Introduction to Software Engineering

SE-110

Software and Software Engineering

Lecture # 1, 2, 3
21,22, 23 Jan

Rubab Jaffar
rubab.jaffar@nu.edu.pk

Today's Outline

- Administrative Stuff
- Overview of 110
- Introduction to Software Engineering
- Why study Software Engineering?

About me

- Graduated from FJWU
- Completed Masters from NUST
- Research interests:
  - Software engineering
  - Database systems
  - Algorithm analysis

Office hours and meeting rules

- Office Room 6 (CS building)
- Office hours: 3:00 to 4:00 pm
  - (Monday, Wednesday, Thursday)

About course

- Study and application of the principles and techniques of computer science, engineering, and mathematical analysis to the design, development, testing, and evaluation of the software and the systems that enable computers to perform their many applications.
- This is perhaps the most important course in your CS curriculum in CAREER PATH FORECAST.

Career Forecast

- Software engineers are in demand in not only at software development companies but also in all other organizations that are involved in the development of significant information systems – including
- governments,
- telecommunications companies,
- the chemical industry,
- the bio-medical industry,
- financial institutions,
- pharmaceuticals,
- healthcare sector corporations,
- engineering and manufacturing firms,
- e-commerce,
- In fact in the words of noted software engineer, David Parnas, "career opportunities for software engineers are essentially unlimited."

Knowledge assumed

- Programming language concepts

Skills assumed

- We assume you have the skills to code in programming language therefore you
  - can design, implement, test, debug, read, and understand the programs.

Tentative Mark Distribution and Grading Policy

- Assignments: 10%
- Quizzes: 10 %
- Mid Exam 1: 15 %
- Mid Exam 2: 15 %
- Class Participation: 3 – 5 %
- Final: 40%
- Presentations / Projects: 10%
- ***Absolute Grading Policy***

Project

- An advance project
  - Provides interesting problem, realistic pressures, unclear/changing
- Small 3-4 member teams
- Emphasis on good SE practice/methodology
  - Homework's and deliverables tied to the project
  - Grading: practices more important than the end product

Course Ethics

Projects/Assignments

- Deadlines are always final
- No credit for late submissions
- One student per assignment at maximum
- Project Proposal Submission : 3rd Week
- Project Presentation : Last week

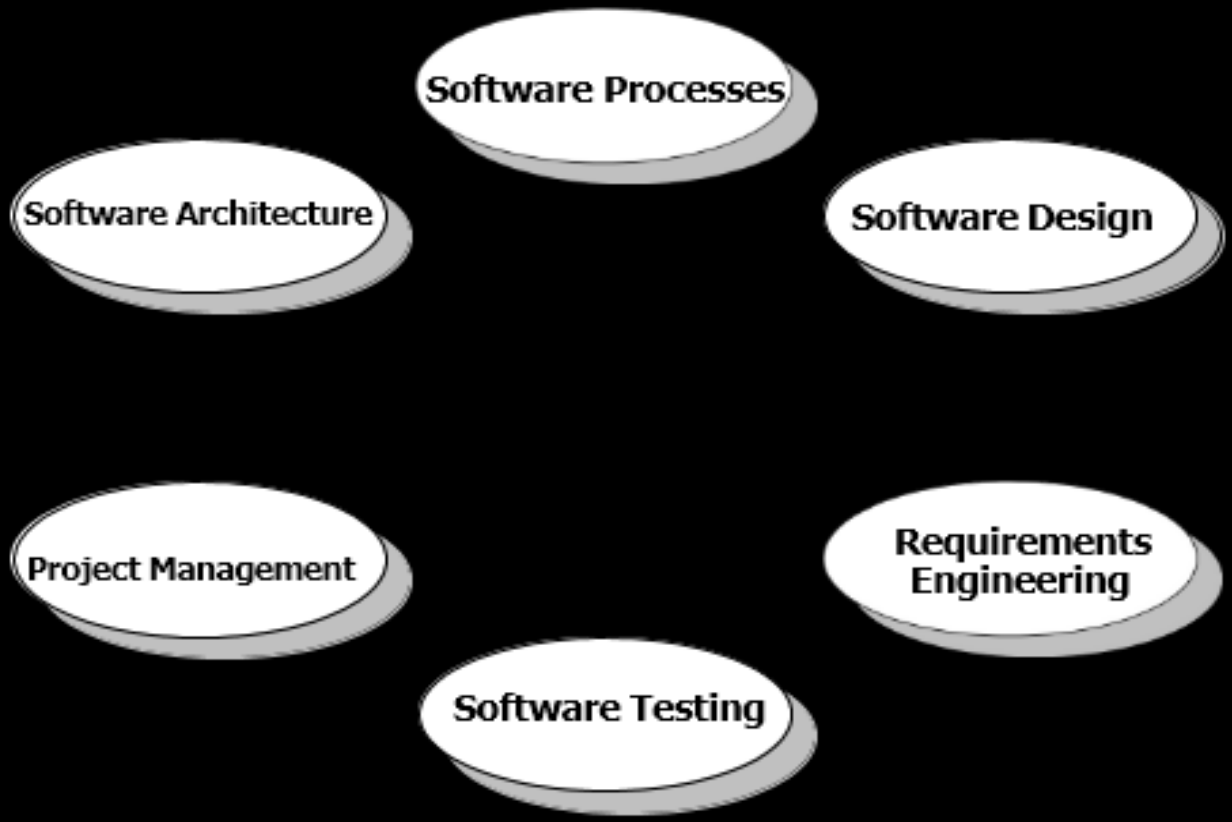
Quizzes

- Announced
- Unannounced

Honesty

- All parties involved in any kind of cheating in any exam will get zero in that exam.

Major Topics of the Course



Course Outline

- Overview of Software Engineering SE
- Process framework and models
- Requirement Engineering
- Requirement Elicitation
- Requirement Modeling
- Software Architecture
- Risk Management
- Software Quality
- Testing
- Project Management
- Work Breakdown structure

Course Material

- You will have Presentations of each topic and ***reference books*** in PDF format will be available at slate.
- Text Books
  - Ian Sommerville, Software Engineering 10th Edition
  - Pressman, R S Software Engineering: A Practitioners Approach (7th Edition, European Adaptation), McGraw Hill, 1994
- Reference Books
  - (Undergraduate Topics in Computer Science) Pankaj Jalote - A Concise Introduction to Software Engineering-Springer (2008)

Course Goals

- Have a sound understanding of the fundamental concepts of the software engineering paradigm
- Understand and apply different practices used in software industry for software development
- Recognize the risks of software failure
- Verify through review practice accuracy, ambiguity and completeness of a software
- Develop a software engineering mindset

Why Study Software Engineering ?

What is Software?

Software is:

(1) instructions (computer programs) that when executed provide desired features, function, and performance;

(2) data structures that enable the programs to adequately manipulate information and

(3) documentation that describes the operation and use of the programs.

What is Software?

- *Software is developed or engineered, it is not manufactured in the classical sense.*
- *Software doesn't "wear out."*
- *Although the industry is moving toward component-based construction, most software continues to be custom-built.*

Wear vs. Deterioration

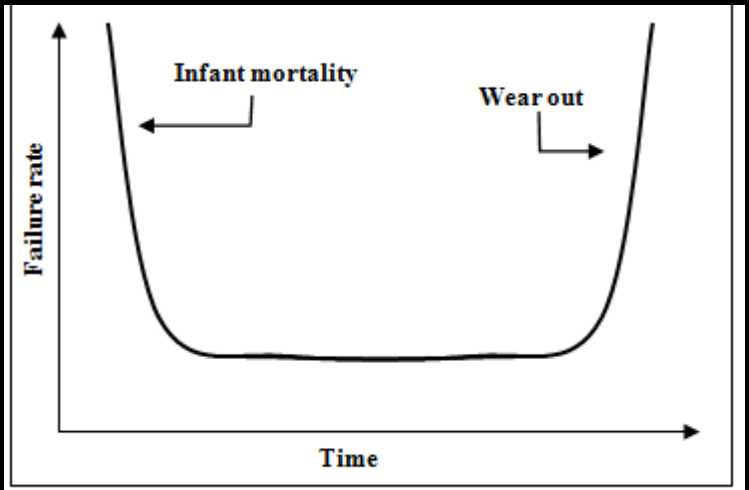


Figure 01: Failure curve for hardware

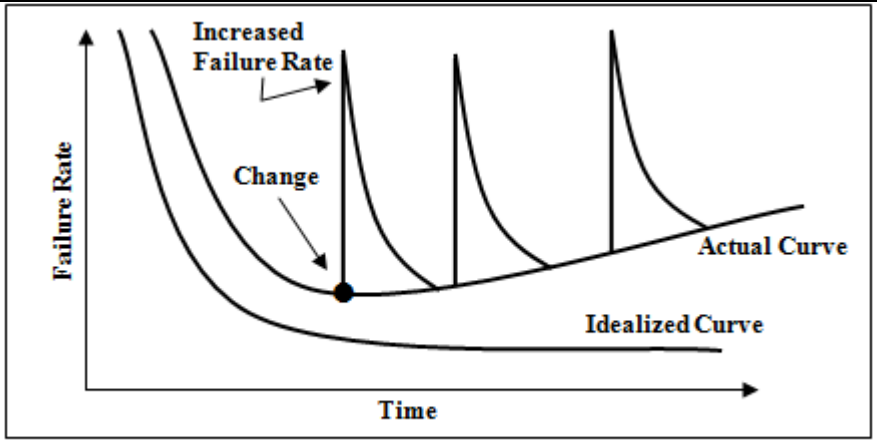


Figure 02: Failure Curves for software

Software Applications

- System software
- Application software
- Engineering/scientific software
- Embedded software
- Product-line software
- WebApps (Web applications)
- AI software

Software Application Domains

1. System software:

   collection of programs written to service other programs. such as compilers, editors, file management utilities

2. Application software:

   stand-alone programs for specific needs.

3. Engineering/scientific software:

   Characterized by “number crunching”algorithms. such as automotive stress analysis, molecular biology, orbital dynamics etc

Software Applications

4. Embedded software:

resides within a product or system. (key pad control of a microwave oven, digital function of dashboard display in a car)

5. Product-line software:

focus on a limited marketplace to address mass consumer market. (word processing, graphics, database management)

6. WebApps (Web applications)

network centric software. As web 2.0 emerges, more sophisticated computing environments is supported integrated with remote database and business applications.

7. AI software

uses non-numerical algorithm to solve complex problem. E.g Robotics, expert system, pattern recognition game playing

Software—New Categories

- Open world computing—pervasive, distributed computing
- Ubiquitous computing—wireless networks
- Netsourcing—the Web as a computing engine
- Open source—"free" source code open to the computing community (a blessing, but also a potential curse!)
- Also …
  - Data mining
  - Grid computing
  - Cognitive machines
  - Software for nanotechnologies

Why study Software Engineering ?

- We will try to understand . .
  - Typical ***Engineering approach***
- How software development ***is different*** . .
  - from ***other engineering*** disciplines ?

Engineering Approach

Before discussing this we need to understand difference between **science** and **Engineering**

Difference b/w Science & Engg

- **What is Science ?**
  - Scientist invented a ***wheel***,
  - ***Engineer*** will use that wheel for bi-cycle, motor-car, for anything.

  - Scientist invented an ***electric motor***,
  - ***Engineer*** will use that motor to construct things like pumps, electric-lifts, toys, etc.

- **What is Engineering ?**
  - Process of ***productive use*** of scientific knowledge is called engineering.

Difference b/w CS & SW-Engg

- ***Computer science*** is knowledge contains theories and fundamentals.
- Somebody has invented followings
  - Database
  - Design methodology
  - Algorithms
  - Testing methods
  - Methods of analysis and Verification.

- ***Software Engineering*** is a approach of applying engineering principles in order to develop software.
- SW-Engg applies ***various phases*** to implement all of the components needed to satisfy the user requirements.
- SW-Engg follows engineering which is to ***plan,*** with ***suitable skills*** and craft for moving into Mass-production.

Engineering Approach

- Day to day life we see successful projects in other fields of engineering
  - Such as in civil, or mechanical engineering.
  - Building bridges & roads are also large projects and the carryout large efforts in construction.

- It is our common observation that usually large software projects are NOT very successful.
  - In the middle of software projects we realize that **more time** is required or
  - **more resources** are required to complete the project, than we initially anticipated.

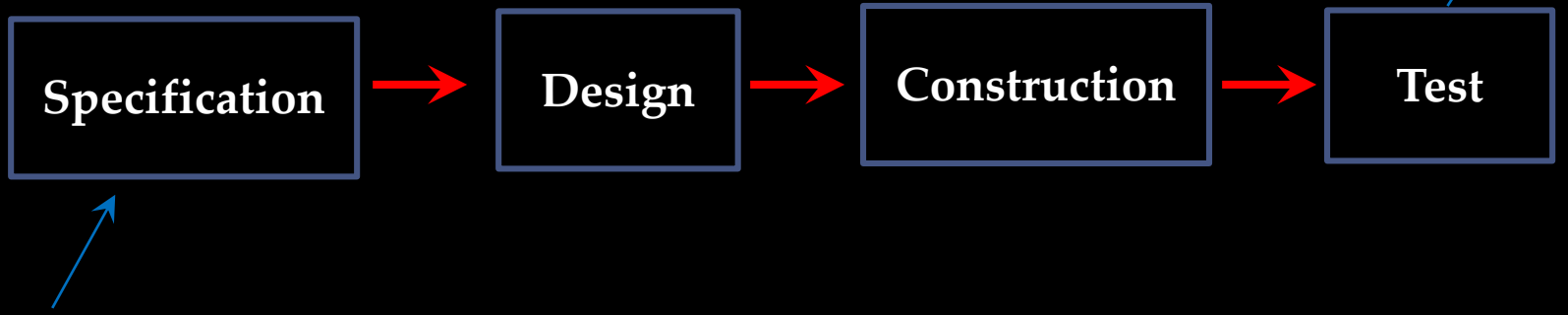
Engineering Approach

- Following are ***Large*** engineering projects ***in other discipline*** and are normally successful
  - Building bridges & roads
  - Power Plants
  - 60 story luxury apartments
  - Aircraft missiles
- Why such kinds of project are generally successful?
- What are the ***practices being followed*** in these kinds of project that make them successful?
- Is there any ***common attributes*** or these are different than developing a software?

Engineering Approach

Using this approach →

- Construct a house
- Construct a bridge
- Or design a CAR



This ***engineering approach*** helps engineers developing what they intend to and complete projects successfully

Example – Construct a house

Following are logical steps to develop a house

- Stable and *detail requirements* from customer.
- ***Record*** requirement so that we ***do not miss*** any requirement.
- After recording it must be *validated* by user again.
- Any ***Change*** means additional cost and time.

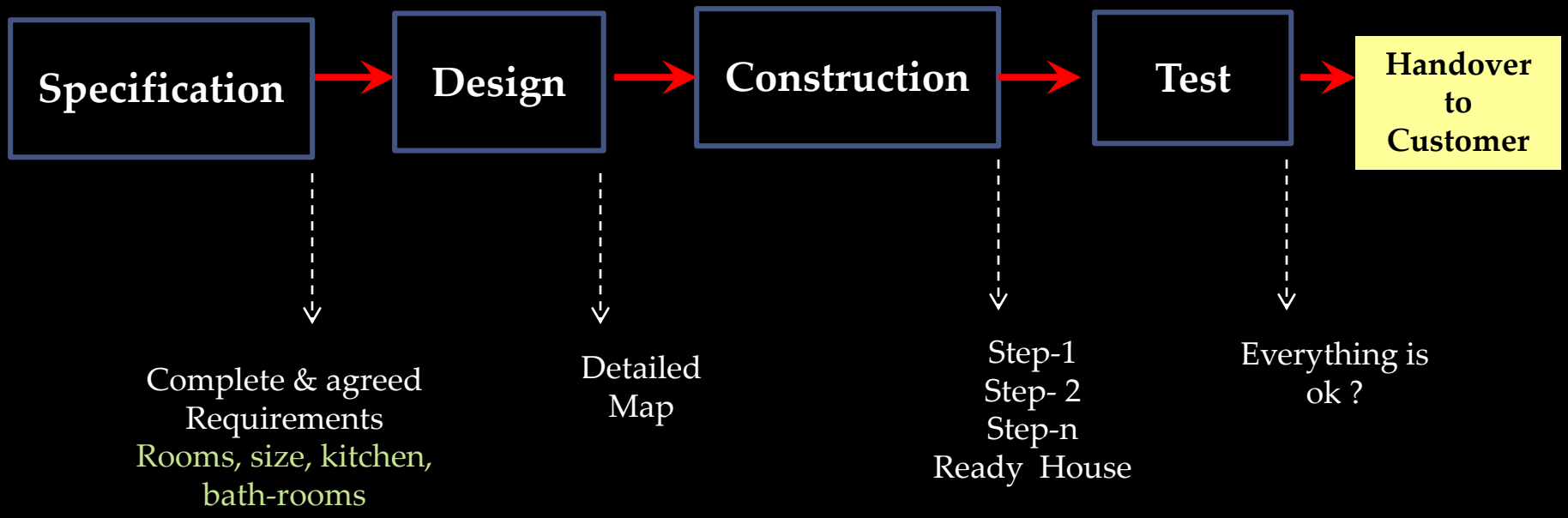
- Once the *requirements are agreed* (baseline) next step we develop ***a map*** (design)
  - Size of rooms
  - Doors and windows placement
  - Size of kitchen, bathrooms etc
  - Design should have electrical cabling layout
  - plumbing layouts etc

Example – Construct a house

- Once the map is *approved* by the customer you ***start*** construction of house.
- ***What if*** you developed the structure of the house and customer request you to ***change*** dimensions of rooms?
  - Can you accommodate that change at this stage?
- How do you manage agreed (scope) map?

Example – Construct a house

How Engineering approach helped **construction** of **House** ?



Software Engg is different

Software Engineering is different

- Requirements are complex and ***multi-dimensional***.
- Difficult to asses *progress* of the project.
- Delivering an ***intangible*** product.
- Software ***get obsolete*** quickly.
- Can be developed using ***different approach***
  - Approach = process, templates, techniques etc
  - Plan-driven (Predictive) & change-driven (adaptive)

Requirements are complex

Components of any automation Solution

- Features and functions
- Process/Practices (need to be automated)
- People (their experience, knowledge, and ability to adopt change)
- Usability (Desktop, Internet, Smart-phone)
- Environment (Temp, Dust-free etc)

Guess my requirements

I need a equipment

- That is equipped with internal combustion engine (such as Car, truck)
- That is available in yellow & black colour
- It has three wheels
- Has 3 speed forward and 3 speed reverse
- Levels uneven surface nicely and quickly

Difficult to asses progress

- You can asses progress in civil engineering
  - **50%** work is completed (in 60 story building)
  - You may verify that by visiting the project site and see 30 stories are built or not
- When Project Manager of software project says 90% work complete.
  - How can you verify that statement **?**
  - Is it really 90% or much less . . .

Delivering an intangible thing

- In civil engineering you are delivering house or a road or a bridge.
  - Can understand the time taken and cost
  - Buy a Car you look for rooms, their size, design etc

- In software project what do you deliver ?
- A CD only cannot asses ***how much efforts*** it had taken and ***what is its cost***.

Can differentiate - 2 kilo potato or 20 Kilos
Cannot differentiate - Small SW or large SW

Software get obsolete quickly

- Software is developed to ***automate business process***, as business grows its process changes.
- Need some changes approximately after very 8-9 months in developed software.
- This is ***not that case*** when you purchased a ***brand-new car*** or new house.

Can be developed using different approach

- Software house use different approaches to develop same functionality
  - Plan-Driven (Sequential or Predictive)
  - Change-Driven (Agile or Adaptive)

- Building a 60 story cannot be build with ***steel***, ***cement*** and ***concrete***. Approach is not very different in other engineering discipline.

Software Engineering is different

- It is difficult to estimates efforts and plan these activities for software development.
- There are so many failures in software development projects.
- It is NOT difficult to estimate efforts in construction of 50-story building.

Software Engineering is different

- We must have, separate specialized team for . . .
  - Requirements elicitation
  - Dedicated team for software design
  - Team of developers
  - separate team for testing etc.
- All these team needs supervisions and monitoring for the work they do.
- Typical engineering approach is necessary for the successful software project.

Software Engineering –

Definition

Software Engineering as defined by **IEEE**

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

Why study Software Engineering ?

Why study Software Engineering ?

- What are ***challenges*** in software development ?
  - that will be addressed by using Engineering approach

- What is the need/required ?
  - Unable to handle large software projects - Why
- Why are we studying software Engineering?

Why study Software Engineering ?

- Historically 2-3 people used to develop ***small*** and ***simple*** software that is ok. No problems in small & medium size projects
  - Example: your FYP is small and simple project

- When a ***large*** software with ***complex*** requirements are developed through a large team (70-80 people) we had serious problems (called *Software crisis*).
  - Example: Developing a core banking software is large project, will require large team and few years.

Software Crisis

- In early ***60s*** software techniques that were used to develop ***small*** software were ***not applicable*** for ***large*** software applications.
  - In most of the cases that software which was tried to be build using those old tools and techniques were either ***not complete***.
  - Most of the times it was delivered ***too late***.
  - Most of the projects were ***over-budgeted***.
  - A few projects caused loss of life
  - Some projects caused property damage.

Cost of Failure

- ***Allstate Insurance*** – (www.allstate.com) in 1982
  - **$8M** were allocated to automate business
  - EDS providing software
  - Initial *5-year* project continued for *10-years*, until 1993
  - Cost approached $100M

- ***Bank of America*** – (www.bankofamerica.com)
  - Spent **$23M** on an initial **5 year** accounting & reporting system
  - Spent $600M trying to make it work
  - Project was *cancelled*
  - Lost customer accounts - $Billons

Cost of Failure

- ***Blue Cross and Blue Shield of Wisconsin*** – 1983
  - EDS was hired to build ***$200M*** computer system
  - Delivered on time in 18 months
  - System didn't work – issued $60M in overpayments and duplicate checks
  - Blue Cross lost 35,000 policy holders by 1987

NATO Conference

- In the fall of 1968 and again the in fall of 1969, NATO hosted a conference devoted to the subject of software engineering.

- The motivation for these conferences was that the computer industry at large was having a great deal of trouble in producing large and complex software systems i.e. the software crisis.

- Participants were solicited from computer manufacturers, computer users, software houses, and academia. Over the years, the conference reports have gained a certain amount of classical aura.

NATO Conference

- In late 1968 the Conference recommended the term Software Engineering. The phrase 'software engineering' was deliberately chosen as being provocative, in implying the need for software manufacture to be based on the types of theoretical foundations and practical disciplines, that are traditional in the established branches of engineering.

Software Engineering

- Writing software has evolved into a profession concerned with
  - How best to maximize the quality of software and of how to create it.
  - How best to create high quality software is a separate and controversial problem covering software design principles
  - How best to deliver software on time and as quickly as possible, work-place "culture", hiring practices, and so forth.

All this falls under the broad rubric of software engineering

That is all