Unit 1
The Software Problem

Software

IEEE defines Software as a collection of

- programs,
- procedures,
- rules, and
- associated documentation and data

Software Engineering

IEEE defines Software Engineering as:

The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software.

Proposed by Fritz Bauer

Software Engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.

Software

- Students build: **Student software**
- Industry builds: **Industrial Strength Software System**

- What is the difference between
  - a student software and
  - industrial strength software

  for the same problem?

S.NO.	STUDENT SOFTWARE SYSTEM	INDUSTRIAL STRENGTH SOFTWARE
1)	Developed for **demonstration** purpose only	Developed to solve real problem of any **organization or client** who pays for it
2)	**Used by the developer** by himself or herself	**Used by clients** organization for their business
3)	**No** importance given to the **functionalities** of software. Main focus is on output only	Much importance given to the correct function of the system
4)	Software **not designed with quality attributes** like portability, robustness, reliability and usability in mind	Software should be **designed with quality attributes** like reliability, user-friendliness, etc.
5)	No importance given for testing. If error occurs, users fix them when they are found. **Bugs are tolerable**	Software Testing is done to ensure the system is working properly. Because malfunction (bug or defect or failure) leads to financial or business loss, inconvenience to users, loss of property or life. **Bugs are not tolerable**

S.NO.	STUDENT SOFTWARE SYSTEM	INDUSTRIAL STRENGTH SOFTWARE
6)	Software developed as **single process (**monolithic architecture).Hard to fix the errors in the early stage	It uses **modular approach.** Development be broken into stages. Each stage is evaluated and reviewed to remove bugs earlier
7)	**No documentation** since it is for personal use	**Documents** needed for the user ,the organization and the project
8)	Only 5% of the total effort spent in testing	Testing takes 30 to 50% of the total effort.
9)	**Less Effort. Productivity is high**	Requires **much effort** that **lowers** the **productivity**
10)	**No Investment**	**Heavy Investment**

Industrial strength software

- If 1/5th productivity, and increase in size by a factor of 2, industrial strength software will take 10 times effort

- Brooks thumb-rule: Industrial strength software costs 10 time more than student SW

- In this course, software == industrial strength software

Late & Unreliable

Software development remains a weak area:

Problem 1: Software project delivered late or over budget → runaway

Problem 2: Lack of reliability – Unreliability leads to failure

- Software does not do what it is supposed to do?
- Software does something not supposed to do.

Problem 3: Hardware wears out due to age

- Software never wears out due to age but deteriorate (weaken)
- Failures are not due to aging related problems
- Failures occur due to bugs or errors that get introduced during development
- The bug that causes a failure typically exists from start, only manifests later

Productivity

- An engineering project driven by cost and schedule
- SE driven by 3 major factors: Cost, Schedule and quality
- Cost: In software cost is mainly manpower cost, hence measured in person-months
- Schedule is in months/weeks
  - Cycle time from concept to delivery to be small
  - Needs to be developed faster and within specified time
- Productivity capture both Cost and Schedule

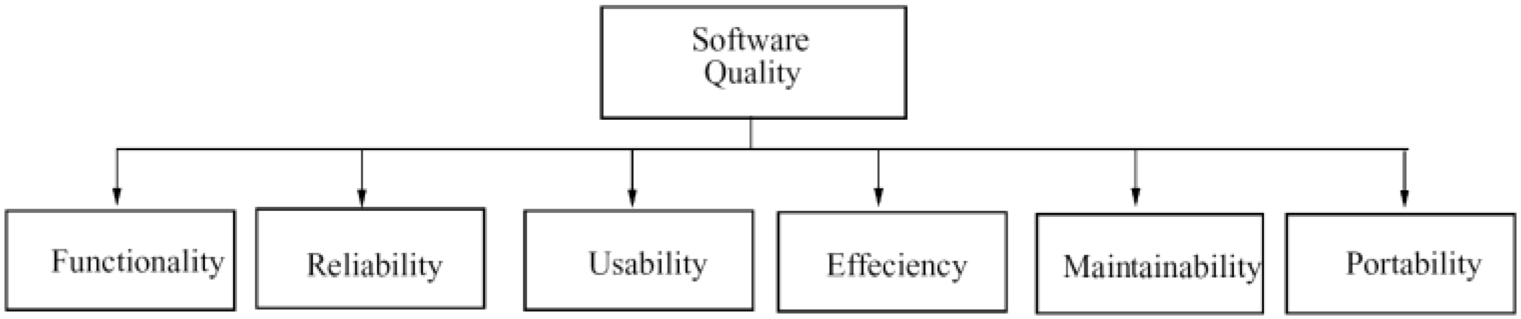
  If P is higher, cost is lower

  If P is higher, time taken can be lesser
- Approaches used by SE must deliver high Productivity

Quality

- Quality is the other major driving factor
- Developing high Quality Software is a basic goal
- Quality of Software is harder to define
- Approaches used should produce a high Quality software

Quality – ISO standard



ISO standard has six attributes

1. Functionality
2. Reliability
3. Usability
4. Efficiency
5. Maintainability
6. Portability

- Functionality (suitability, accuracy, security,…)
  The capability to provide functions which meet stated and implied needs when the software is used

- Reliability (trust-worthy)
  The capability to maintain a specified level of performance

- Usability (understandability, learn ability, operatability)
  The capability to be understood, learned and used

- Efficiency (capability, competence,..)
  The capability to provide appropriate performance relative to the amount of resources used

- Maintainability (changeability, testability, stability,…)
  The capability to be modified for purposes of making corrections, improvements or adaptations

- Portability (independence, strong,…)
  The capability to be adapted for different specified environments

Quality

- Reliability = Probability of failure
  - hard to measure
  - approximated by no. of defects in software

- To normalize Quality = Defect density
  - Quality = No. of defects delivered / Size

- Objective- to reduce no. of defects per KLOC

- Current practices: less than 1 def/KLOC

- What is a defect? Project specific!

Maintenance & Rework

- Once SW delivered or deployed, it enters Maintenance phase

- What is Maintenance?
  - Changes done after development
  - Modification done to the existing software
  - Not a part of development

- Maintenance Activities:
  - Correct residual errors requires corrective maintenance
  - Environment changes – adaptive maintenance
  - Upgrades & enhancing features requires enhanced maintenance

Maintenance & Rework

- Corrective Maintenance
  - Software system developed with residual errors (difference between observed & predicted value)
  - Remove errors that leads to change

- Enhanced Maintenance
  - Even without errors, software needs to be changed
  - Must be upgraded or enhanced by adding more features and provide more services that leads to change

- Adaptive Maintenance
  - Environment in which it operates changes.
  - Change of environment → change in software
  - Change in software → changes the environment → requires change

What’s happen in Maintenance Phase?

- Based on the existing software, it creates new software system
    - Understand the existing s/w before modification
    - Modifier should understood the effects of change before modification
    - Modification leads to undesirable side effects -> regression testing should be done to test the old test cases so that no new errors have been introduced
    - Making the changes → code and document
    - Testing the new parts & retest the old parts that were changed

Rework

- Biggest problem in large and complex system development is: rework
- To avoid rework: State/ specify the requirements, functionalities, interfaces and constraints before development

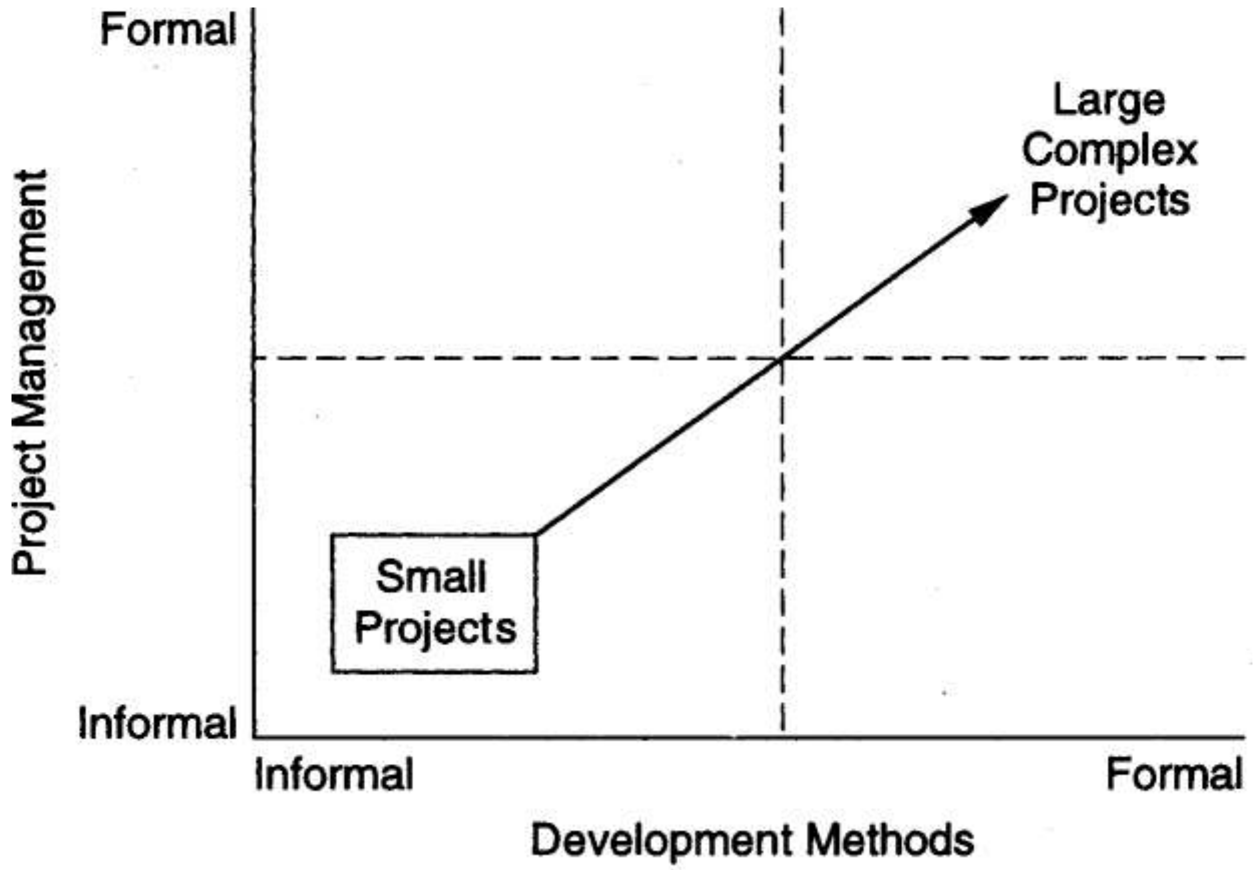
- What leads to rework?
  - Requirement is not understood completely and clearly before development leads to rework
  - Clients needs additional requirements during development – rework
  - Large and complex systems take long years to complete. During that time, clients needs change of requirements

- 30 – 40% of effort spent on rework
- 30-40% of development cost spent on rework

Scale

- SE must deal with problem of scale
  - methods for solving small problems do not scale up for large problems
  - industrial strength SW problems tend to be large
- Scales – small, medium, large, very large
- Two clear dimensions in this
  - Engineering methods
  - Project management
- An illustration of issue of scale is counting the number of people in a room vs. taking a census
  - Both are counting problems
  - Methods used in first not useful for census
  - For large scale counting problem, must use different techniques and models
  - Management will become critical

Scale



Formal
Informal
Project Management
Large Complex Projects
Small Projects
Informal
Formal
Development Methods

Scale Examples

Size	**Software**	**Languages**
Gcc	980KLOC	C, C++, yacc
Perl	320 KLOC	C, perl, sh
Appache	100 KLOC	C, sh
Linux	30,000 KLOC	C, c++
Windows XP	40,000 KLOC	C, C++

Rate of Change

- Only constant in business is change
- Software must change to support the changing business needs
- Software Engineering practices must accommodate change
  - Methods that disallow change, even if high Q and P, are of little value