

# **CZ3002 - Advanced Software Engineering**

## **Quality Management**

**Faculty : Dr Shen Zhiqi**  
**School : School of Computer Science and Engineering**  
**Email : [zqshen@ntu.edu.sg](mailto:zqshen@ntu.edu.sg)**  
**Office : N4-02B-43**

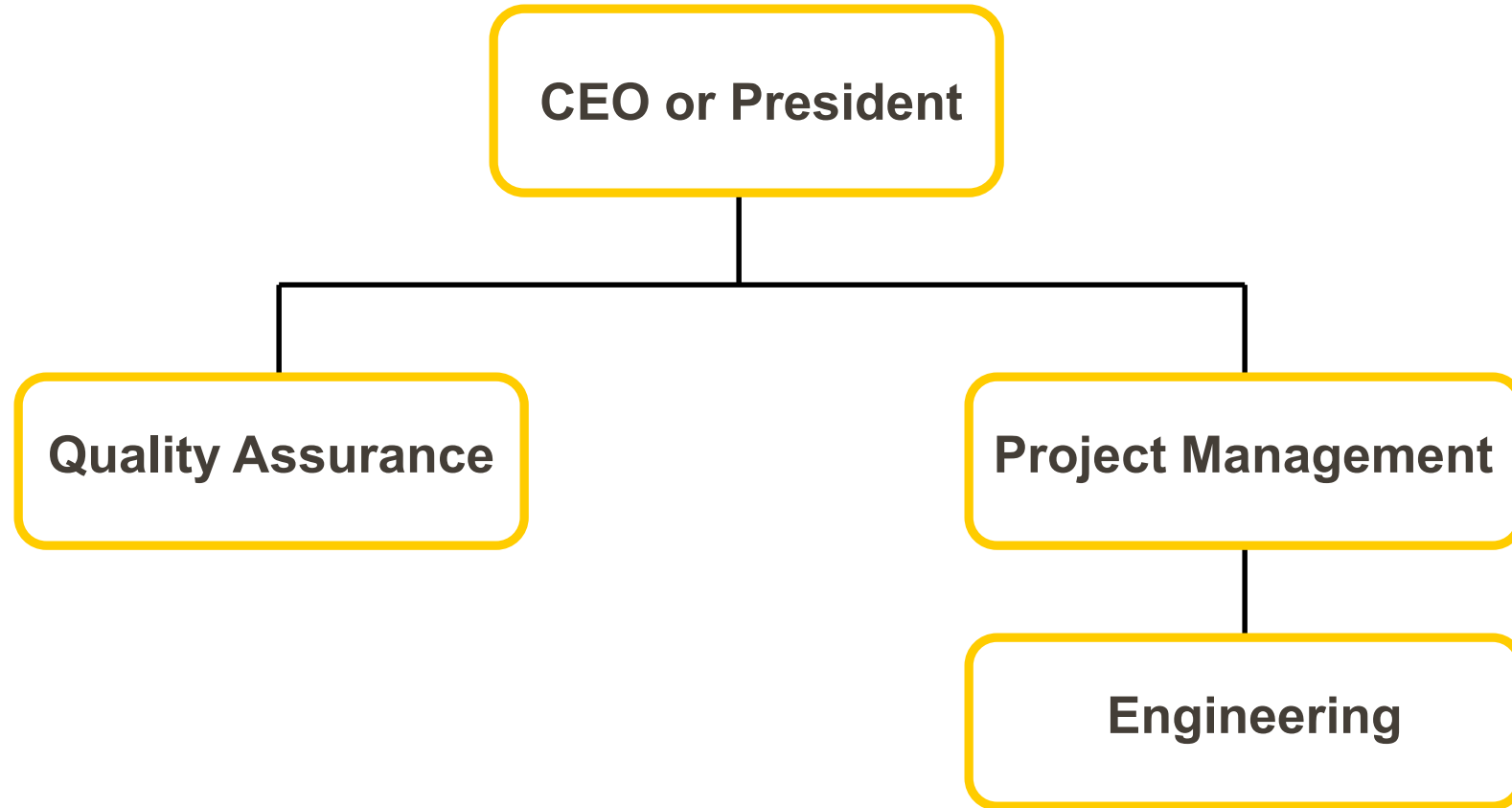
# Lesson Objectives

At the end of the lesson, you should be able to:

- ▶ Describe the 3 key quality management activities (quality assurance, quality planning and quality control)
- ▶ Explain the importance of standards in the quality management process
- ▶ Explain the purpose of software metrics and its current limitations of software measurement
- ▶ Describe the commonly used software product metrics and object-oriented metrics



# An Example of Software Organization



# What is Quality?

- ▶ Quality simply means that a product should **meet its specification** and **meet the customers' needs**.



## Software Quality Management

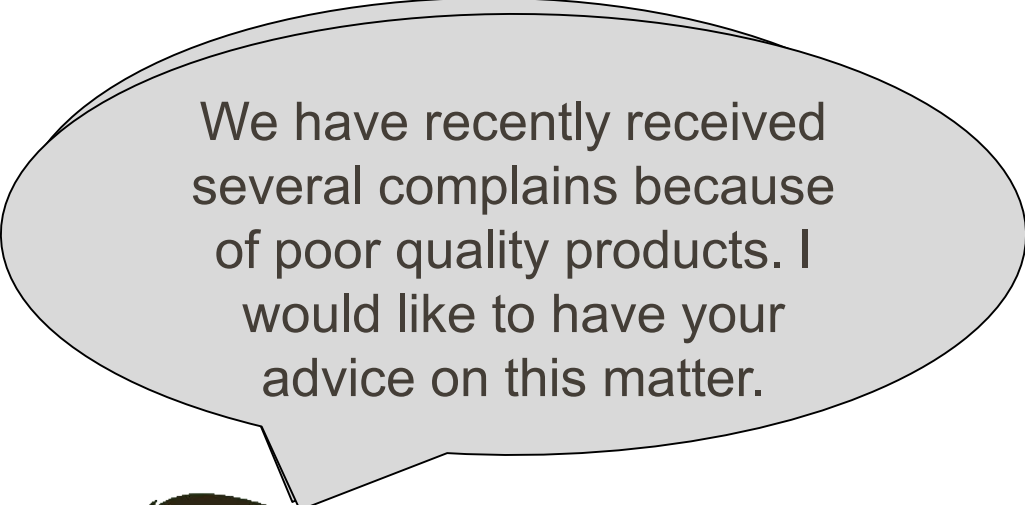
```
graph TD; A[Software Quality Management] --> B[Concerned with ensuring that the required level of quality is achieved in a software product.]; A --> C[Aims to develop a 'quality culture' where quality is seen as everyone's responsibility.]; A --> D[Involves defining appropriate quality standards and procedures and ensuring that these are followed.];
```

Concerned with ensuring that the required **level of quality** is achieved in a software product.

Aims to develop a '**quality culture**' where quality is seen as everyone's responsibility.

Involves defining appropriate **quality standards** and **procedures** and ensuring that these are followed.

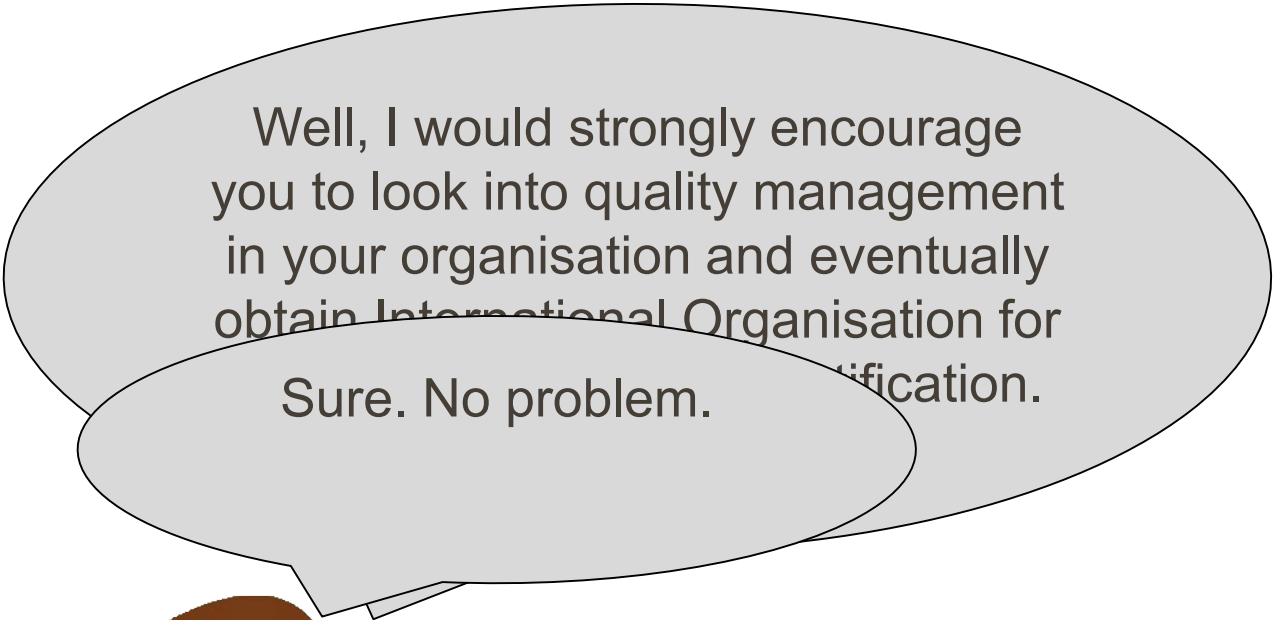
Below is the conversation between the CEO of a software development company and an adviser.



We have recently received several complains because of poor quality products. I would like to have your advice on this matter.



CEO



Well, I would strongly encourage you to look into quality management in your organisation and eventually obtain International Organisation for Standardisation certification.

Sure. No problem.

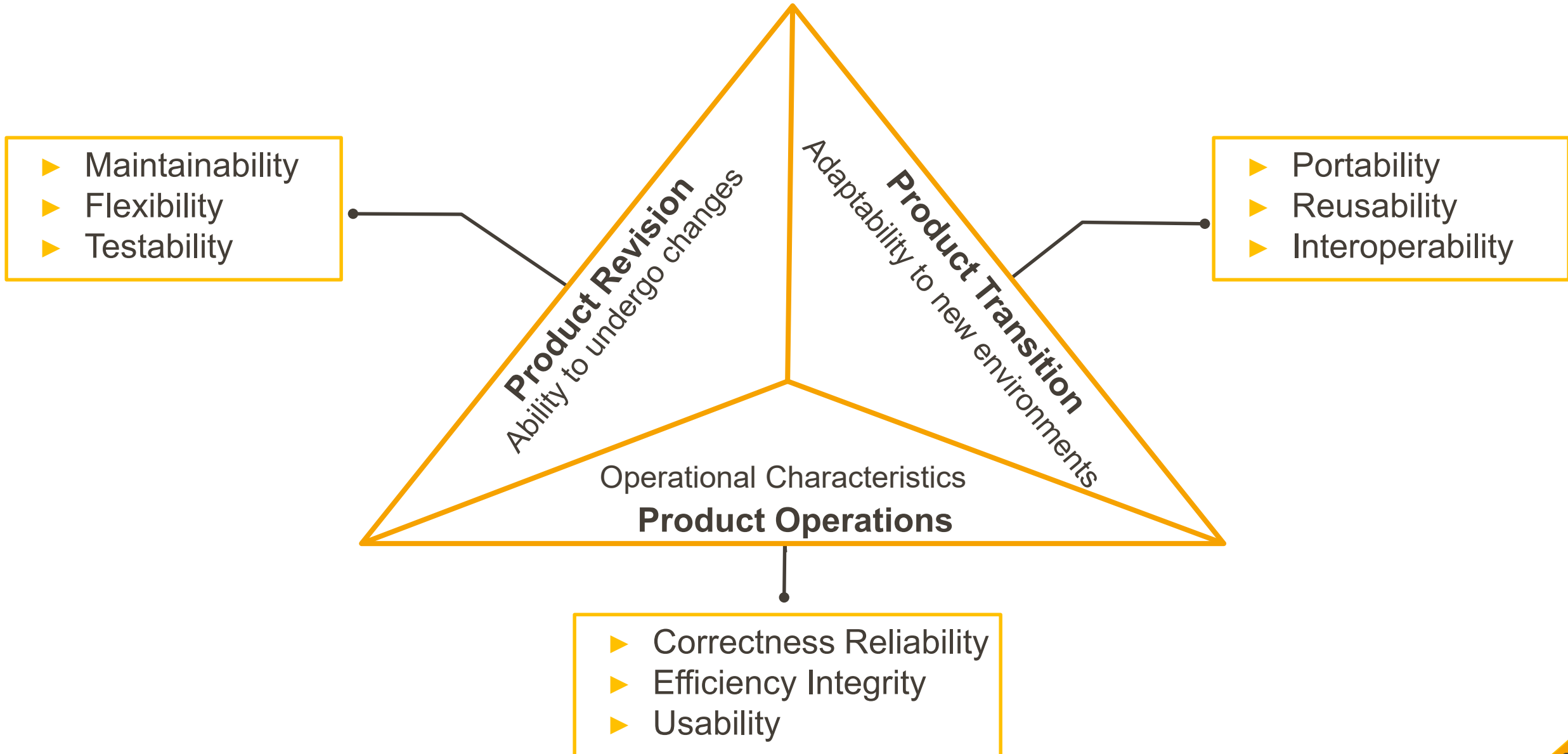


Adviser

Defining the quality of a product:

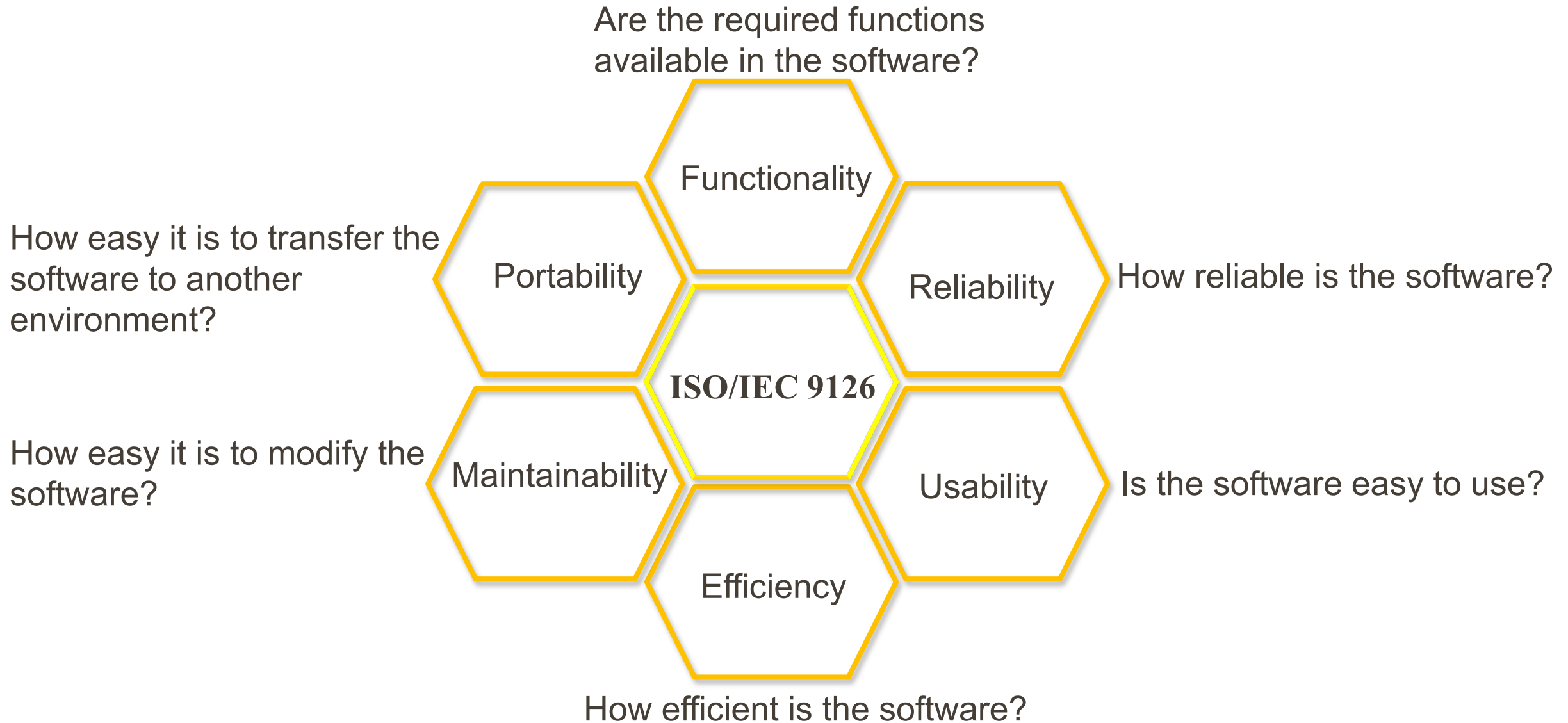
- ▶ Identify quality criteria
- ▶ Define quality assessment process
- ▶ Review the quality of the product
- ▶ Improve the quality of the product and the processes of the production

# McCall's Quality Model (1977)





# ISO 9126 Quality Model (2001)



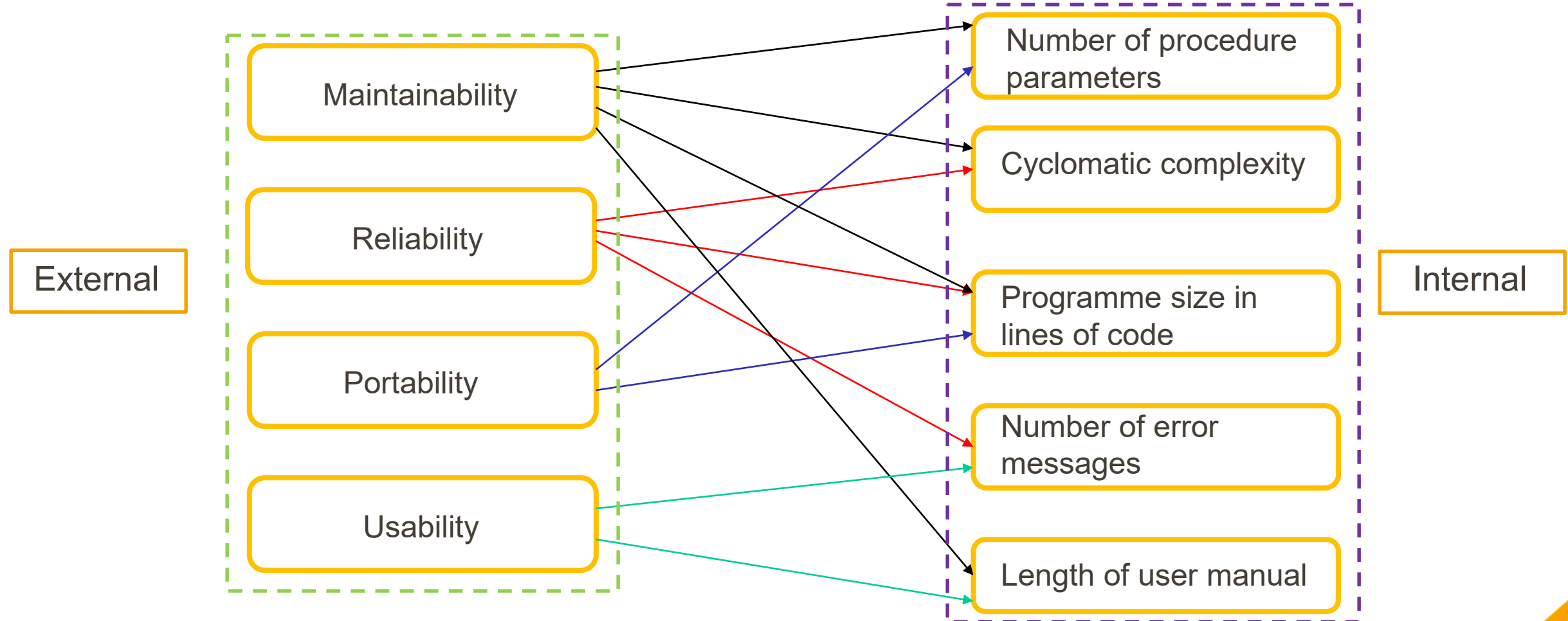
# ISO 9126 Quality Model (2001)

Characteristics	Description
Functionality	The capability of the software product to provide functions which meet stated and implied needs when the software is used under specified conditions.
Reliability	The capability of the software product to maintain its level of performance under stated conditions for a stated period of time.
Usability	The capability of the software product to be understood, learned, used and attractive to the user, when used under specified conditions.
Efficiency	The capability of the software product to provide appropriate performance, relative to a number of resources used, under stated conditions.
Maintainability	The capability of the software product to be modified. Modifications may include corrections, improvements or adaptations of the software to changes in the environment and in the requirements and functional specifications.
Portability	The capability of the software product to be transferred from one environment to another. The environment may include organisation, hardware or software environment.

# Internal and External Attributes

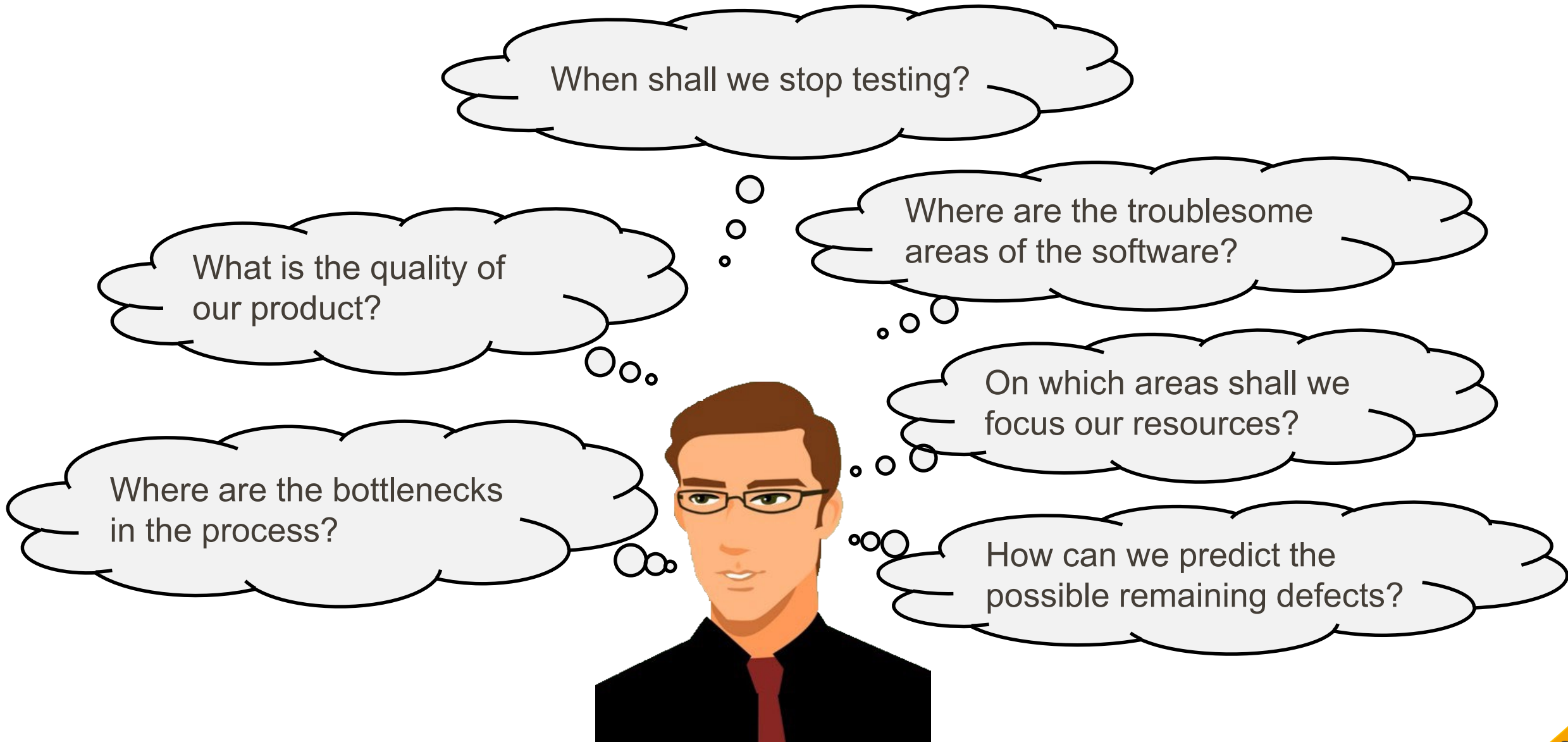
All the properties of the software as a product that users can experience and enjoy

All the properties of the software as seen by the developers that are desirable in order to facilitate the process of creating a good product



- ▶ Any type of **measurement** which relates to a software system, process or related documentation
- ▶ Lines of code in a program, the Fog index, number of person-days required to develop a component.
- ▶ Allow the software and the software process to be **quantified**.
- ▶ May be used to **predict** product attributes or to **control** the software process.
- ▶ **Product metrics** can be used for **general predictions** or to **identify** anomalous components.

# Metrics Help in Decision-Making



- ▶ A software property can be **measured**.
- ▶ The relationship exists between what we can measure and what we want to know. We can only measure internal attributes but are often more **interested** in external software attributes.
- ▶ A quality metric should be a predictor of product quality.

- Table below are some commonly used software metrics:

Software metric	Description
Fan in/Fan-out	Fan-in is a measure of the number of functions or methods that call some other function or method (say X). Fan-out is the number of functions that are called by function X. A high value for fan-in means that X is tightly coupled to the rest of the design and changes to X will have extensive knock-on effects. A high value for fan-out suggests that the overall complexity of X may be high because of the complexity of the control logic needed to coordinate the called components.
Length of code	This is a measure of the size of a program. Generally, the larger the size of the code of a component, the more complex and error-prone that component is likely to be. Length of code has been shown to be one of the most reliable metrics for predicting error-proneness in components.
Cyclomatic complexity	This is a measure of the control complexity of a program. This control complexity may be related to program understandability.
Length of identifiers	This is a measure of the average length of distinct identifiers in a program. The longer the identifiers, the more likely they are to be meaningful and hence the more understandable the program.
Depth of conditional nesting	This is a measure of the depth of nesting of if-statements in a program. Deeply nested if statements are hard to understand and are potentially error-prone.
Fog index	This is a measure of the average length of words and sentences in documents. The higher the value for the Fog index, the more difficult the document is to understand.

- Table below are some commonly used object-oriented metrics:

OO Metric	Description
Depth of inheritance tree	This represents the number of discrete levels in the inheritance tree where sub-classes inherit attributes and operations (methods) from super-classes. The deeper the inheritance tree, the more complex the design. Many different object classes may have to be understood to understand the object classes at the leaves of the tree.
Method fan-in/fan-out	This is directly related to fan-in and fan-out as described above and means essentially the same thing. However, it may be appropriate to make a distinction between calls from other methods within the object and calls from external methods.
Weighted methods per class	This is the number of methods that are included in a class weighted by the complexity of each method. Therefore, a simple method may have a complexity of 1 and a large and complex method a much higher value. The larger the value for this metric, the more complex the object class. Complex objects are more likely to be more difficult to understand. They may not be logically cohesive so cannot be reused effectively as super-classes in an inheritance tree.
Number of overriding operations	This is the number of operations in a super-class that are over-ridden in a sub-class. A high value for this metric indicates that the super-class used may not be an appropriate parent for the sub-class.



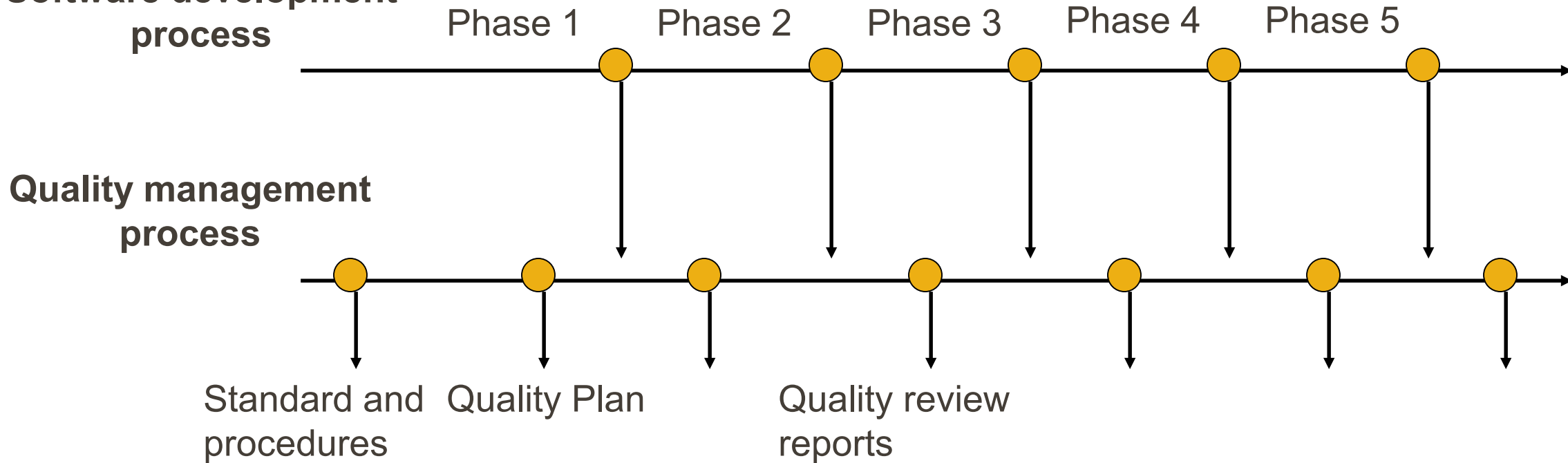
# Example

Identify the internal attributes of

Quality Characteristic	Procedures / Considerations
Functionality	Traceability is important
Reliability	
Usability	Size of font is important to elderly, colour is also a factor
Efficiency	
Maintainability	
Portability	

# Quality of Process Affects Quality of Product

**Software development  
process**



Defining the quality of a process:

- ▶ Identify quality criteria
- ▶ Define quality assessment process
- ▶ Review the quality of the process
- ▶ Improve the quality of the process

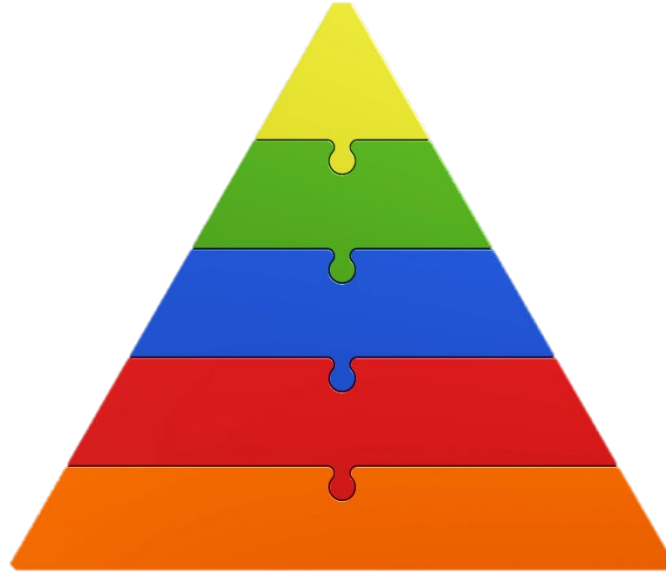
# Capability Maturity Model Overview

LEVEL	CHARACTERISTIC	KEY PROCESS AREA	RESULTS
5 Optimising	Improvement fed back into process	Process change management Technology innovation Defect prevention	Productivity and quality
4 Managed	(Quantitative) Measured Process	Quality management Process measurement & analysis	
3 Defined	(Qualitative) Process defined and Institutionalised	Peer reviews Intergroup coordination Software product engineering Integrated software management Training program Organisation process definition Organisation process focus	
2 Repeatable	(Intuitive) Process dependent on individuals	Software configuration management Software quality assurance Software project tracking & oversight Software subcontract management Software project planning Requirements management	
1 Initial	Ad hoc/ chaotic	Survival	Risk

# Importance of Standards



Encapsulation of best practice-  
avoids repetition of past mistakes.



They are a framework for quality  
assurance processes - they involve  
checking compliance to standards.



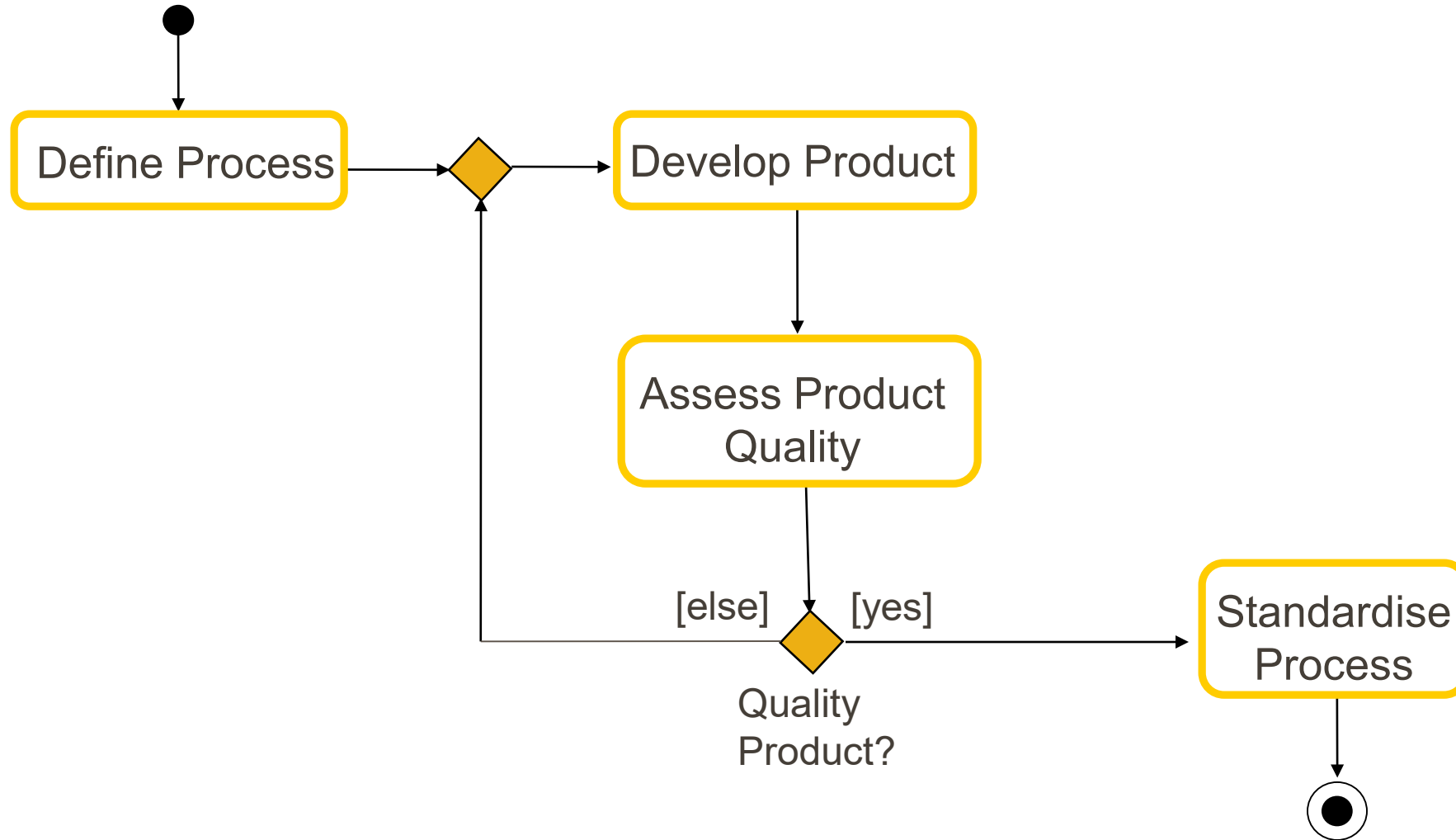
They provide continuity - new staff  
can understand the organisation  
by understanding the standards  
that are used.



- ▶ Quality standards and procedures should be documented in an organisational quality manual.
- ▶ An external body may certify that an organisation's quality manual conforms to ISO 9000 standards.
- ▶ Some customers require suppliers to be ISO 9000 certified although the need for flexibility here is increasingly recognised.



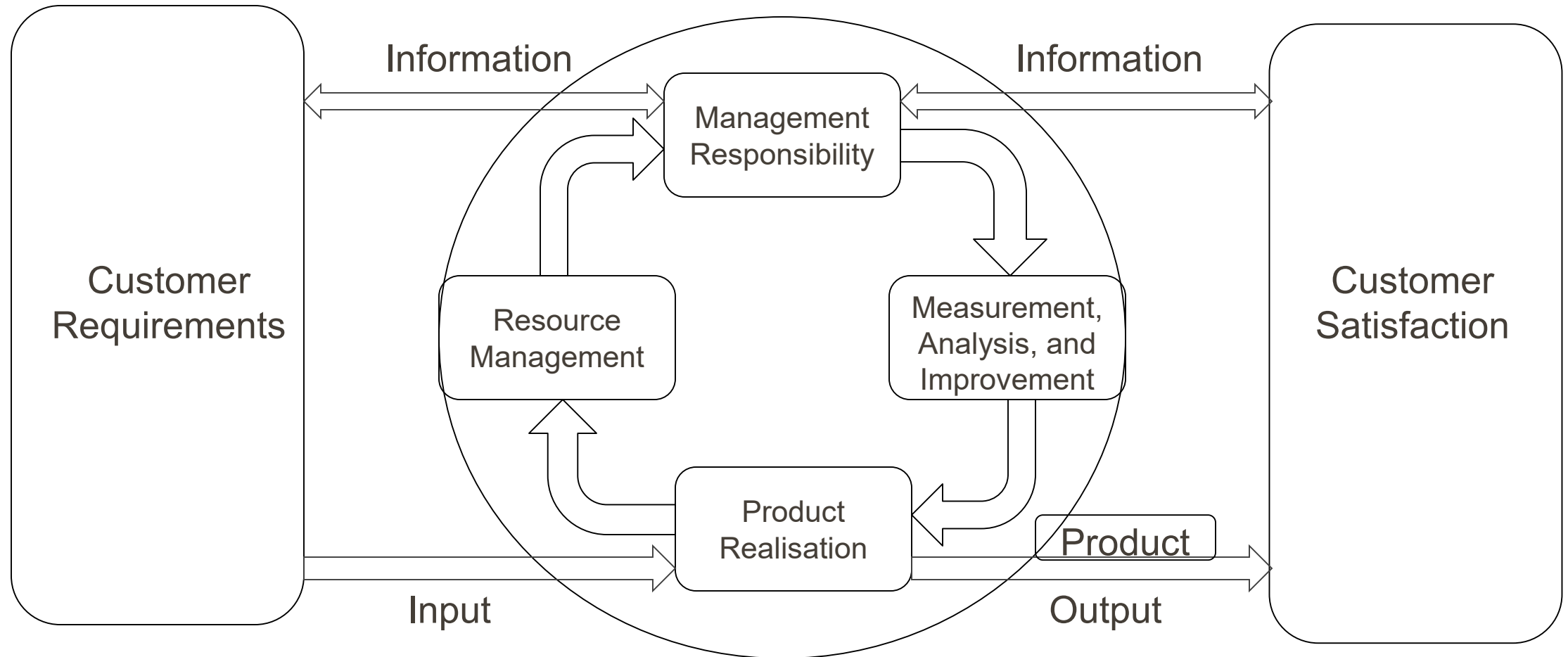
# Process-Based Quality Assurance





# Model of Process-Based QMS

## Continual Improvement of the Quality Management System



- ▶ This involves **checking** the software development process to ensure that procedures and standards are being followed.
- ▶ There are two approaches to quality control:
  - ❖ Quality reviews;
  - ❖ Automated software assessment and software measurement.

Quality assessment includes:

- ▶ Product assessment – output of processes
- ▶ Process assessment – tasks, activities and procedures that produce products

Quality assessment process:

- ▶ Define quality assessment system/mechanism (e.g. review every month)
- ▶ Choose the right metrics (measurement)
- ▶ Collect the data during project execution
- ▶ Analyze the data, conclude assessment results
  - ▶ Measure time, cost, quality, output (products), complexity (process), etc.

# Use Software Metrics

How is **software quality** in each phase of the SDLC being **evaluated**?

**Specifications/  
Requirements**

**Analysis and  
Design**

**Implementation**

**Testing**

**Maintenance**

**Software Development Life Cycle (SDLC)**



# Example

For each characteristic, we need to do the following at each SDLC stage  
There may be one table for each internal attribute of the characteristic  
e.g. For traceability under Functionality:

SDLC Stage	Procedures/ Considerations
Requirements/ Specification	Change history (number of changes, approved changes, etc.), Review
Analysis and Design	Cross reference (number of uncited designs, etc.)
Implementation	
Testing	
Maintenance	

- ▶ A quality plan sets out the desired product qualities and how these are assessed and defines the most significant quality attributes.
- ▶ The quality plan should define the **quality assessment** process.
  - ▶ Product assessment
  - ▶ Process assessment
- ▶ It should set out which organisational standards should be applied and, where necessary, define new standards to be used.
- ▶ A quality plan writing guideline and template is given under the lab folder of NTULearn
- ▶ You can add/delete/modify the headings/chapters according to your own project.

# Software Quality Assurance Plan

- ▶ Purpose, scope and quality objectives
- ▶ Reference documents/ standards
- ▶ Organizational roles, responsibilities and **assessment mechanisms (Ch3.2)**
- ▶ Document standards/ Templates
- ▶ **Standard, processes, practices, conventions, and metrics (Ch5)**
- ▶ **Review and audit (Ch6)**
- ▶ Others (if any)
  - ▶ Testing, reporting, selecting tools, recording and training
  - ▶ Media control, supplier control, risk management and change management

# Summary

