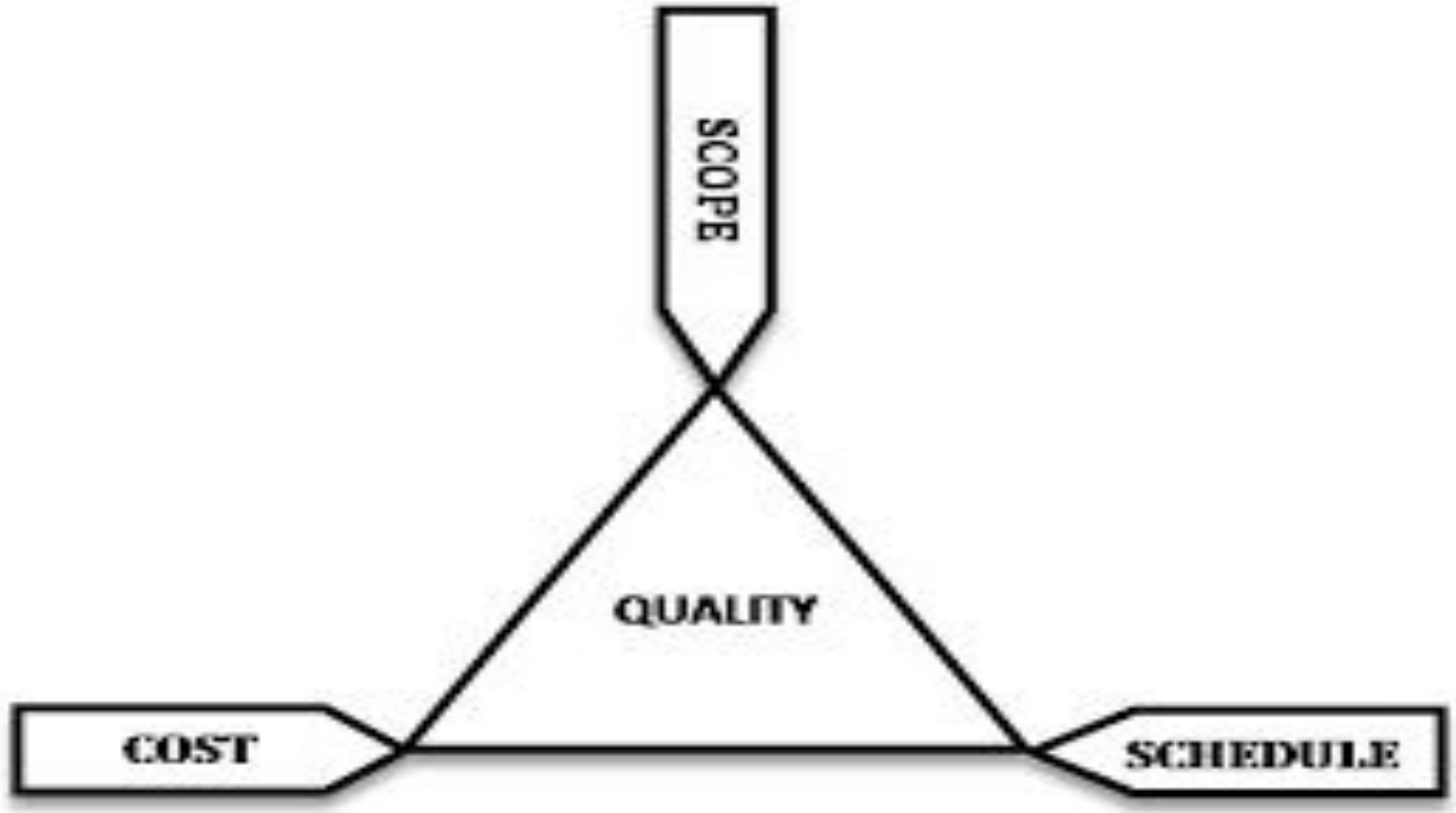
Introduction to software engineering

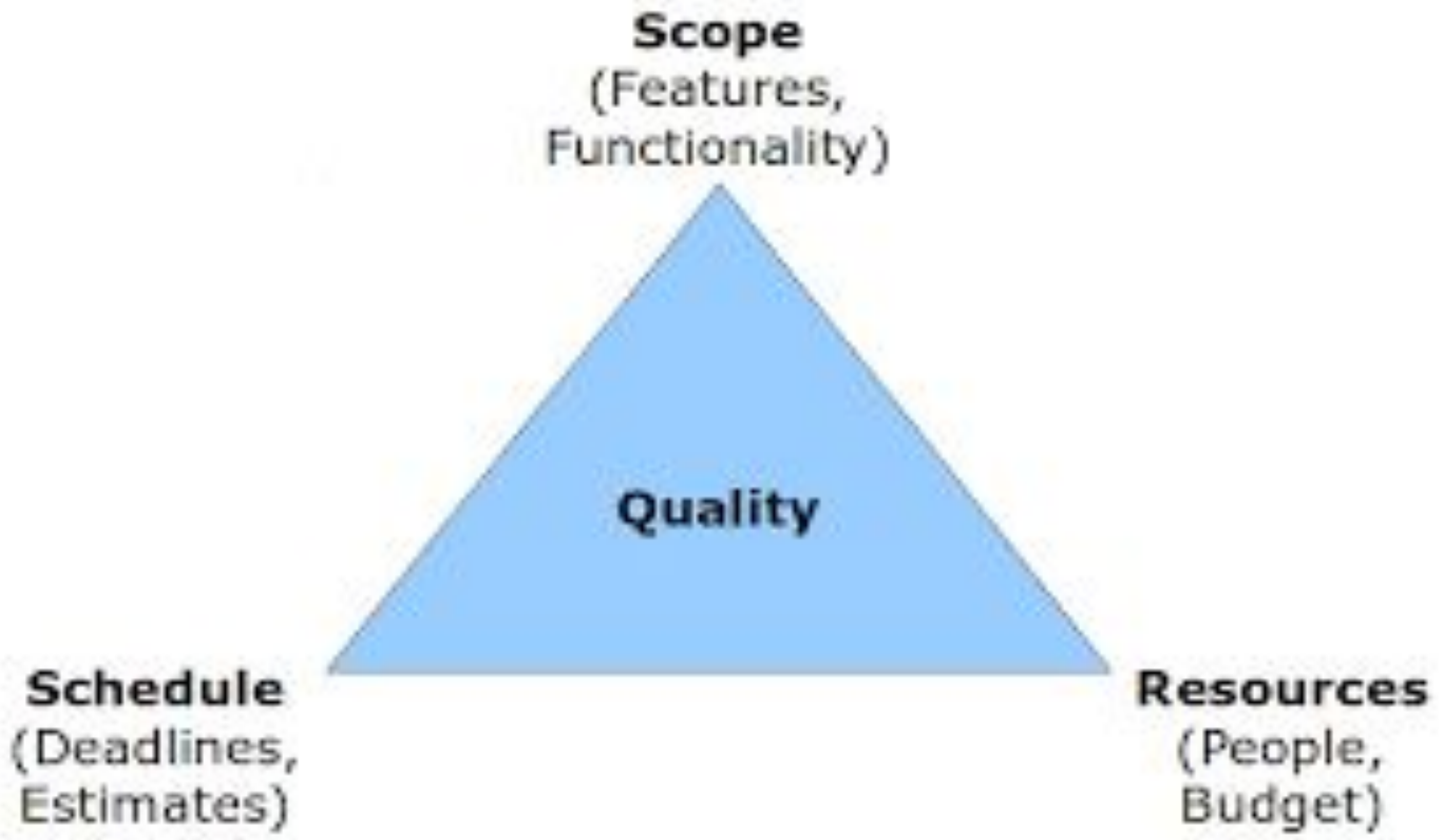
Dr. Bharati Wukkadada

contents

- Meaning of s/w, s/w engg
- s/w components
- s/w characteristics
- s/w crisis
- s/e processes
- s/w quality attributes

Project triangle




Scope
(Features, Functionality)
Quality
Schedule
(Deadlines, Estimates)
Resources
(People, Budget)

software

- Is a PRODUCT- software professoionals build it
- Every one in the world use s/w
- It effects every aspect of our lives, it build complex systems in a timely manner and with high quality
- Apply- agile, adaptable process that leads to high quality results, that meets the needs of people
- Work-product- has set of prgs, content(data) , s/w

Programmers have few questions:

- ✔ Why does it take so long to get s/w finished
- ✔ Why are development costs so high
- ✔ Why can’t we find all errors before we give the s/w to our cust
- ✔ Why do we spend so much time and effort maintain existing progrms
- ✔ Why do we continue to have difficulty in measuring progress as s/w is being developed and maintained

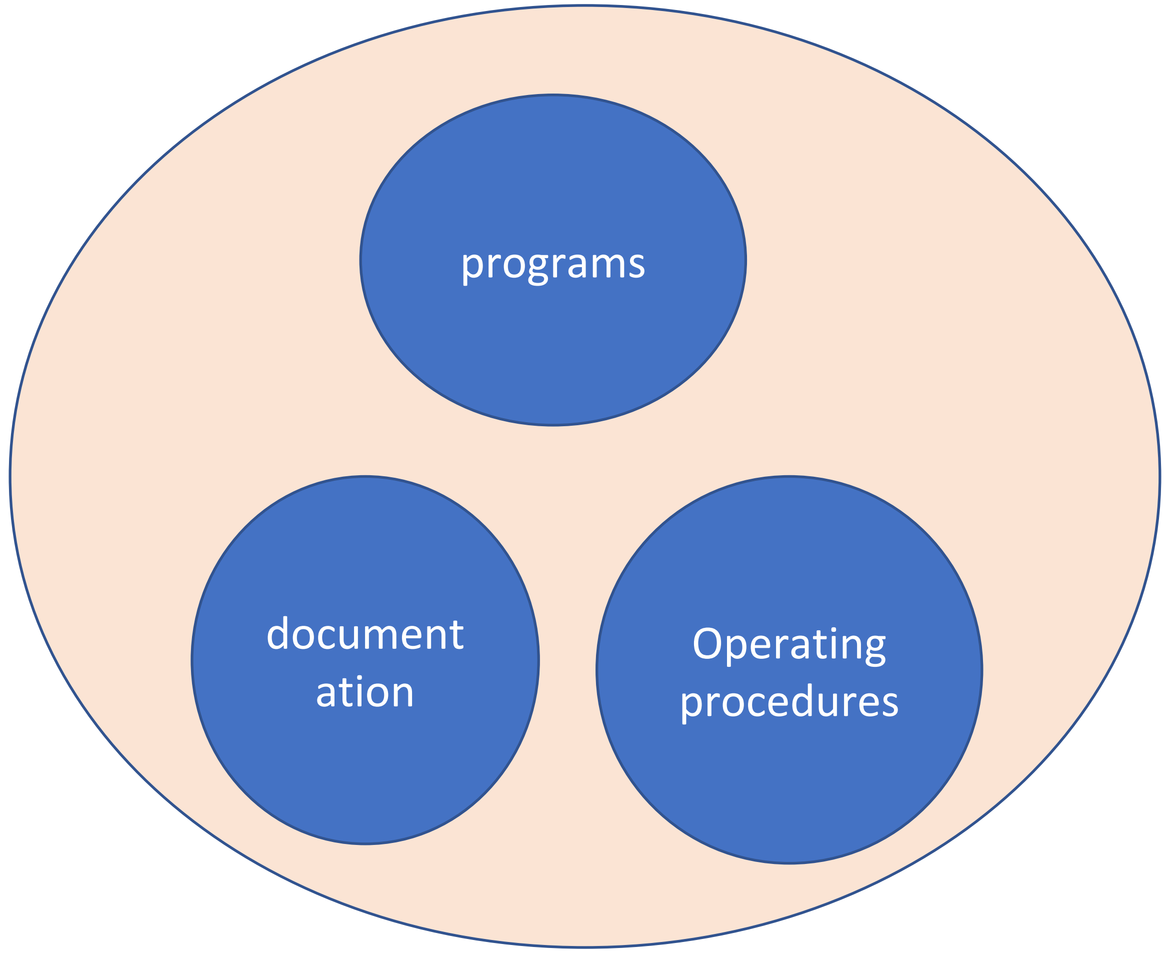
Software

- Instructions(computer prgs) that when executed provide desired features, functions and performances
- Data structures that enable the prgs to adequately manipulate information
- Descriptive information in both hardcopy and virtual forms that describes the operation and use of the prgs

Components of software

Any prg is a subset of s/w and it become s/w only if documentation and operating procedure manuals are prepared

- **Prg: is a combination of source code and object code**
- **Documentation consists of different types of manuals**
- **OP: consists of instructions to setup and use of the s/w system**



Documentation manual

- Analysis
    - formal specification
    - Context dia
    - dfd
- Design
    - Flow charts
    - ER dia
- Implementation
    - Source code listings
    - Cross-reference listings
- Testing
    - Test data
    - Test results

Operating procedure

- User manuals
    - System overview
    - Beginner’s guide tutorial
    - Reference guide
- Operational manuals
    - Installation guide
    - System administration guide

Software components

- “ a component represents a modular, deployable and replaceable part of the system that encapsulates implementation and exposes a set of interfaces”
- A reusable module is an independent and deliverable s/w part that encapsulates a functional specification and implementation for reuse by a third party. Based on unit’s specification, implementation and well defined contracted interfaces

Software components

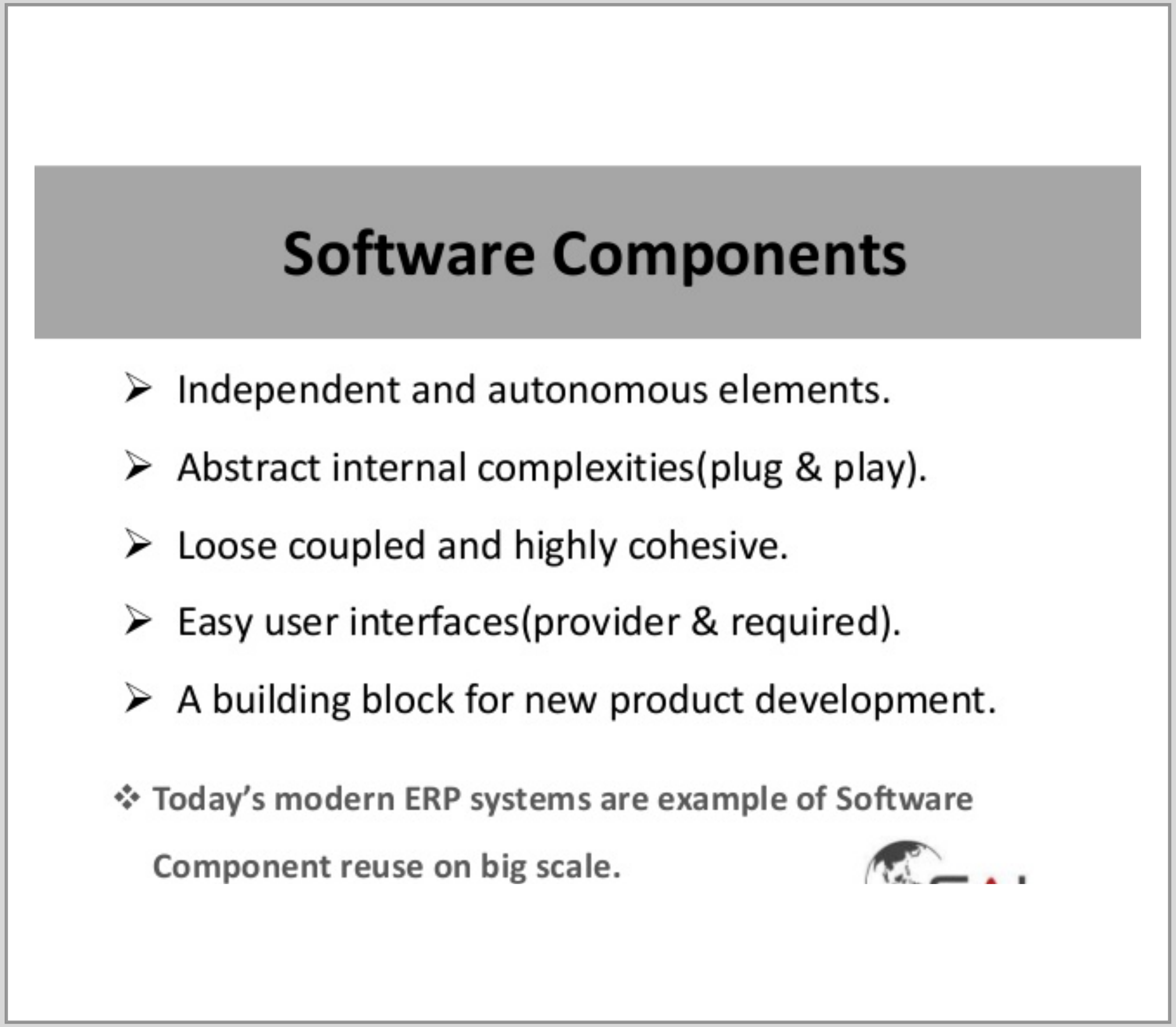
- s/w is more than a programs.
- A s/w component is a unit of composition with contracts specified interfaces and explicit contract dependencies only.

✔ Off the shelf components: existing s/w that can be acquired from a third party

✔ Full experience components: existing past projects that are similar to the s/w to be built for the current project and team member have full experience

✔ Partial experience components: existing past projects that are related to the s/w to be built for the current project needs substantial modifications

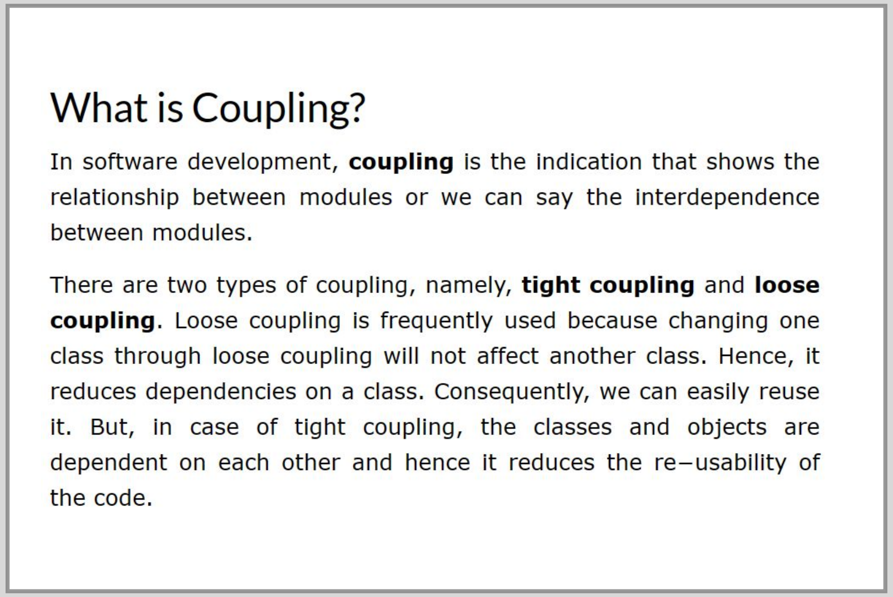
✔ New components: s/w components that must be built by the s/w team specifically for the needs of the current project



Software Components

- ➢ Independent and autonomous elements.
- ➢ Abstract internal complexities(plug & play).
- ➢ Loose coupled and highly cohesive.
- ➢ Easy user interfaces(provider & required).
- ➢ A building block for new product development.

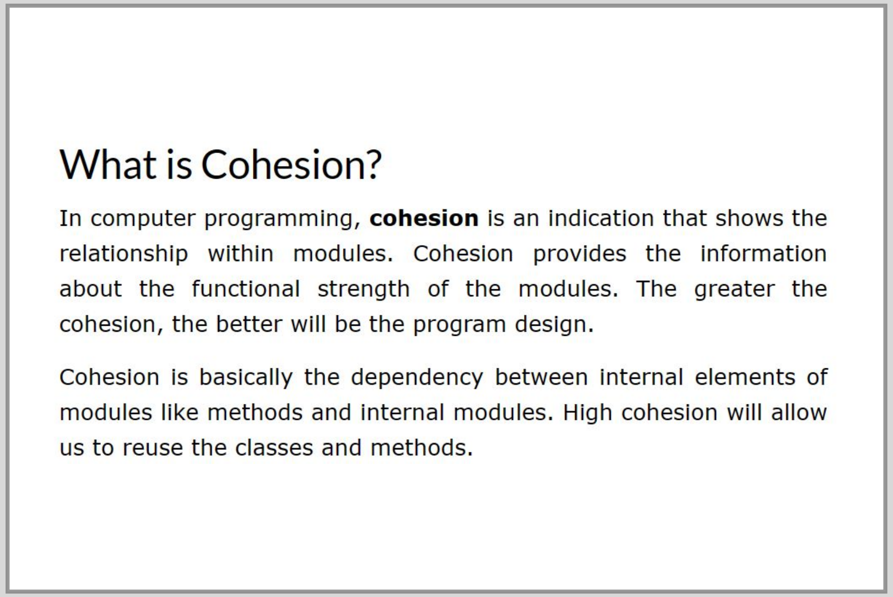
❖ Today's modern ERP systems are example of Software Component reuse on big scale.



What is Coupling?

In software development, **coupling** is the indication that shows the relationship between modules or we can say the interdependence between modules.

There are two types of coupling, namely, **tight coupling** and **loose coupling**. Loose coupling is frequently used because changing one class through loose coupling will not affect another class. Hence, it reduces dependencies on a class. Consequently, we can easily reuse it. But, in case of tight coupling, the classes and objects are dependent on each other and hence it reduces the re−usability of the code.



What is Cohesion?

In computer programming, **cohesion** is an indication that shows the relationship within modules. Cohesion provides the information about the functional strength of the modules. The greater the cohesion, the better will be the program design.

Cohesion is basically the dependency between internal elements of modules like methods and internal modules. High cohesion will allow us to reuse the classes and methods.

Cohesion	Coupling
Cohesion is the measure of degree of relationship between elements of a module.	Coupling is the measure of degree of relationship between different modules.
It is an intra module concept.	It is an inter module concept.
It represents relationships within the module.	It helps represent the relationships between the modules.
Increased cohesion is considered to be good for the software.	Increased coupling has to be avoided in software.
It represents the functional strength of the modules.	It represents the independence among the modules.
When modules are highly cohesive, a high quality software is built.	When the modules are loosely coupled, it results in a high quality software.
In cohesion, module focuses on the single thing.	The modules would be connected to each other.

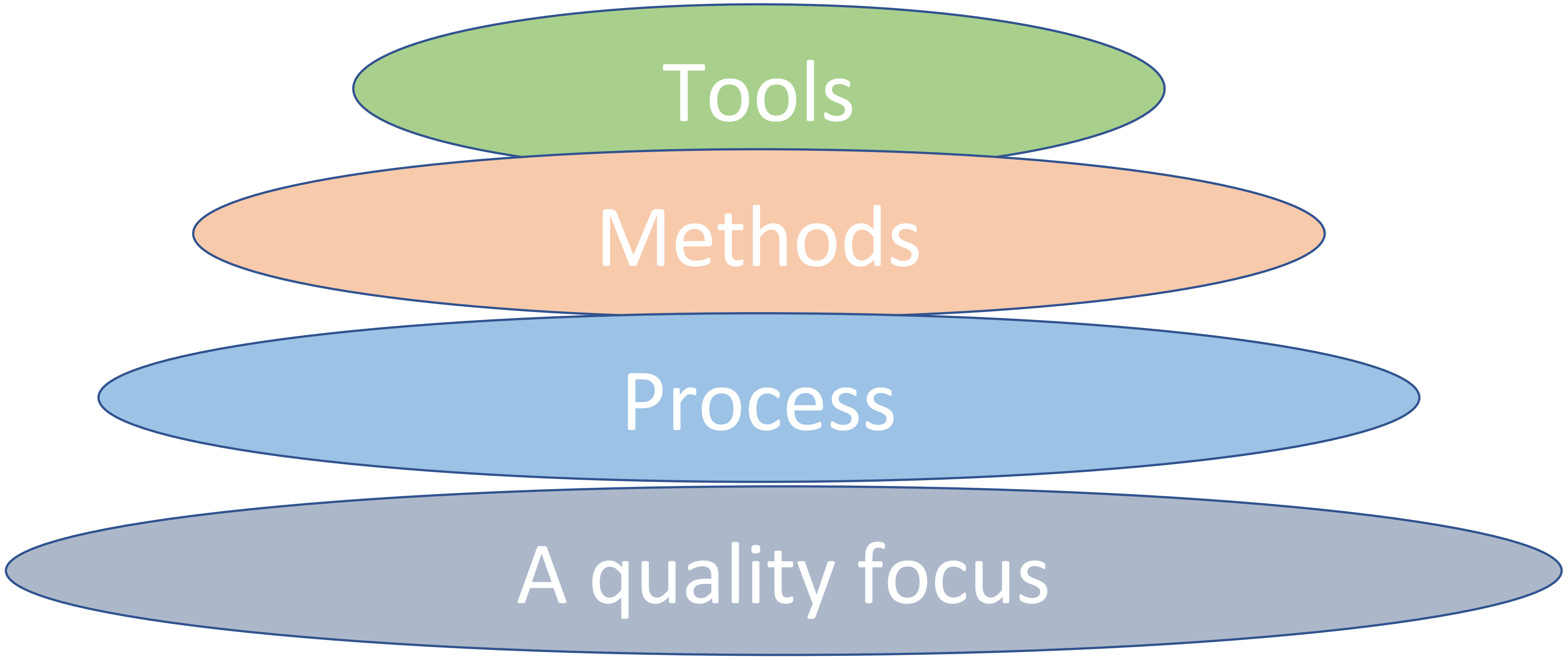
Why software components?

- Software Industry emerged as Global Market.
- Rapid demands for new Softwares in least time and cost.
- Plug and Play concept.
- Maximize reusability to maintain business profitability.

Software engineering

- According to IEEE "*software is a collection of computer programs, procedures, rules and associated documentation and data pertaining to the operation of a computer system."*
- ***SE encompasses a process, a collection of methods [practice] and an array of tools that allow professionals to build high-quality computer software***
- ***a discipline whose aim is the production of quality s/w, s/w that delivered on time, with the budget and that satisfies its requirements***

Software engineering layers



- A Quality focus:

SE is a layered technology, any engineering approach must rest on a organizational commitment to quality . Eg TQM, six sigma etc will foster a continuous process improvement culture

- Process: the foundation for se is a process layer and forms the basis for magt control of software projects- framework, methods, work products creation(models , documents, data ,reports, forms etc), milestones, quality and change is properly managed
- Methods:

It will provide technical how-to’s for building s/w- basic principles, tasks, communication, requirement analysis, design modelling, prg construction, testing and support.

- Tools:

provide automated and semi-automated support for the process and methods, when tools are integrated so that information created by one tool can be used by another, a system for the support of software development, is called as computer-aided-se

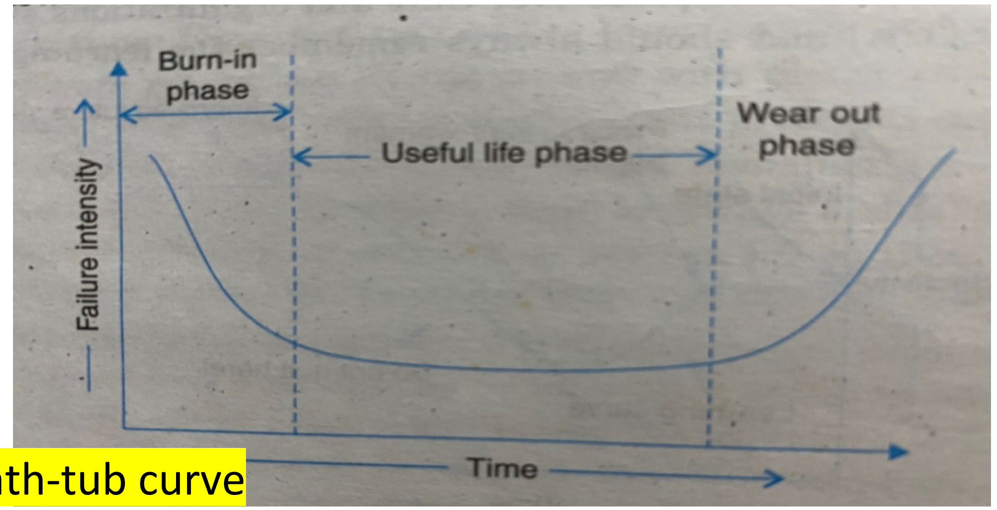
Software Characteristic

1. *Software is developed or engineered but it is not manufactured in the classical sense*

2. *Software doesn't wear out*

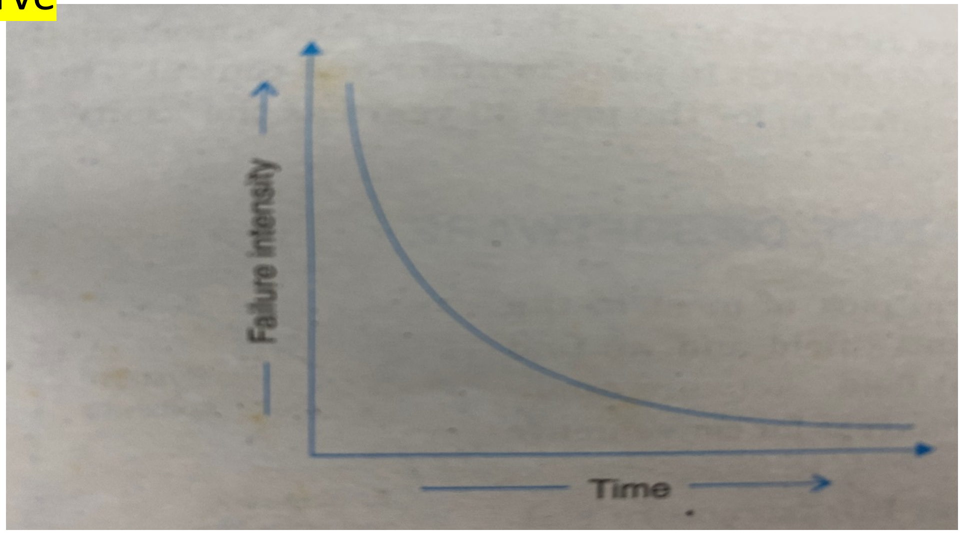
3. *Reusability of components*

4. *s/w is flexible*



Bath-tub curve

software curve



Software Application

1. System Software
2. Real-Time software
3. Business Software
4. Engineering and Scientific Software
5. Embedded Software
6. Personal Computer Software
7. Artificial Intelligence Software
8. Web based software

The Evolving Role of Software

- The following are the different eras’ of software engineering:

1. **The Pioneering Era (1955-1965)**
2. **The Stabilizing Era (1965-1980)**
3. **The Micro Era (1980-Present)**

1. The Pioneering Era (1955-1965)

- **As new computer systems were introduced very frequently. Software people had to rewrite all their programs to run on these new machines.**

- **Using punched cards** for input.

- The field was so new that **the idea of management by schedule was non-existent.**

- Making **predictions of a project's completion date was almost impossible.**

1. The Pioneering Era (1955-1965)

- hardware was application-specific
- Hardware vendors gave away **systems software for free** as hardware could not be sold without software.

A few companies sold the service of building custom software but no software companies were selling packaged software.

2. The Stabilizing Era (1965-1980)

- **turnaround time**, the time between job submission and completion. At worst it was measured in days.

- Then came IBM 360. It signaled the beginning of the stabilizing era.

2. The Stabilizing Era (1965-1980)

- The job control language (JCL) raised a whole new class of problems.
- "Structured Programming" burst on the scene in the middle of this era.
- PL/I, introduced by IBM to merge all programming languages into one, failed.
- Most customized applications continued to be done in-house.

PL/I (Programming Language One, pronounced /written **PL**/1) is a procedural, imperative computer programming language developed and published by IBM.

3. The Micro Era (1980-Present)

- The price of computing has dropped dramatically making **ubiquitous computing** possible.
- Now every programmer can have a computer on his desk. The old JCL has been replaced by the user-friendly GUI.
- Job Control Language(**JCL**) is a scripting language that describe jobs, to the Operating System that runs in the IBM large server(Mainframe) computers. **JCL** acts as an interface between your application programs (COBOL, PL/1 , Assembler etc) and Mainframe OS

Ubiquitous computing (or "ubicomp") is a concept in software engineering and **computer** science where **computing** is made to appear anytime and everywhere

3. The Micro Era (1980-Present)

- **The most-used programming languages today are between 15 and 40 years old.**

- **The Fourth Generation Languages never achieved the dream of "programming without programmers”**

Software myths

- myth means- wrong belief or mis information
- s/w myth means

" s/w myths are belief about s/w and the process used to build it "

- The myths have number of attributes that causes serious problems on s/w
- 3 types of myths:
  - Management myths
  - Customer myths
  - Practitioner myths

Management Myths

s/w myth believed management think about s/w development

1. **We already have a book that's full of standards and procedures for building software.[standard tools are present and they think they are sufficient for developers]**

2. **If your behind schedule –they think, we can add more programmers to catch up**

3. **They think they have latest computers**

4. **A good manager can manage any projects**

- **Customer Myths**
- **What cust always thinks/   s/w myth believed by cust who can be internal/external**

- ***A general statement of objectives is sufficient to start writing coding***

- ***Changing the requirements is easy because he thinks s/w is flexible***

- ***The cust always think that s/w development is an easy process***

Practitioner’s / developer Myths

What developer think

- ***If I mis something now, I can fix it later (if error come , will think)***
- ***Once the program is written and running , my job is done / over***
- ***Until a program is running, there is no way of assessing its quality***
- ***The only deliverable for s/w project is working program***

SEI-CMM

- Software engineering institution –Capability Maturity Model
- Helped organisation to improve the quality of the s/w they developed
- Adoption of SEI-CMM has significant business benefits
- Used for 2 ways
  - Capability evaluation : provides a way to access the s/w process capability
  - S/w process assessment: it is used by an organisation with the objective to improve process capability
- SEI-CMM classifies s/w development industries into 5 maturity levels

- Level 1:   initial level

for a s/w development, organisation at this level is characterized by adhoc activities

- Level 2:   repeatable

At this level, the basic project management practices, such as tracking cost and schedule are established

- Level 3:   defined

At this level processes for both management and development activities are defined and documented

- Level 4:   managed

At this level, the focus is an s/w metrics(ups n downs)

- Level 5:   optimizing

At this stage, processes and product metrics are collected

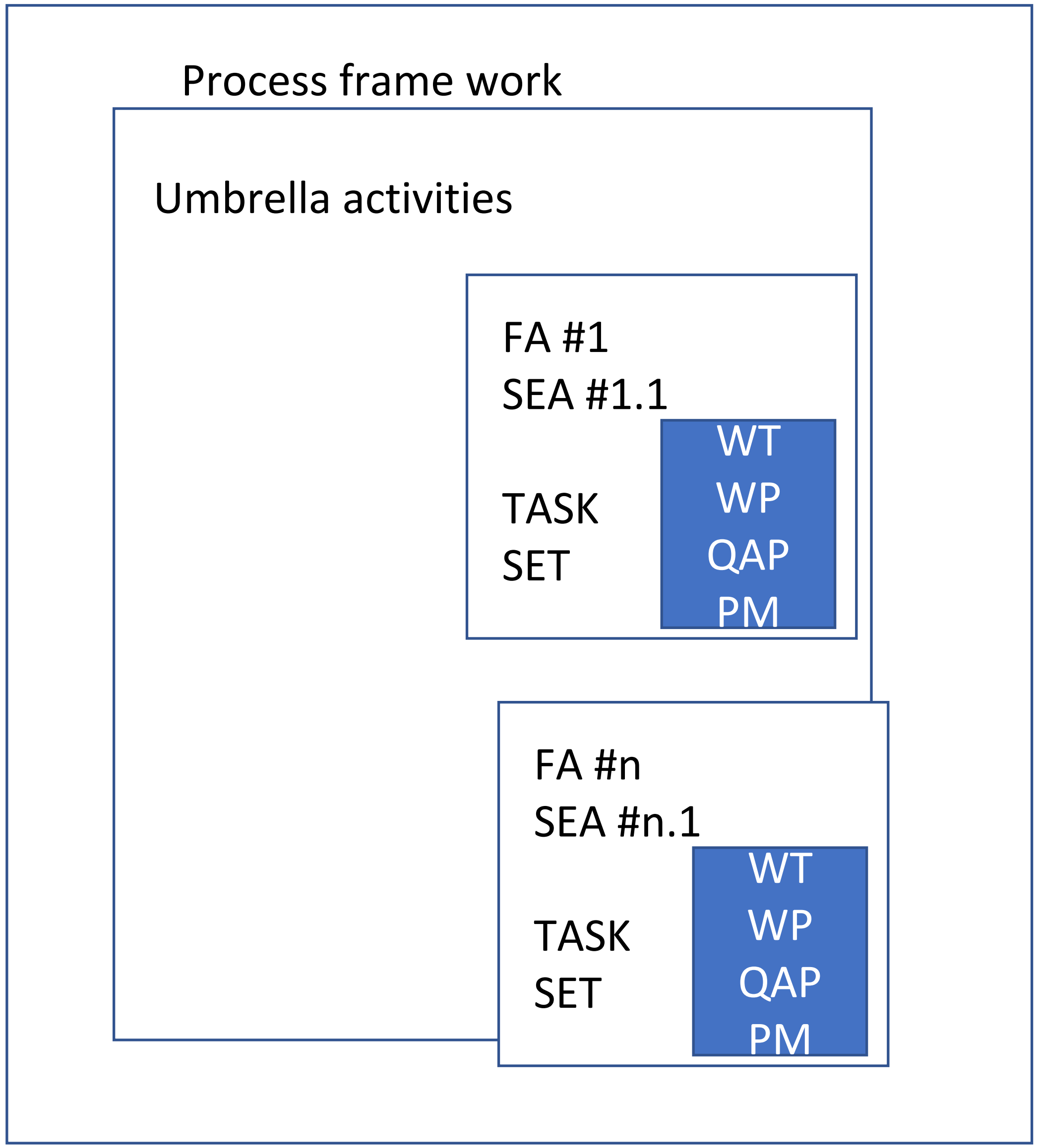
Software process

Software Engineering Processes

1. **A process is a collection of activities, actions and tasks that are performed when some work product is to be created.**

- Activity: strives to achieve a broad objective (eg communication with stakeholders) and applied regardless of the application domain, size of the project, complexity of the effort, or degree of rigor with which se is to be applied
- Action: (architectural design) encompasses a set of tasks that produce a major work product.
- Tasks: focuses on a small , but well-defined objective (conducting a unit test) that produces a tangible outcome

This differs from organization to organization, a *process framework* establishes the foundation for a complete se process by identifying a small number of framework activities that are applicable to all s/w projects.It is a set of umbrella activities applicable to entire s/w process.



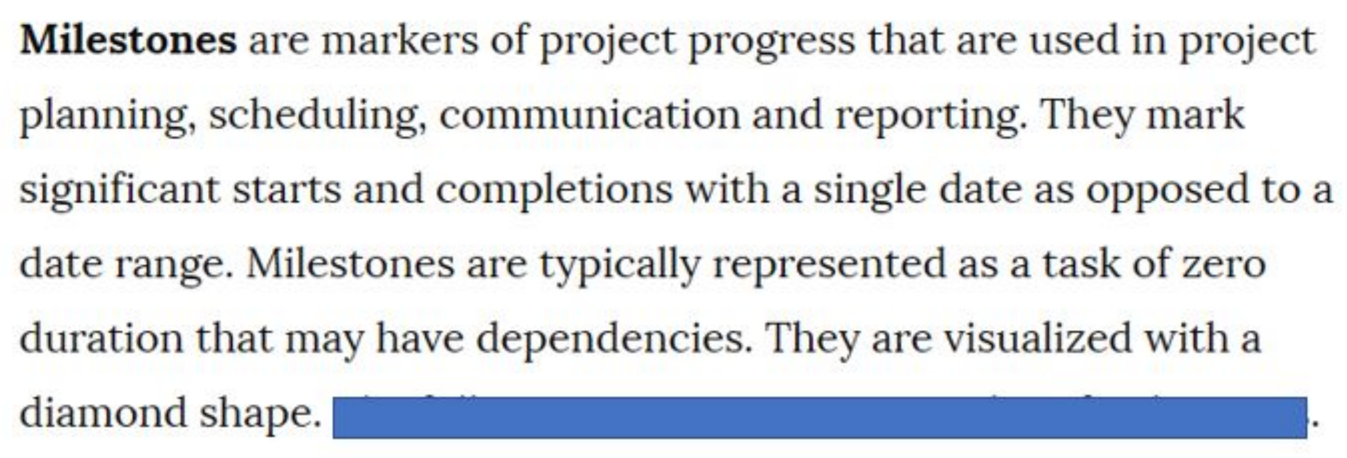
PF	PROCESS FRAMEWORK
FA	FRAME WORK ACTIVITY
Sea	SE activity
WT	WORK TASK
WP	WORK PRODUCTS
QAP	QUALITY ASSURANCE POINTS
PM	PROJEC MILSTONES FOR PROGRESS

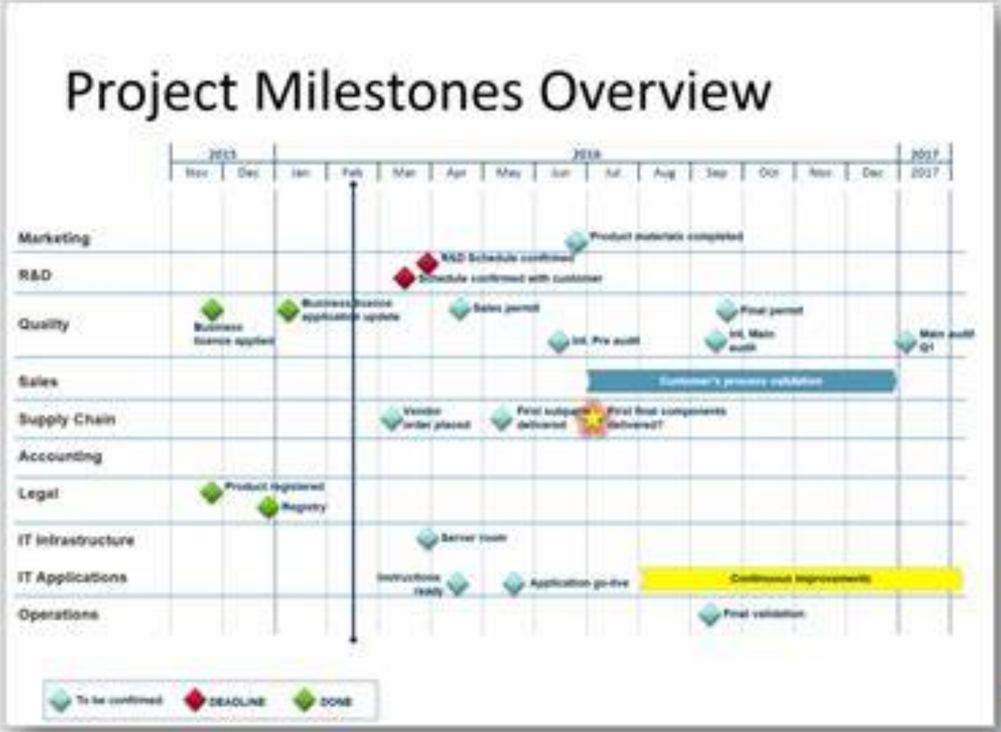
UMBRELLA ACTIVITIES

THE ACTIVITIES WHICH ARE APPLIED THROUGH THE S/W PROCESS /PROJECT

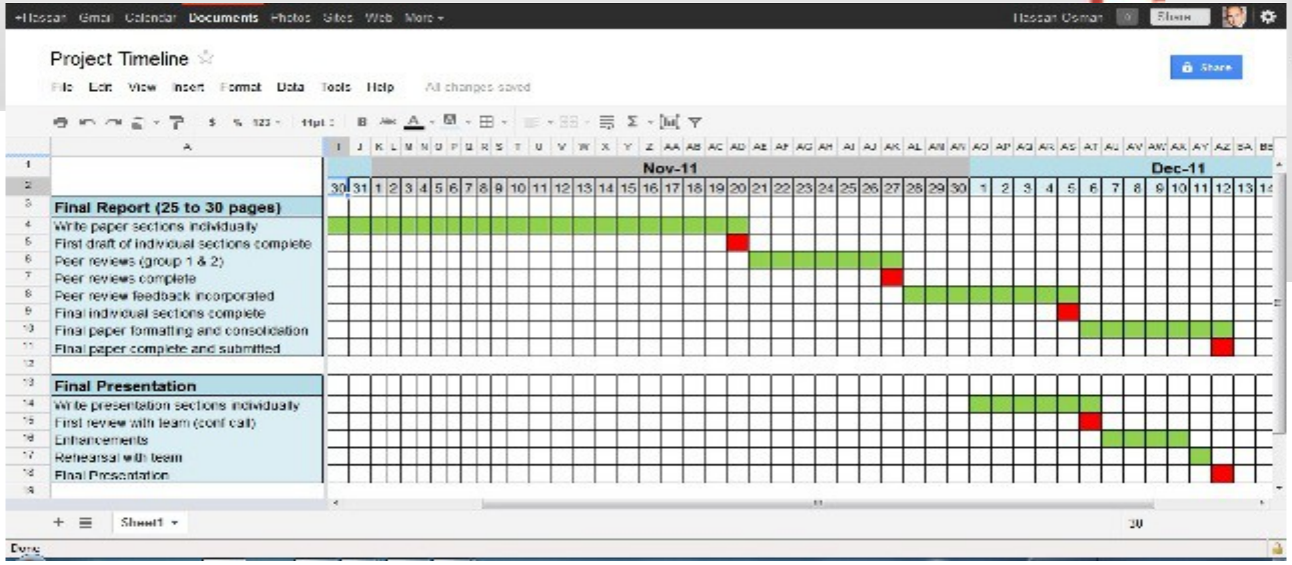
project milestone

- A project milestone marks a significant point in time in a project’s life cycle. It could be the start or end of a project. Or it could note the completion of an important phase.
- When a project reaches this point in time, the milestone is achieved.


Milestones are markers of project progress that are used in project planning, scheduling, communication and reporting. They mark significant starts and completions with a single date as opposed to a date range. Milestones are typically represented as a task of zero duration that may have dependencies. They are visualized with a diamond shape. [redacted].



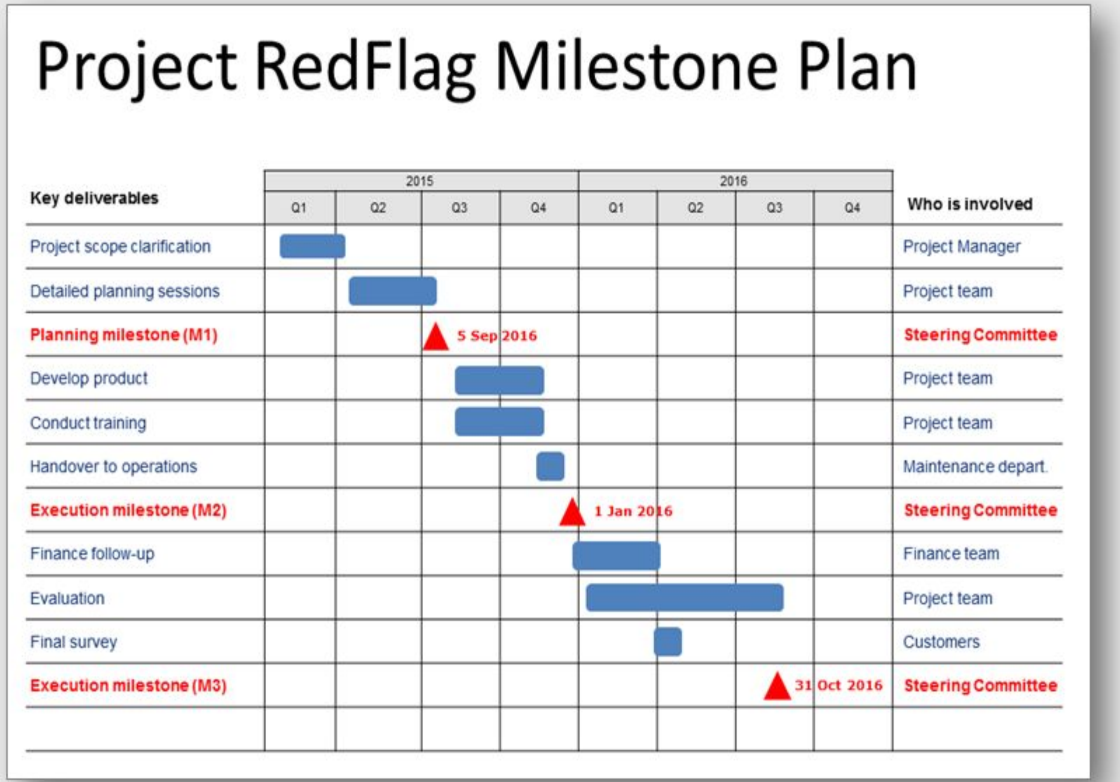
Project Milestones Overview



Project Timeline

Final Report (25 to 30 pages)
- Write paper sections individually
- First draft of individual sections complete
- Peer reviews (group 1 & 2)
- Peer reviews complete
- Peer review feedback incorporated
- Final individual sections complete
- Final paper formatting and consolidation
- Final paper complete and submitted

Final Presentation
- Write presentation sections individually
- First review with team (conf call)
- Enhancements
- Rehearsal with team
- Final Presentation



Project RedFlag Milestone Plan

Key deliverables	Milestone	Who is involved
Project scope clarification		Project Manager
Detailed planning sessions		Project team
Planning milestone (M1)	5 Sep 2016	**Steering Committee**
Develop product		Project team
Conduct training		Project team
Handover to operations		Maintenance depart.
Execution milestone (M2)	1 Jan 2016	**Steering Committee**
Finance follow-up		Finance team
Evaluation		Project team
Final survey		Customers
Execution milestone (M3)	31 Oct 2016	**Steering Committee**

Software Engineering Processes generic Framework Activities

- *Communication   [communicate and collaborate all stakeholders, define objective,all features and functions]*
- *Planning [any complicated journey can be simplifying if a map exists-software project plan]*
- *Modeling [creating models to better understand s/w requirements and design that will achieve those requirements]*
- *Construction [combines code generation (manual /auto) and the testing that is required to uncover errors in the code]*
- *Deployment [delivered to the cust and feedback based on the evaluation]*

- **Project Documentation by Project Phase**
- Projects vary in size and complexity and some require more comprehensive project documentation than others. Here's an overview of some of the project documents that are usually created throughout the project life cycle.
- **Project Initiation:** Project charter, project summary, business case, project kickoff meeting agenda
- **Project Planning:** Project management plan, work breakdown structure, project budget, project schedule, change management plan, scope management plan, risk management plan
- **Project Execution:** Project status report, project execution plan lessons learned template, timesheets, change requests, change orders
- **Project Monitoring and Control:** Project status report, lessons learned template, timesheets
- **Project Closure:** Project closure template, punch list

Project documentation

Work breakdown structure

Task organization tool

Project charter

Foundational document

Stakeholder register

Identifies project stakeholders

Risk register

Catalogs potential risks

Project plan

Comprehensive roadmap

Issue log

Tracks project issues

Minutes

Records project meetings

Scope statement

Defines project boundaries

Communication Plan

Information sharing strategy

SE process -Umbrella activities

Umbrella activities are applied throughout a s/w project and help a s/w team manage and control progress, quality change , and risk

Umbrella activities span all the stages of the SDLC. They are not specific to any particular life cycle stage.

- ✔ s/w project tracking and control [schedule, budget etc]
- ✔ Risk management [assess the risk on the o/p]
- ✔ s/w quality assurance [activities , define - good and maintained the quality]
- ✔ Technical reviews [uncover errors and remove them before next phase]
- ✔ Measurement [process, project, product]
- ✔ s/w configuration management [to manage effects of the changes in the projects]
- ✔ Reusability management [Reusability management: **Define the standards for the reuse of work products (including software components), and develop mechanisms to implement reusable components**. This includes the approval of any part of a backing-up software project or any type of support provided for updates or updates in the future.]
- ✔ Work product preparation and production [models , doc, logs, lists etc]

software crisis

software crisis

early days: 1970s large no. of s/w projects failed mostly due to human factors. This problem was referred to as the “s/w crisis”

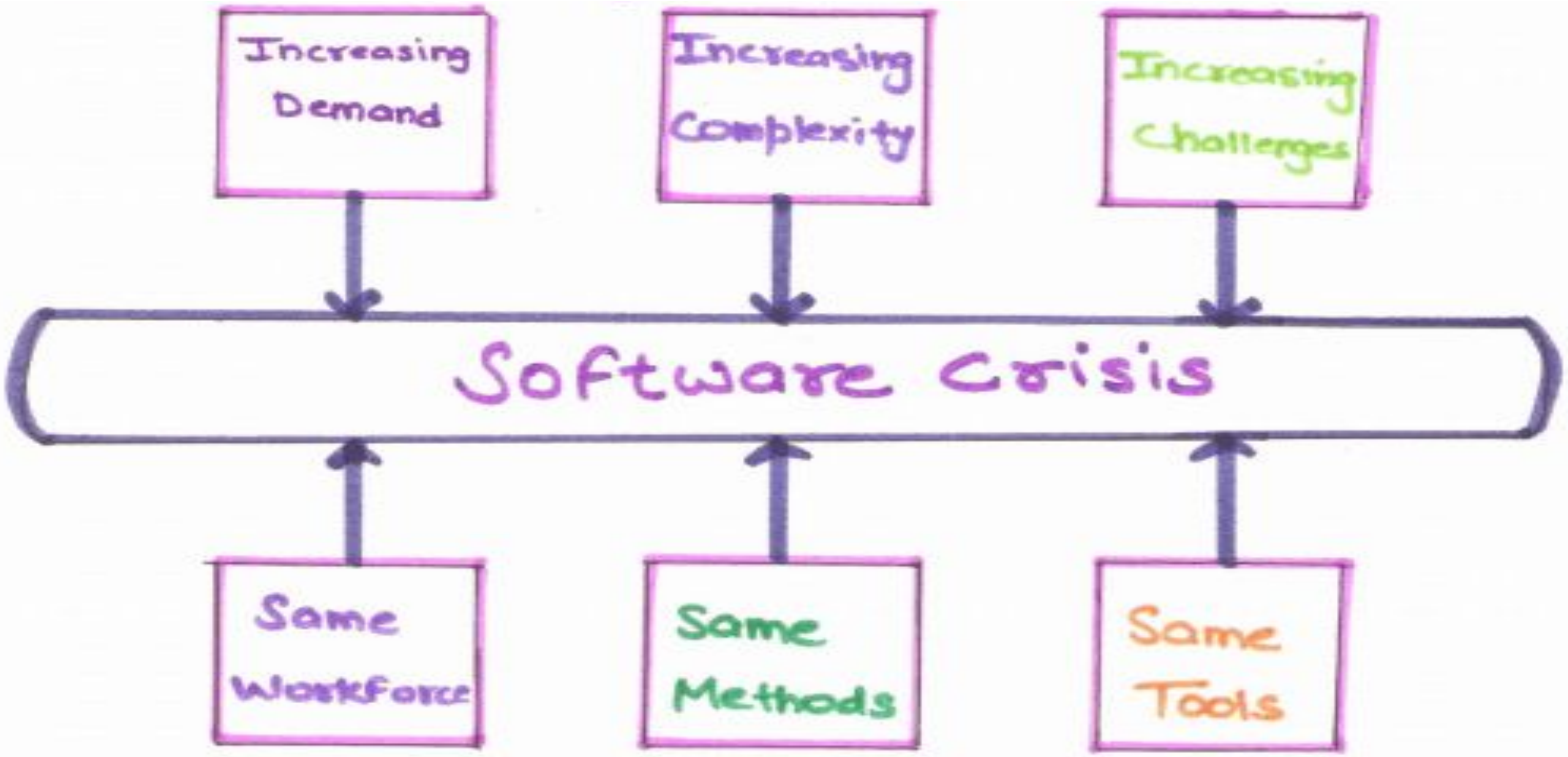
- There were many difficulties in the development of large software systems during the 1960s and 1970s.
- The term “software crisis” dates from that time.
- The problems stemmed from an inability to apply the techniques used to build small software systems to the development of larger and more complex systems. The typical way to develop small systems can be described as “code-and-fix”.

- The term used in early days of computing science for the difficulty for writing useful and efficient program in required time

Causes of s/w crisis:

- ✓ If the projects are Running over budget
- ✓ If the projects are over time
- ✓ s/w was very inefficient
- ✓ s/w of low quality
- ✓ s/w often didn’t meet requirements

- Major projects are meaningfully late.
- Software costs more than predicted.
- Software is unreliable.
- Software is difficult to maintain.
- Poor performance.
- While hardware costs were decreasing, software costs were rising.



That is Beginning of SOFTWARE ENGINEERING

The only solution for the s/w crisis is SE-

- ✓ systematic approach to s/w development
- ✓ Application of engg principles
- ✓ Provides a set of methods, tools and procedures

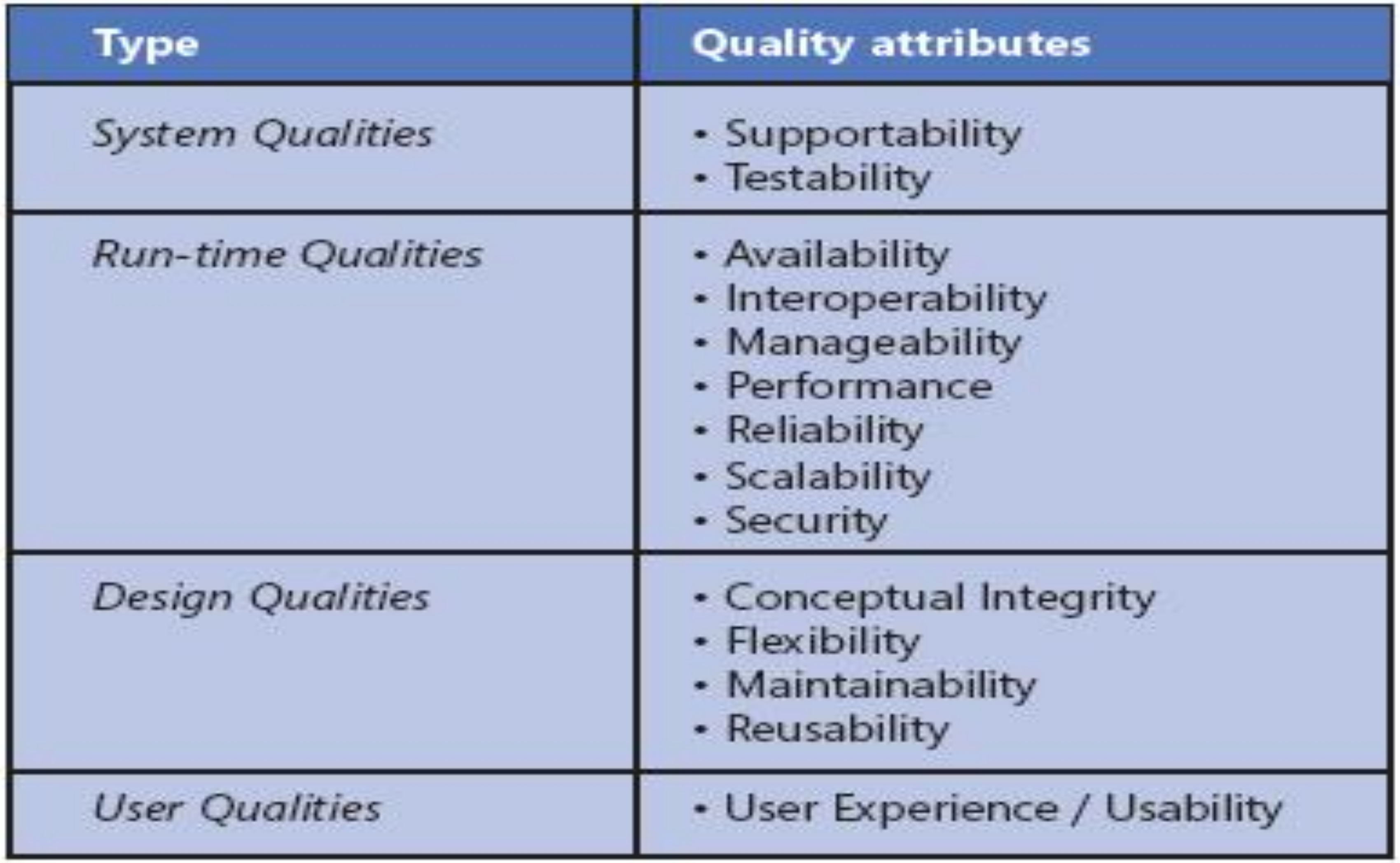
Software Quality Attributes]

Software Quality Attributes

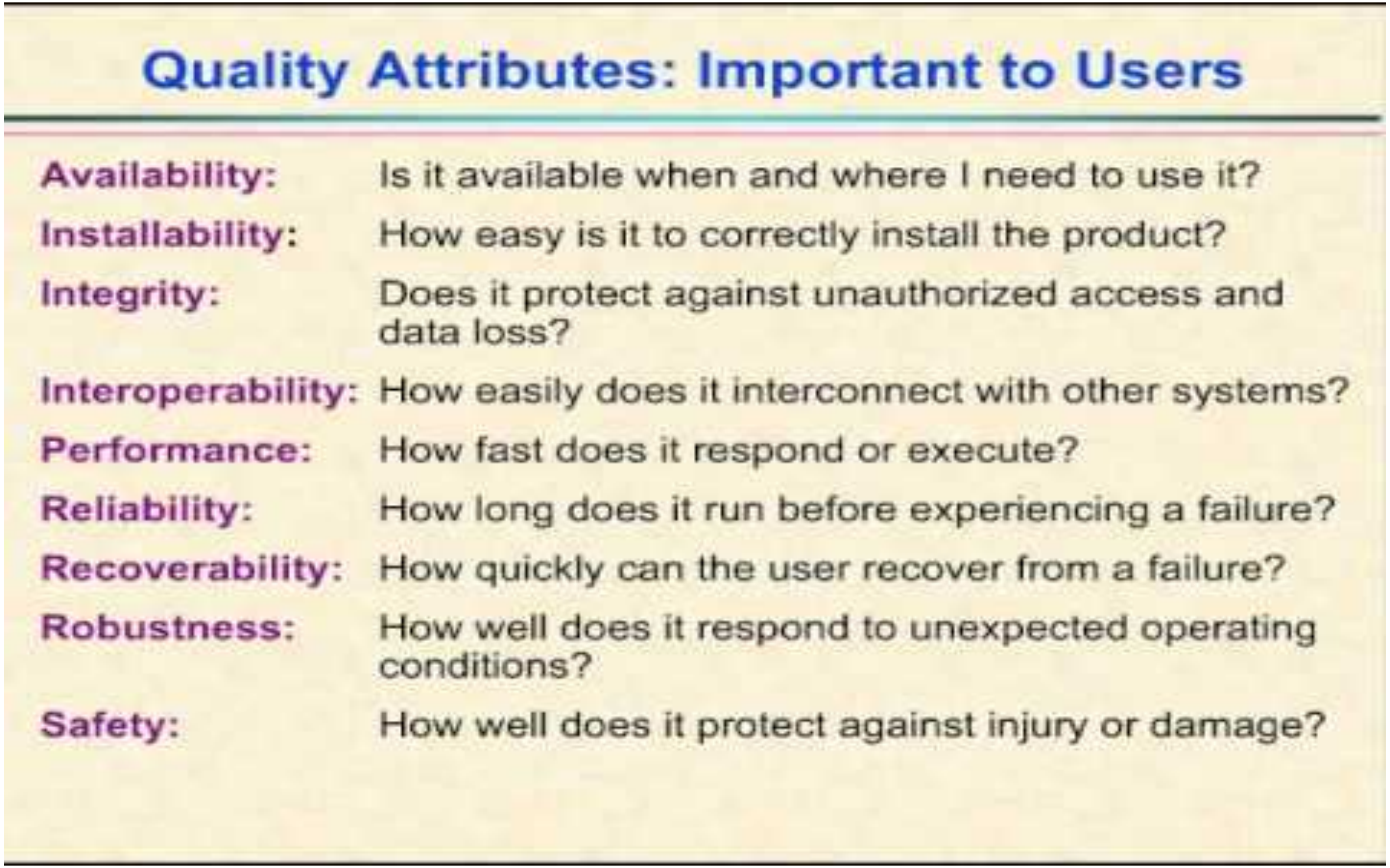
- A quality product does exactly what the user want it to do
- Software Quality Attributes are features that facilitate the measurement of performance of s/w product by Software Testing professionals, and include attributes such as availability, interoperability, correctness, reliability, learnability, robustness, maintainability, readability, extensibility, testability, efficiency, and portability.
- High scores in **Software Quality Attributes** enable software architects to guarantee that a software application will perform as the specifications provided by the client.

- Portability
- Usability
- Reusability
- Correctness
- maiantability

Software Quality Attributes [SRDU]



Type	Quality attributes
System Qualities	• Supportability • Testability
Run-time Qualities	• Availability • Interoperability • Manageability • Performance • Reliability • Scalability • Security
Design Qualities	• Conceptual Integrity • Flexibility • Maintainability • Reusability
User Qualities	• User Experience / Usability

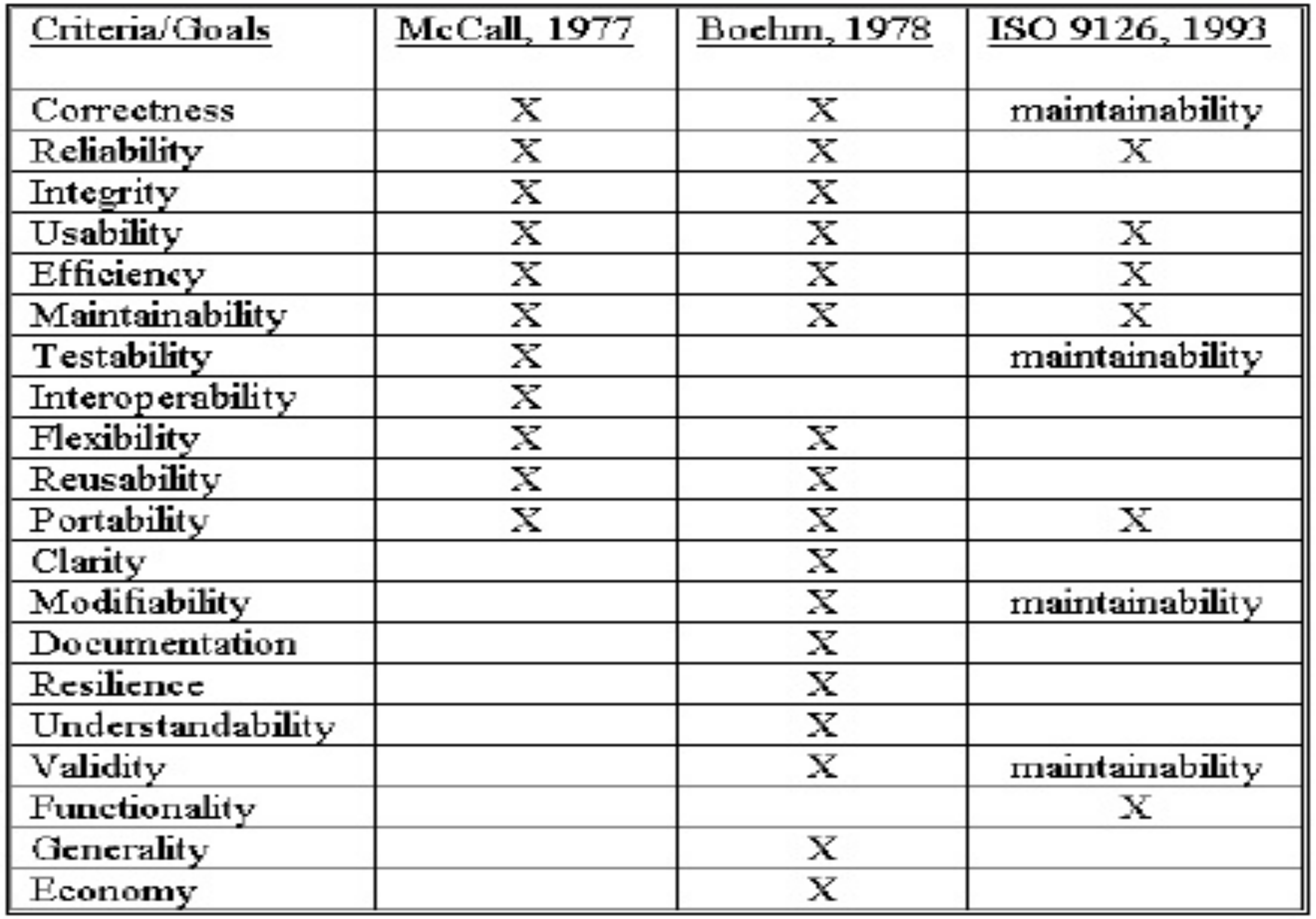


Quality Attributes: Important to Users

Availability:	Is it available when and where I need to use it?
Installability:	How easy is it to correctly install the product?
Integrity:	Does it protect against unauthorized access and data loss?
Interoperability:	How easily does it interconnect with other systems?
Performance:	How fast does it respond or execute?
Reliability:	How long does it run before experiencing a failure?
Recoverability:	How quickly can the user recover from a failure?
Robustness:	How well does it respond to unexpected operating conditions?
Safety:	How well does it protect against injury or damage?

Software Quality Attributes

http://satc.gsfc.nasa.gov/support/STC_APR96/qualtiy/stc_qual.html



Criteria/Goals	McCall, 1977	Boehm, 1978	ISO 9126, 1993
Correctness	X	X	maintainability
Reliability	X	X	X
Integrity	X	X	
Usability	X	X	X
Efficiency	X	X	X
Maintainability	X	X	X
Testability	X		maintainability
Interoperability	X		
Flexibility	X	X	
Reusability	X	X	
Portability	X	X	X
Clarity		X	
Modifiability		X	maintainability
Documentation		X	
Resilience		X	
Understandability		X	
Validity		X	maintainability
Functionality			X
Generality		X	
Economy		X	

questions

- Define term S/w engineering and explain s/w process
- What is s/w crisis? Was y2k a s/w crisis
- How are s/w mlyths affecting s/w process? Explain with the help of examples
- Diff
  - s/w components and components of s/w
  - Deliverables and milestones
  - Measures , metrics and measurement

ANY QUESTIONS