Software Engineering

Sankita Patel

sjp@coed.svnit.ac.in

Reference Books

RECOMMENDED READING :

1. Ian Sommerville, “Software Engineering”.
2. Pankaj Jalote, “An Integrated approach to Software Engineering”
3. Ghezzi, Jazayeri, Mandrioli, “Fundamentals of Software Engineering”
4. Stephen R Schach, “Software Engineering with JAVA”
5. Rajib Mall, Software Engineering.

An Introduction to Software Engineering

- Let us start with a question[P. Jalote]

You are given a problem for which you have to build a software system that most students feel will be approximately 10,000 LOC. If your are working full time on it, how long will it take you to build this system ?

- What could be your answer ???
  - Perhaps 2 to 3 months
  - What can you say about productivity?
  - If completion time is 2 months, we can say that the productivity is 5000 LOC per person-month

Now, let us take an alternative scenario – we act as clients and pose the same problem to a company that is in the business of developing software for clients

What would you say about the productivity?

Now, let us take an alternative scenario – we act as clients and pose the same problem to a company that is in the business of developing software for clients

What would you say about the productivity?

- it is fair to say a productivity figure of 1,000 LOC per person-month !!!!

Why this difference in two scenarios ????

The Problem Domain

- Student System
  - Built to illustrate something or for hobby
  - Not for solving any real problem
- Industrial Strength Software [P. Jalote]
  - Solves some problem of some users where larger systems or businesses may depend on the software
  - Where problems in the software can lead to significant direct or indirect loss

Student system versus Industrial Strength Software

- Usually small in size
- Author himself is sole user
- Single developer
- Lacks detailed requirements
- Lacks proper user interface
- Lacks proper documentation
- Less efforts in testing
- Ad hoc development

- Large
- Large number of users
- Team of developers
- Detailed requirements desirable
- Well-designed interface
- Well documented & user-manual prepared
- Rigorous testing
- Systematic development

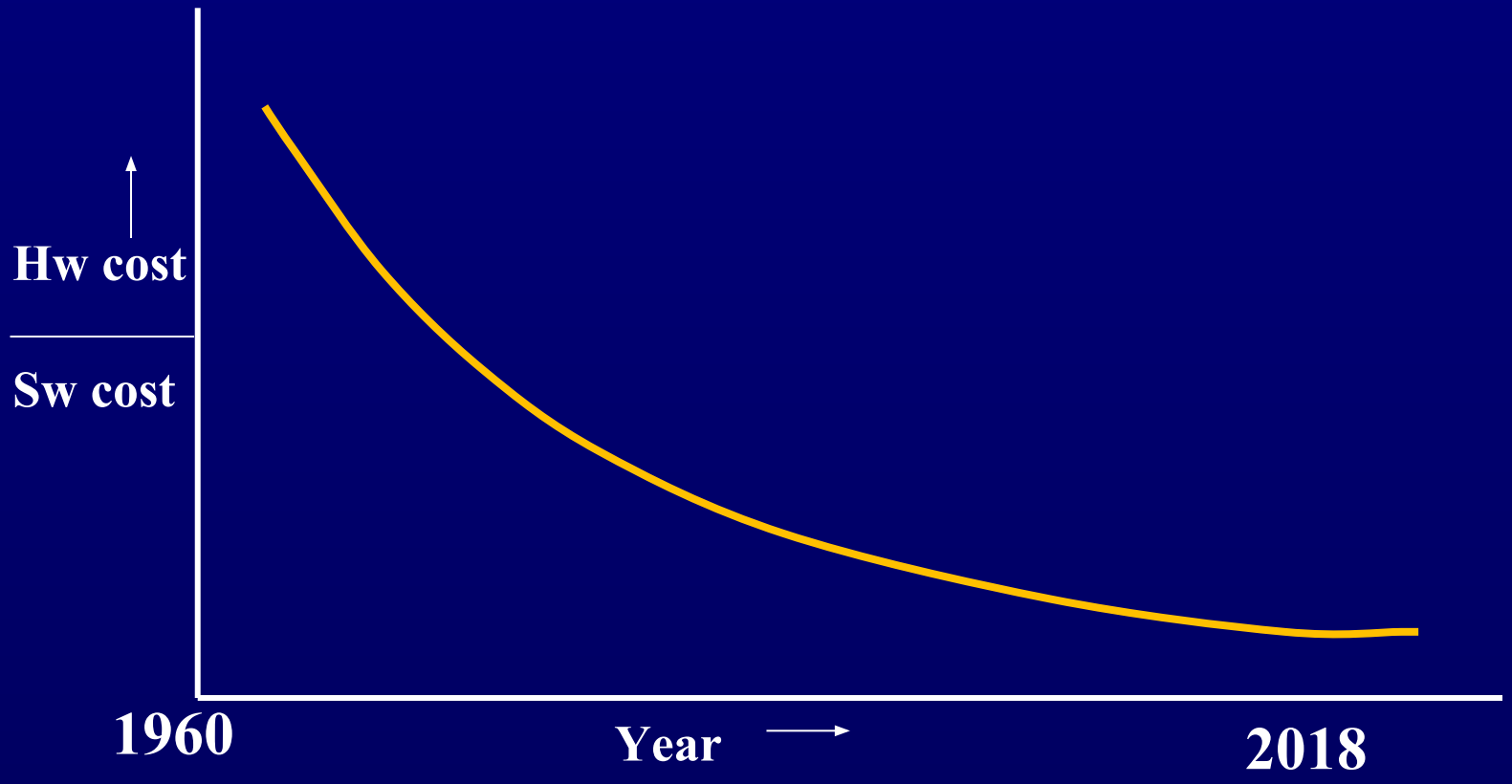
IEEE definition of Software

Collection of computer programs, procedures, rules, and associated documentation and data.

Software is Expensive !!

- Primarily due to the fact that the software development is extremely labor-intensive
  - Consider the cost of 50,000 LOC software !!!!
    - Let us consider the productivity of 1000 LOC per person/month
    - Company charges client $8000 per person per month
    - Comes out to be $0.5 million !!!!!
    - Now compare this with the cost of hardware on which this software is deployed
- Software forms the major component of the total automated system with the hardware forming a very small component !!!!

Software is Expensive(cont.)



Relative Cost of Hardware and Software

Software is Late and Unreliable !!

- Software development remains a weak area despite the considerable progress
- More than 35% projects in firms are runaway
- Software does not do what it is supposed to do ... or vice versa
  - Often the software is the weakest component among all....
    - In one defence survey, it was reported that, more than 70% of all the equipment failures were due to software!
    - Failure of an early Apollo flight was attributed to software!
    - Failure of a test firing of a missile in India was attributed to software!
  - What is the reason behind this ????

Software is Late and Unreliable(cont.)

- Failure of the electrical or mechanical systems is due to the aging!!!!!
- Does software wear out due to age???
  - Failures occur due to the bugs introduced during the design and development process.

Maintenance and Rework !!

- Why is the maintenance needed for software, when software does not age???
- Due to residual errors ....
  - Corrective maintenance
- Even without errors..the software undergo changes..
  - Due to up gradation and enhancement
  - Adaptive maintenance
- Changes during software development itself
  - Leads to rework
  - Due to lack of complete requirements.
  - Due to changing needs of clients

Maintenance and Rework !!

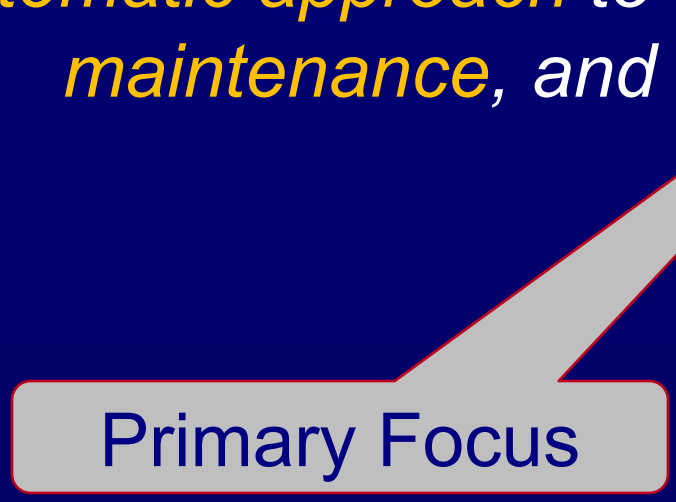
- Maintenance-to-development cost ratio is 80:20, 70:30 or 60:40 !!
- Why this much cost for maintenance???
- Maintenance involves
  - Understanding existing software
  - Understanding the effects of change
  - Making the changes
  - Testing the new parts
  - Retesting the old parts that were not changed (Regression testing)

Maintenance and Rework !!

The complexity of the maintenance task, coupled with the neglect of maintenance concerns during the development, makes maintenance the most costly activity in the life of a software product....

Software Engineering : definition

Software Engineering is defined as the systematic approach to the development, operation, maintenance, and retirement of software.



Software Engineering : definition...

- Systematic approach
  - Methodologies used are repeatable
  - To take the software development close to science and engineering and away from adhoc processes
- The fundamental problem that software engineering deals with,
  - Software satisfies user needs.....

Software Engineering: Challenges

- Scale
  - Methods that are used for development of small systems do not scale up for large systems
  - Less management of small projects, more for large projects...
  - Informal methods may mostly work !!!!
    - What about large projects??
  - How will you define small project and large project ???

Software Engineering: Challenges(cont.)

- Scale(cont.)
  - How will you define small project and large project ???
    - Small - size <10 KLOC
    - Medium - 10 KLOC < size < 100 KLOC
    - Large - 100 KLOC < size < 1 MLOC
    - Very large - many MLOC
  - A table to have an idea of software sizes...

Size in KLOC of some well known products [P. Jalote]

Size(KLOC)	Software
980	Gcc
320	Perl
305	teTex
200	Openssl
200	Python
100	Apache
65	Sendmail
45	Gnuplot
38	Openssh
30,000	Red hat linux
40,000	Windows XP

Software Engineering: Challenges(cont.)

- Quality and productivity
  - Software engineering is driven by three major factors
    - Cost,
    - Schedule, and
    - Quality
      - Functionality
      - Reliability
      - Usability
      - Efficiency
      - Maintainability
      - Portability

Software Engineering: Challenges(cont.)

- Consistency and Repeatability
  - To ensure that the successful results can be repeated
  - Needs some standards to be followed such as ISO9001 and CMM
- Change
  - A usual perception : software is easy to change !!!!

The Software Engineering Approach

For success in large software development, it is important to follow an engineering approach, consisting of a well-defined process

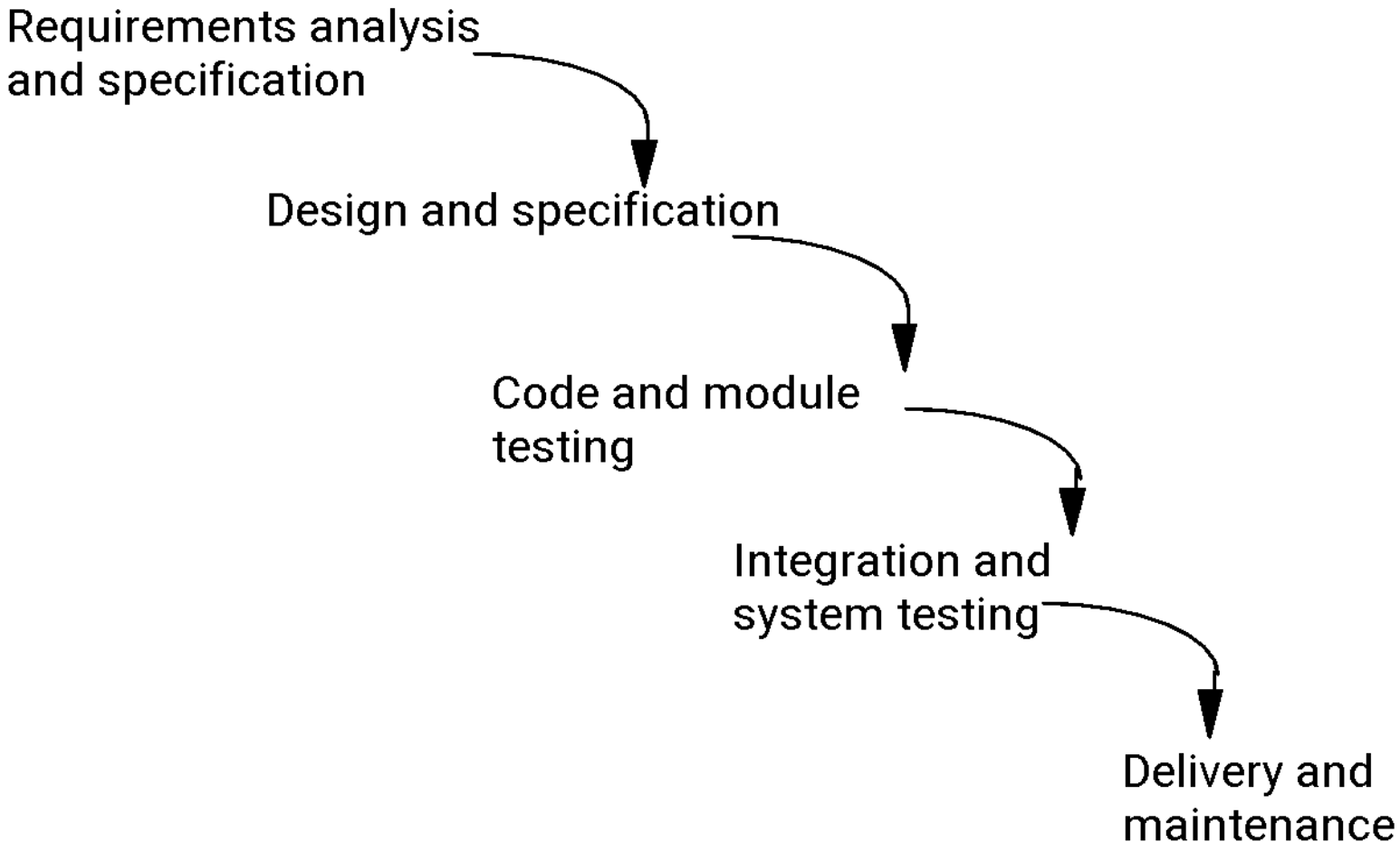
Software Process

- Main goal : High Q&P (Quality & Productivity) under the dynamics of change !!!!
- Three main factors that govern Q&P
  - People
  - Processes
  - Technology
- Key goal of software engineering research
  - Design of proper software processes and their control
  - Software Process : The process that deals with the technical and management issues of software development
- Process distinguish software engineering from most other computing disciplines

Software Process (cont.)

- Phased Development Process
  - Development process consists of various phases each phase ending with a defined output
  - Phases are performed in an order specified by the process model being followed
  - Cost of development is reduced .. Why ????
- Managing the Process

The software lifecycle (a preview)



Brooks' No Silver Bullet

- Tutorial assignment:
  - Read a paper, make a presentation of 10 minutes on it and present it in a next tutorial class.

Software Requirements

Reference:
Software Engineering By Sommerville (8th Edition)
Chapter 6-7

Outline

- Functional and non-functional requirements
- User requirements
- System requirements
- Interface specification
- The software requirements document

Requirements engineering

- Requirements:
  - The description of the services provided by the system and its operational constraints
- Requirement Engineering(RE):
  - The process of finding out, analysing, documenting and checking these services and constraints

What is a requirement?

- Requirement:
  - Can be a high-level abstract statement of a service
  - Can be a detailed, formal definition of a system function
- This is inevitable as requirements may serve a dual function
  - May be the basis for a bid for a contract - therefore must be open to interpretation;
  - May be the basis for the contract itself - therefore must be defined in detail;
  - Both these statements may be called requirements.

Types of requirement

- User requirements
  - High level abstract requirement
  - Statements in natural language plus diagrams of the services the system provides and its operational constraints.
- System requirements
  - Detailed description of what the system should do
  - A structured document setting out detailed descriptions of the system's functions, services and operational constraints.
  - Defines what should be implemented so may be part of a contract between client and contractor.

Definitions and specifications

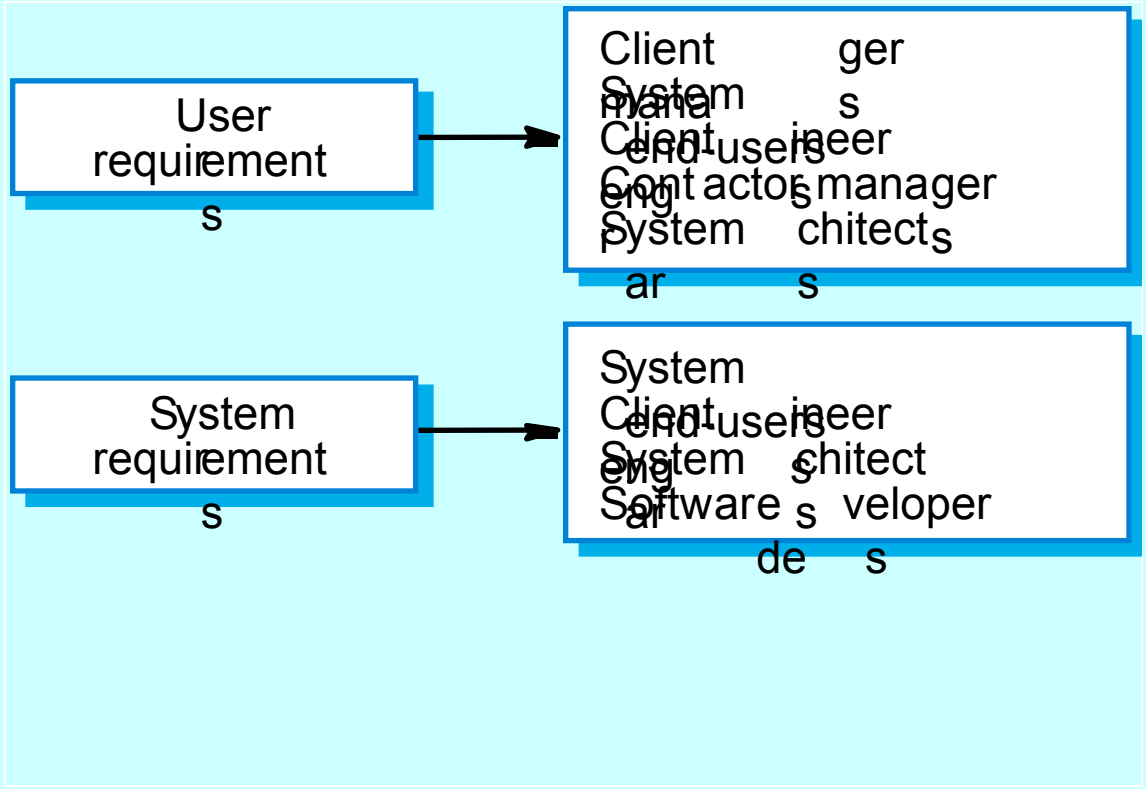
User requirement definition

1. The software must provide a means of representing and accessing external files created by other tools.

System requirements specification

1.1 The user should be provided with facilities to define the type of external files.
1.2 Each external file type may have an associated tool which may be applied to the file.
1.3 Each external file type may be represented as a specific icon on the user's display.
1.4 Facilities should be provided for the icon representing an external file type to be defined by the user.
1.5 When a user selects an icon representing an external file, the effect of that selection is to apply the tool associated with the type of the external file to the file represented by the selected icon.

Requirements readers



Functional and non-functional requirements

- Functional requirements
  - Statements of services the system should provide
  - How the system should react to particular inputs
  - How the system should behave in particular situations.
- Non-functional requirements
  - Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
  - Generally applied to the system as a whole.
- Domain requirements
  - Requirements that come from the application domain of the system and that reflect characteristics of that domain.

Functional requirements

- Describe functionality or system services.
- Depend on the type of software, expected users and the type of system where the software is used.
- Functional requirements
  - User requirements
    - May be high-level statements of what the system should do
  - System requirements
    - Should describe the system services in detail.

The LIBSYS system

- A library system that provides a single interface to a number of databases of articles in different libraries.
- Users can search for, download and print these articles for personal study.

Examples of functional requirements

1. *The user shall be able to search either all of the initial set of databases or select a subset from it.*
2. *The system shall provide appropriate viewers for the user to read documents in the document store.*
3. *Every order shall be allocated a unique identifier (ORDER_ID) which the user shall be able to copy to the account's permanent storage area.*

Requirements imprecision

- Problems arise when requirements are not precisely stated.
- Ambiguous requirements may be interpreted in different ways by developers and users.
- Consider the term ‘appropriate viewers’
  - User intention - special purpose viewer for each different document type;
  - Developer interpretation - Provide a text viewer that shows the contents of the document.

Requirements completeness and consistency

- Complete
  - Should include descriptions of all facilities required.
- Consistent
  - Should be no conflicts or contradictions in the descriptions of the system facilities.
- In practice, it is impossible to produce a complete and consistent requirements document
  - Mistakes and omissions are made while writing requirements for large systems
  - Different system stakeholders have inconsistent requirements

Non-functional requirements

- Define system properties and constraints e.g. Reliability, response time and storage requirements.
- Constraints are I/O device capability, system representations, etc.
- Process requirements may also be specified mandating a particular CASE system, programming language or development method.
- Non-functional requirements may be more critical than functional requirements.
  - If these are not met, the system is useless.
  - E.g. What if aircraft system doesn't meet reliability requirement???
  - E.g. What if a real-time control system fails to meet its performance requirement??

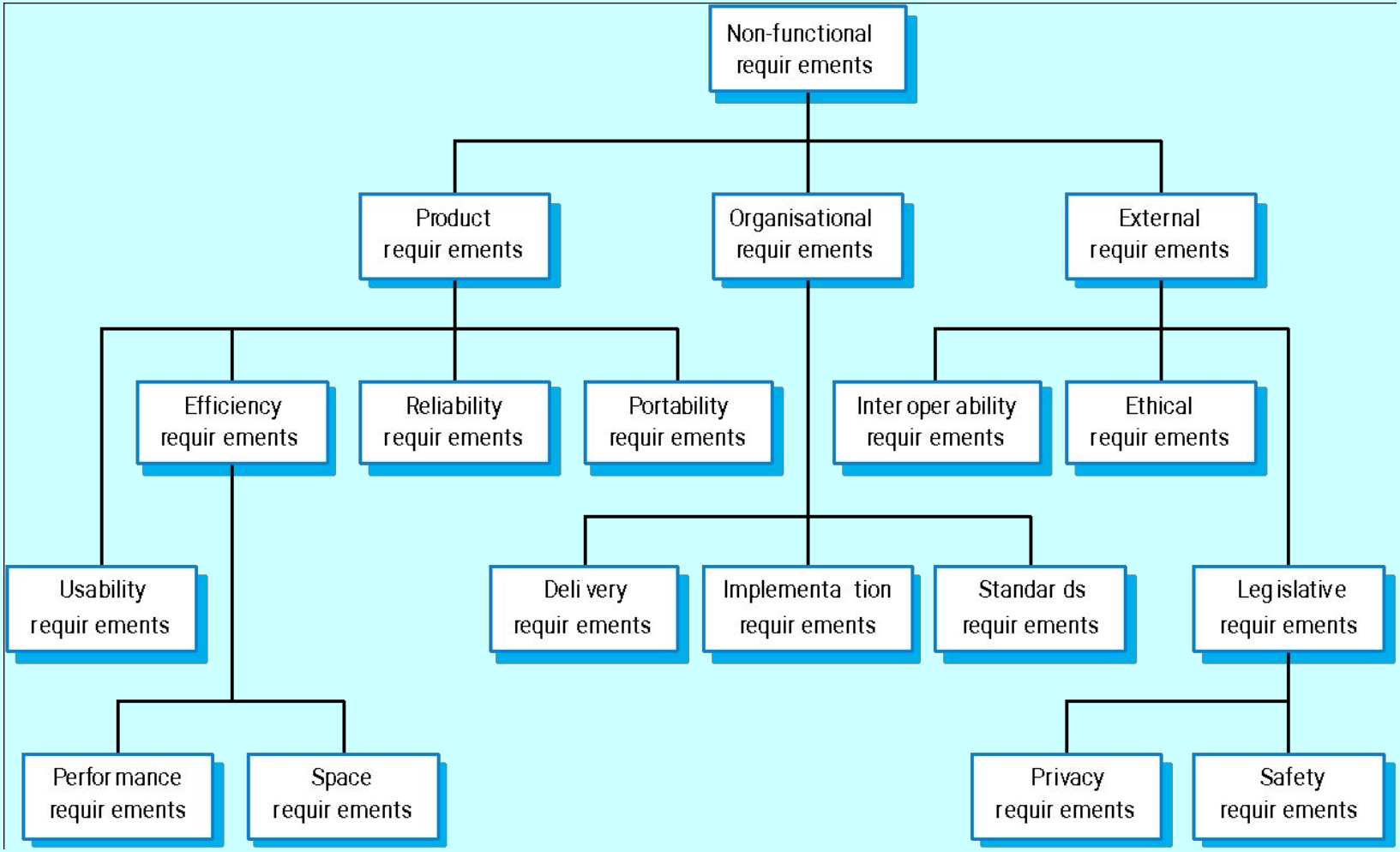
Non-functional classifications

- Product requirements
  - Requirements which specify that the delivered product must behave in a particular way e.g. execution speed, reliability, etc.
- Organisational requirements
  - Requirements which are a consequence of organisational policies and procedures e.g. process standards used, implementation requirements, etc.
- External requirements
  - Requirements which arise from factors which are external to the system and its development process e.g. interoperability requirements, legislative requirements, etc.

Non-functional requirements examples

- Product requirement
  - The user interface for LIBSYS shall be implemented as simple HTML without frames or Java applets.
- Organisational requirement
  - The system development process and deliverable documents shall conform to the process and deliverables defined in XYZCo-SP-STAN-95.
- External requirement
  - The system shall not disclose any personal information about customers apart from their name and reference number to the operators of the system.

Non-functional requirement types



Goal and Requirements

- Non-functional requirements may be very difficult to state precisely and imprecise requirements may be difficult to verify.
- Goal
  - A general intention of the user such as ease of use.
- Verifiable non-functional requirement
  - A statement using some measure that can be objectively tested.
- Goals are helpful to developers as they convey the intentions of the system users.

Examples

- A system goal
  - The system should be easy to use by experienced controllers and should be organised in such a way that user errors are minimised.
- A verifiable non-functional requirement
  - Experienced controllers shall be able to use all the system functions after a total of two hours training. After this training, the average number of errors made by experienced users shall not exceed two per day.

Requirements measures

Property	Measure
Speed	Processed transactions/second User/Event response time Screen refresh time
Size	M Bytes Number of ROM chips
Ease of use	Training time Number of help frames
Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target dependent statements Number of target systems

Domain requirements

- Derived from the application domain and describe system characteristics and features that reflect the domain.
- Domain requirements be new functional requirements, constraints on existing requirements or define specific computations.
- If domain requirements are not satisfied, the system may be unusable.

Library system domain requirements

- There shall be a standard user interface to all databases which shall be based on the Z39.50 standard.
- Because of copyright restrictions, some documents must be deleted immediately on arrival. Depending on the user's requirements, these documents will either be printed locally on the system server for manually forwarding to the user or routed to a network printer.

Domain requirements problems

- Understandability
  - Requirements are expressed in the language of the application domain;
  - This is often not understood by software engineers developing the system.
- Implicitness
  - Domain specialists understand the area so well that they do not think of making the domain requirements explicit.