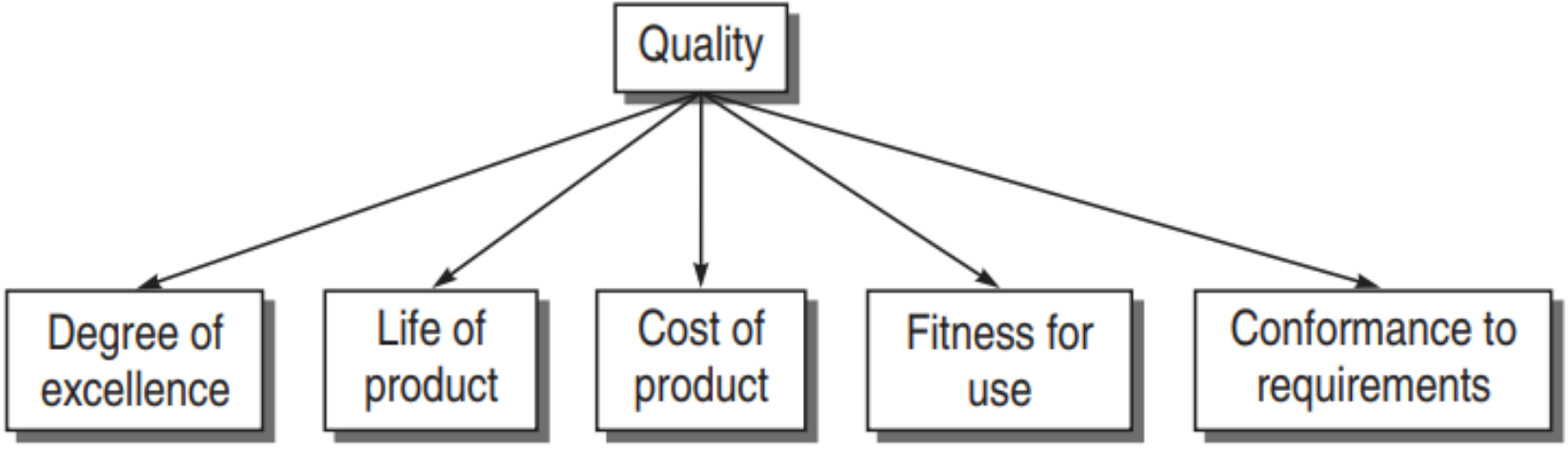
Software Quality Management

Software Testing

Quality

- We use the term *quality* very often in our daily life. We expect a certain level of quality from a product. But how do we define quality?
- Crosby defines quality as *conformance to requirements*. This means that the product should be developed according to the pre-specified requirements.
- Pressman defines quality as *an attribute of an item; quality refers to measurable characteristics—things we are able to compare to known standards*.

Quality as multi-dimensional concept



QUALITY COST

- Prevention costs are related with activities that identify the cause of defects and those actions that are taken to prevent them, e.g. quality planning, formal technical reviews, test and laboratory equipments, training, defect causal analysis and prevention.
- Appraisal costs include the costs of evaluating the quality of software products at some level, e.g. testing, in-process and inter-process inspec- tion by implementing software metrics programs (SMP), equipment calibrations, and maintenance.
- Failure costs include the costs to analyse and remove the failures.

QUALITY FACTORS (1/3)

- *Functionality* The major visible factor for software quality is its functionality such that all the intended features are working.
- *Correctness* Software should be correct in nature largely satisfying specifications and fulfilling user's objectives. Another measure of correctness may be the extent to which a software is fault-free.
- *Completeness* A software product should be complete to the extent that all its parts are present and fully developed. This means that if the code calls a sub- routine from an external library, the software package must provide reference to that library and all the required parameters must be passed. All the required input data must be available.

QUALITY FACTORS (2/3)

- *Efficiency* The software should be efficient in its working, i.e. utilizing all resources.
- *Portability* A software product should be operated on all well-known platforms and configurations.
- *Testability* A software should be testable such that it is easy to debug and maintain.
- *Usability* The software should be easy to use, operate, and understand in every manner that its user wants. The usability criterion may differ from project to project.
- *Maintainability* The software should have the scope of maintaining it, if a bug appears or a new requirement needs to be incorporated.

QUALITY FACTORS (3/3)

- *Reliability* It is the ultimate factor that a user demands from any product. A software product should also be reliable in the sense that all its desired features must run forever without bugs; it should not stop after working for some years.
- *Reusability* It is an important factor, as it can greatly reduce the effort required to engineer a software product. However, it is not easy to achieve.
- *Integrity* It defines the extent to which an unauthorized access or a modification can be controlled in a computer system (either manually or with automated tools).

SOFTWARE QUALITY METRICS: Product Quality Metrics

- ***Mean-time to failure (MTTF)*** MTTF metric is an estimate of the average or mean time until a product's first failure occurs. It is most often used with safety critical systems such as the airline traffic control system.
- ***Defect density*** *metrics* It measures the defects relative to the software size. Thus, it is the defect rate metrics or the volume of defects generally used for commercial software systems. It is important for cost and resource estimate of the maintenance phase of the software lifecycle.

  Defect density = Number of defects / Size of product

SOFTWARE QUALITY METRICS : Process Quality Metrics

- *Defect-density during testing* Higher defect rates found during testing is an indicator that the software has experienced higher error injection during its development process.
- *Defect-arrival pattern during testing* The pattern of defect arrivals or the time between consecutive failures gives more information.
- *Defect-removal efficiency DRE =*

```
(Defects removed during the month / Number of problem
  arrivals during the month) * 100
```

SOFTWARE QUALITY METRICS : Process Quality Metrics

- ***Customer problem metrics*** This metric measures the problems which customers face while using the product. The problems may be valid defects or usability problems, unclear documentation, etc. The problem metrics is usually expressed in terms of problems per user month (PUM), as given below:

  PUM = Total problems reported by the customer for a time period / Total number of licensed months of the software during the period

  where number of licensed months =

  number of installed licenses of the software × number of months in the calculation period

SOFTWARE QUALITY METRICS : Process Quality Metrics

- ***Customer satisfaction metrics***
  Customer satisfaction is usually measured through various methods of customer surveys via the Likert five-point scale

  1. Very satisfied
  2. Satisfied
  3. Neutral
  4. Dissatisfied
  5. Very dissatisfied

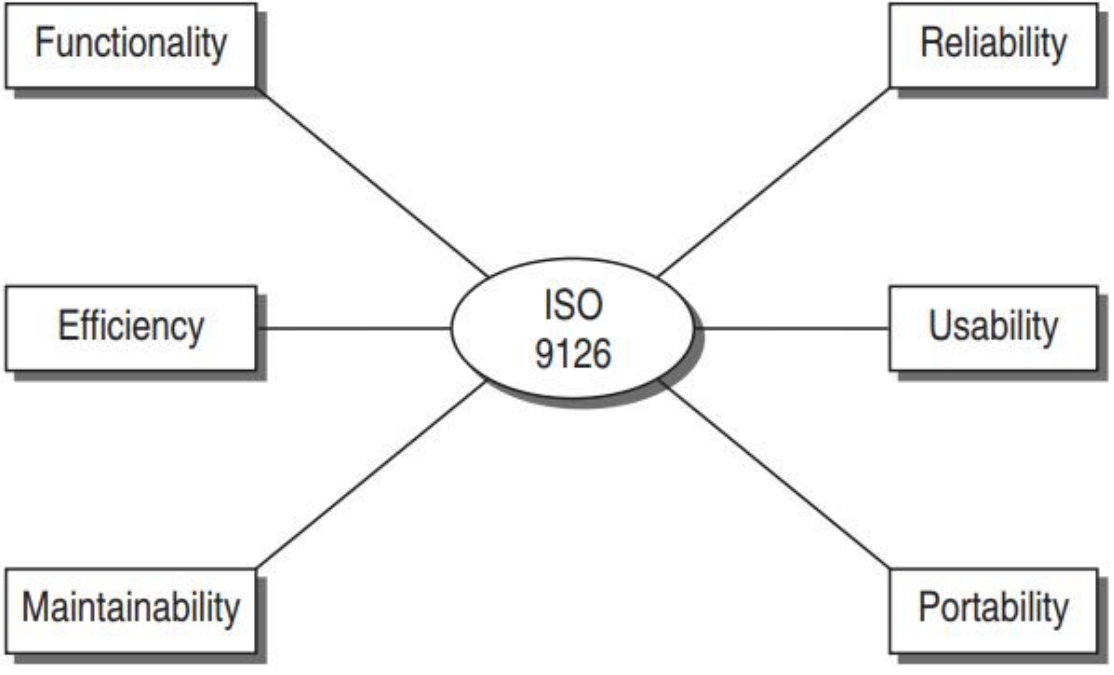
Metrics for Software Maintenance

- *Fix backlog and backlog management index*
  Fix backlog metrics is the count of reported problems that remain open at the end of a month or a week. This metric can provide meaningful information for managing the maintenance process. Backlog management index (BMI) is also the metric to manage the backlog of open unresolved problems.

$$BMI = \frac{Number\ of\ \ problems\ closed\ during\ the\ month}{Number\ of\ \ problem\ arrivals\ during\ the\ month} \times 100$$

- If BMI is larger than 100, it means the backlog is reduced; if BMI is less than 100, the backlog is increased. Therefore, the goal is to have a BMI larger than 100

SQA MODELS: ISO 9126



CAPABILITY MATURITY MODEL (CMM)

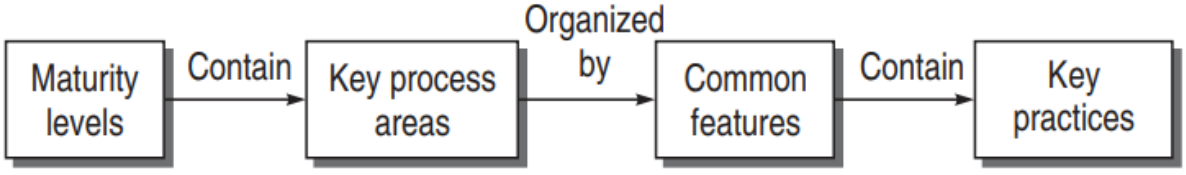


Figure 13.7(a) CMM Structure

Maturity levels	Indicate	Process capability
Key process areas	Achieve	Goals
Common features	Address	Implementation/Institutionalization
Key practices	Describe	Infrastructure/Activities

Figure 13.7(b) CMM Structure

5 key process areas (KPAs)

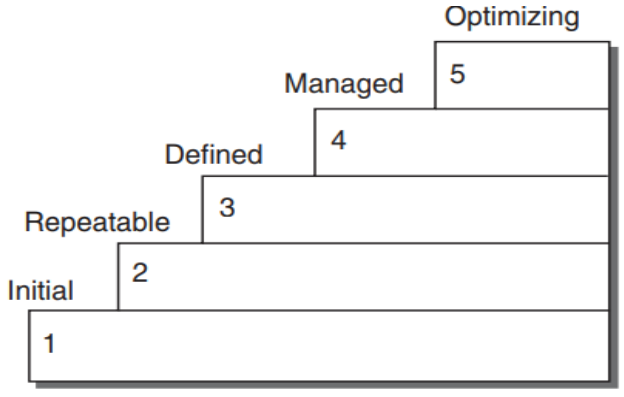


Figure 13.8 Maturity levels

Table 13.1 Process capability at each maturity level

Maturity level	Process Capability
1	—
2	Disciplined process
3	Standard consistent process
4	Predictable process
5	Continuously improving process

SIX SIGMA

- Six sigma is a quality model originally developed for manufacturing processes.
- It was developed by Motorola. Six sigma derives its meaning from the field of statistics.
- Sigma is the standard deviation for a statistical population.
- Six sigma means, if one has six standard deviations between the mean of a process and the nearest specification limit, there will be practically no items that fail to meet the specifications
- Six sigma is not just all about statistics. It can be applied to software problems which affects its quality.

Thank You