Software Engineering

Week 1

Muhammad Daniyal Liaquat

Introduction to Software Engineering

- Software Engineering is composed of two words “Software” and “Engineering”.
- Software: A program for a computer could be called as a software but many things other than software are also part of software. These are:
- Data: On which program operates
- Documentation: Could contain many things like help on how to use software, how program is written etc.

2

Importance of Software

- Some of the major areas in which software is playing an important role:
- *Business Decision Making:* Analysing large data and making decisions on its basis. Accurate and easy to use
- *Modern Scientific Investigation and Engineering problem solving:* Intensive calculations and data analysis. Examples include DNA encoding, biometrics

3

Importance of Software

- *Games:* For all ages
- *Embedded Systems*: Many kind of gadgets being employed in our daily life like micro-controllers in cars, TV's etc.
- Similarly internet applications, office automation, education use software. Due to central and massive use it is contributing a lot in activity. Billions and trillions are being invested in this field.

4

Engineering

- Engineering is defined as *"The process of productive use of scientific knowledge is called engineering".*
- *Computer Science and Software Engineering:* Various branches of engineering like electrical and mechanical are based on physics. Physics itself is not engineering but use of physics in making buildings, devices or electronic circuits become engineering. The relation of computer science with software engineering is similar one. Considering this we can define software engineering as "The process of utilizing knowledge of computer science in effective production of software systems"

5

Difference between software system and other systems

- Non software systems like TV, car etc may malfunctioned during their functioning. So they wear out. However software systems does not wear out.
- Software systems do have defects called as bugs but these bugs remain their from the first day to the end.
- If part of a car wear out it could be replaced with other component. However in software same process of replacing faulty part may not work.
- Frequency of changes in other systems is very low once the design is finalized. These changes are very frequent in software system in the form of enhancements, interface changes or making a new system altogether.

6

Software Crisis

- Due to advancements in hardware in early 1960's hardware required more powerful and complex software. This complex software was difficult to write. So the tools and techniques that were used for less complex software became inapplicable for more complex software.
- Techniques used to develop small software became inapplicable for large software and resulted in to following consequences:

7

Software Crisis

- The complex software being built from old tools remained incomplete.
- Most of times it was delivered late
- Most of the projects were over budgeted
- Most of the software were not reliable as they were not able to do for what they were built for

8

Software Crisis

- The above problems resulted a conference in 1960's in which term "software crisis" was introduced
- Term "software engineering" was coined in the same conference.
- Attendees agreed to use engineering principles as were being used in other engineering
- So software engineering is outcome of software crisis when people realized that it is not possible to construct complex software using old techniques. Similarly people also realized that just coding is not enough.

9

"Software Engineering" Definition

- IEEE define Software Engineering as

"The application of systematic, disciplined, quantifiable approach to the development, operation and maintenance of software; that is application of engineering to software".

- Definition by Ian Somerville "Software Engineering is not just concerned with the technical processes of software development but also with the activities such as software project management and with the development of tools, methods and theories to support software production".

10

"Software Engineering" Definition

- According to definition software engineering encompasses all those things that are used in software production like:
  - Programming Language
  - Programming Language Design
  - Software Design Techniques
  - Tools
  - Testing
  - Maintenance
  - Development etc.

11

Well Engineered Software

- Well engineered software has the following characteristics
  - It is reliable
  - It has good user-interface (User friendly)
  - It has acceptable performance (efficient)
  - It is of good quality
  - It is cost-effective
  - Provide the required functionality
  - Maintainable

Most of times software engineers ends up in conflict among all these goals. It is a big challenge for SE's to resolve conflicts.

12

The Balancing act

- To resolve the conflicts mentioned before Software Engineering is a way to go. So it is a balancing act. Software engineer have to balance cost, efficiency, reliability etc.
- There is always a trade off among all these requirements of a software. Improvement in one may cause decrement in other. For example there may be tension among the following
  - Cost Vs. Efficiency
  - Cost Vs. Reliability
  - Efficiency Vs. User Interface

  A software Engineer is required to analyse these conflicting entities and tries to strike a balance

13

The Balancing act

- Software Engineer is required to analyse weights to different components so that resulting software will have acceptable quality, acceptable performance and will have acceptable user-interface.
- Some software should be more efficient. For example a cruise missile or a nuclear reactor should have high performance and reliability. In this case cost and user friendliness could be compromised.
- It is an art to come up with such a good balance and that art could be learned with hard work and experience.

14

Law of Diminishing Act

Example:

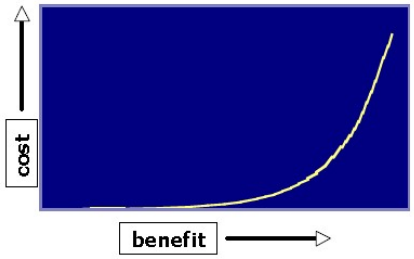
Sugar can dissolve into water up to certain quantity. After high saturation no more sugar could dissolve in the solution.

Law of diminishing act describe same phenomenon in Software Engineering

To improve software reliability, efficiency, user friendliness or other components require a cost. After investing money quality can be improved but after that amount of money being invested return less result as compared to investment. This could be explained from the following diagram

15

Law of Diminishing Act



Therefore in most cases after reaching a reasonable level of quality there are no further efforts to improve the quality of software.

16

Software Related Activities

Caper Jones after analysing 10,000 project came up with 25 categories related with software development. Eight of these activities are

1. Project Management (PM)
2. Requirement Engineering (RE)
3. Design
4. Coding
5. Testing
6. Software Quality Assurance (SQA)
7. Software Configuration Management (SCM) (Software change control)
8. Software Integration etc.

None of these activities are dominant above others in term of effort put into it. For example coding is very important but is no more than 13-14% of whole effort in Software Development

17