse rates and facial expressions.

MAINTAINABILITY MEASURES

- *Maintainability* is the degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers.
- Mainly four types of maintenance activities: *corrective, adaptive, preventive, and perfective*
- Many measures of maintainability are expressed in terms of MTTR (*mean time to repair*).

SECURITY MEASURES

- The ISO/IEC 25010 standard defines security as follows:

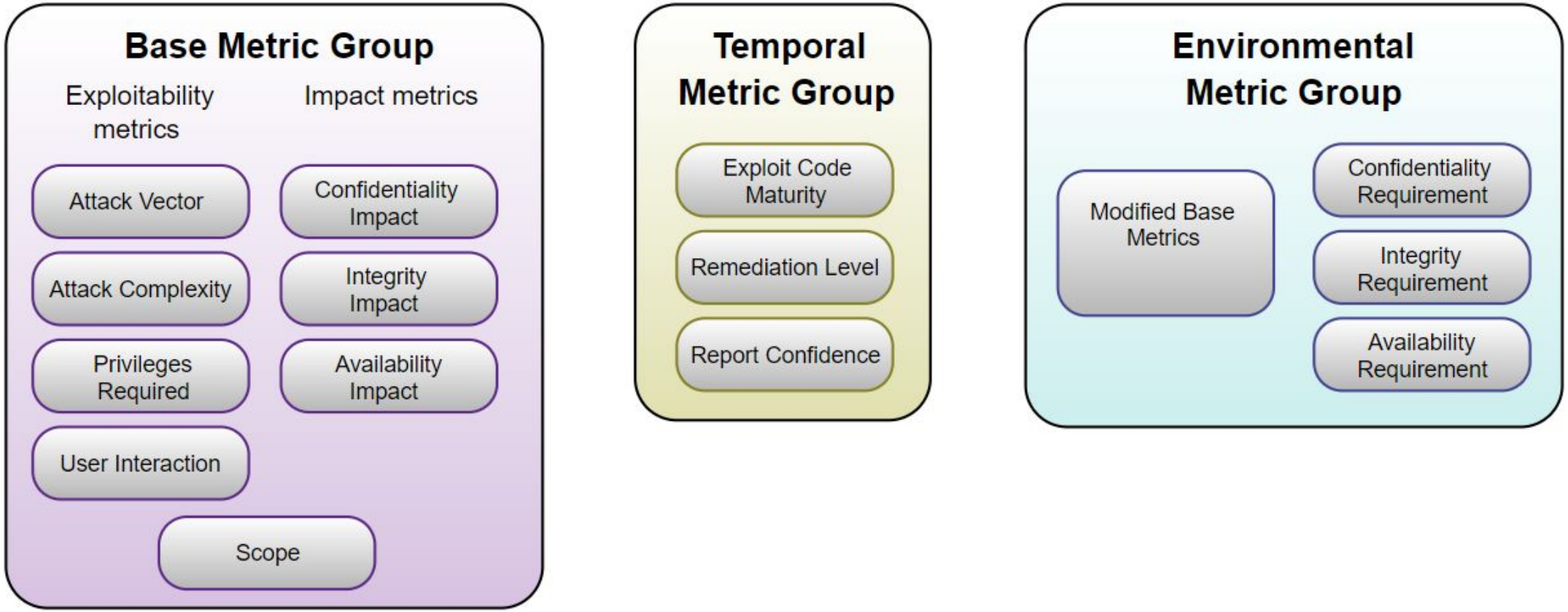
  Security is the degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization.

- Security is an external attribute primarily because of the interactions between external attackers and the systems.

Common Vulnerability Scoring System (CVSS)

- The Common Vulnerability Scoring System (CVSS) is **a free and open industry standard for assessing the severity of computer system security vulnerabilities**
- It is a widely used method, developed by the Forum of Incident Response and Security Teams (FIRST). The CVSS base metric group can have a score between zero and one, with *zero representing no vulnerability and one representing the maximum vulnerability*.
- It consists of three main metric groups: Base, Temporal, and Environmental. In turn, each of these groups consists of a set of metrics.
- The *base metric* group includes **six** individual measures related to the required access to exploit the vulnerability and the impact of the vulnerability
- Scores are calculated based on a formula that depends on several metrics that approximate the ease and impact of an exploit. **Scores range from 0 to 10, with 10 being the most severe**
- The current version of CVSS (CVSSv3.1) was released in June 2019.

Common Vulnerability Scoring System (CVSS)



Common Vulnerability Scoring System (CVSS)

- The Base Score reflects the severity of a vulnerability according to its intrinsic characteristics which are constant over time and assumes the reasonable worst-case impact.
- The Temporal Metrics adjust the Base severity of a vulnerability based on factors that change over time, such as the availability of exploit code.
- The Environmental Metrics adjust the Base and Temporal severities to a specific computing environment. They consider factors such as the presence of mitigations in that environment.

Common Vulnerability Scoring System (CVSS)

1. ***Access Vector (AV)*** indicates how remote an attacker can be to exploit the vulnerability.

Value	Description	Score
Local (L)	The attacker must either have physical access to the vulnerable system (e.g. firewire attacks) or a local account	0.395
Adjacent Network (A)	The attacker must have access to the broadcast or collision domain of the vulnerable system (e.g. ARP spoofing, Bluetooth attacks).	0.646
Network (N)	The vulnerable interface is working at layer 3 or above of the OSI Network stack. These types of vulnerabilities are often described as remotely exploitable (e.g. a remote buffer overflow in a network service)	1.0

Common Vulnerability Scoring System (CVSS)

2. *Access Complexity (AC)* indicates how complex the attack method needs to be in order to mount a successful attack.

Value	Description	Score
High (H)	Specialised conditions exist, such as a race condition with a narrow window, or a requirement for social engineering methods that would be readily noticed by knowledgeable people.	0.35
Medium (M)	There are some additional requirements for the attack, such as a limit on the origin of the attack, or a requirement for the vulnerable system to be running with an uncommon, non-default configuration.	0.61
Low (L)	There are no special conditions for exploiting the vulnerability, such as when the system is available to large numbers of users, or the vulnerable configuration is ubiquitous.	0.71

Common Vulnerability Scoring System (CVSS)

3. *Authentication (Au)* indicates whether an attacker needs to authenticate two or more times

Value	Description	Score
Multiple (M)	Exploitation of the vulnerability requires that the attacker authenticate two or more times, even if the same credentials are used each time.	0.45
Single (S)	The attacker must authenticate once in order to exploit the vulnerability.	0.56
None (N)	There is no requirement for the attacker to authenticate.	0.704

Common Vulnerability Scoring System (CVSS)

4. The ***Confidentiality (C)*** metric describes the impact on the confidentiality of data processed by the system.

Value	Description	Score
None (N)	There is no impact on the confidentiality of the system.	0.0
Partial (P)	There is considerable disclosure of information, but the scope of the loss is constrained such that not all of the data is available.	0.275
Complete (C)	There is total information disclosure, providing access to any / all data on the system. Alternatively, access to only some restricted information is obtained, but the disclosed information presents a direct, serious impact.	0.660

Common Vulnerability Scoring System (CVSS)

5. ***Integrity Impact*** (I) indicates whether there is no impact to the system integrity

Value	Description	Score
None (N)	There is no impact on the integrity of the system.	0.0
Partial (P)	Modification of some data or system files is possible, but the scope of the modification is limited.	0.275
Complete (C)	There is total loss of integrity; the attacker can modify any files or information on the target system.	0.660

Common Vulnerability Scoring System (CVSS)

6. The ***Availability (A)*** metric describes the impact on the availability of the target system. Attacks that consume network bandwidth, processor cycles, memory, or any other resources affect the availability of a system

Value	Description	Score
None (N)	There is no impact on the availability of the system.	0.0
Partial (P)	There is reduced performance or loss of some functionality.	0.275
Complete (C)	There is total loss of availability of the attacked resource.	0.660

Calculation

- These six metrics are used to calculate the ***exploitability*** and ***impact*** sub-scores of the *vulnerability*. These sub-scores are used to calculate the overall **BaseScore.**

$$\text{Exploitability} = 20 \times \text{AccessVector} \times \text{AccessComplexity} \times \text{Authentication}$$

$$\text{Impact} = 10.41 \times (1 - (1 - \text{ConfImpact}) \times (1 - \text{IntegImpact}) \times (1 - \text{AvailImpact}))$$

$$f(\text{Impact}) = \begin{cases} 0, & \text{if Impact} = 0 \\ 1.176, & \text{otherwise} \end{cases}$$

$$\text{BaseScore} = \text{roundTo1Decimal}(((0.6 \times \text{Impact}) + (0.4 \times \text{Exploitability}) - 1.5) \times f(\text{Impact}))$$