## Introduction to Software Engineering

This course is a continuation of the first course on Software Engineering. In order to set the context of our discussion, let us first look at some of the definitions of software engineering.

Software Engineering is the set of processes and tools to develop software. Software Engineering is the combination of all the tools, techniques, and processes that used in software production. Therefore 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.

So all those thing that are related to software are also related to software engineering.

Some of you might have thought that how programming language design could be related to software engineering. If you look more closely at the software engineering definitions described above then you will definitely see that software engineering is related to all those things that are helpful in software development. So is the case with programming language design. Programming language design is one of the major successes in last fifty years. The design of Ada language was considered as the considerable effort in software engineering.

These days object-oriented programming is widely being used. If programming languages will not support object-orientation then it will be very difficult to implement object-oriented design using object-oriented principles. All these efforts made the basis of software engineering.

**Well-Engineered Software**

Let’s talk something about what is well-engineered software. Well-engineered software is one that has the following characteristics.

* It is reliable
* It has good user-interface
* It has acceptable performance
* It is of good quality
* It is cost-effective

Every company can build software with unlimited resources but well-engineered software is one that conforms to all characteristics listed above.

Software has very close relationship with economics. When ever we talk about engineering systems we always first analyze whether this is economically feasible or not. Therefore you have to engineer all the activities of software development while keeping its economical feasibility intact.

The major challenges for a software engineer is that he has to build software within limited time and budget in a cost-effective way and with good quality

Therefore well-engineered software has the following characteristics.

* Provides the required functionality
* Maintainable
* Reliable
* Efficient
* User-friendly
* Cost-effective

But most of the times software engineers ends up in conflict among all these goals. It is also a big challenge for a software engineer to resolve all these conflicts.

**The Balancing Act!**

Software Engineering is actually the balancing act. You have to balance many things like cost, user friendliness, Efficiency, Reliability etc. You have to analyze which one is the more important feature for your software is it reliability, efficiency, user friendliness or something else. There is always a trade-off among all these requirements of software. It may be the case that if you try to make it more user-friendly then the efficiency may suffer. And if you try to make it more cost-effective then reliability may suffer. Therefore there is always a trade-off between these characteristics of software.

These requirements may be conflicting. 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 analyze these conflicting entities and tries to strike a balance.

**Challenge is to balance these requirements.**

Software Engineers always confront with the challenge to make a good balance of all these things depending on the requirements of the particular software system at hand. He should analyze how much weight should all these things get such that it will have acceptable quality, acceptable performance and will have acceptable user-interface.

In some software the efficiency is more important and desirable. For example if we talk about a cruise missile or a nuclear reactor controller that are droved by the software systems then performance and reliability is far more important than the cost-effectiveness and user-friendliness. In these cases if your software does not react within a certain amount of time then it may result in the disaster like Chernobyl accident.

Therefore software development is a process of balancing among different characteristics of software described in the previous section. And it is an art to come up with such a good balance and that art can be learned from experience.

**Law of diminishing returns**

In order to understand this concept lets take a look at an example. Most of you have noticed that if you dissolve sugar in a glass of water then the sweetness of water will increase gradually. But at a certain level of saturation no more sugar will dissolved into water. Therefore at that point of saturation the sweetness of water will not increase even if you add more sugar into it.

The law of diminishing act describes the same phenomenon. Similar is the case with software engineering. Whenever you perform any task like improving the efficiency of the system, try to improve its quality or user friendliness then all these things involves an element of cost. If the quality of your system is not acceptable then with the investment of little money it could be improved to a higher degree. But after reaching at a certain level of quality the return on investment on the system’s quality will become reduced. Meaning that the return on investment on quality of software will be less than the effort or money we invest. Therefore, in most of the cases, after reaching at a reasonable level of quality we do not try to improve the quality of software any further. This phenomenon is shown in the figure below.



**Software Background**

Caper Jones a renounced practitioner and researcher in the filed of Software Engineering, had made immense research in software team productivity, software quality, software cost factors and other fields relate to software engineering. He made a company named.

Software Productivity Research in which they analyzed many projects and published the results in the form of books. Let’s look at the summary of these results. He divided software related activities into about twenty-five different categories listed in the table below. They have analyzed around 10000 software projects to come up with such a categorization. But here to cut down the discussion we will only describe nine of them that are listed below.

* Project Management
* Requirement Engineering
* Design
* Coding
* Testing
* Software Quality Assurance
* Software Configuration Management
* Software Integration and
* Rest of the activities

One thing to note here is that you cannot say that anyone of these activities is dominant among others in terms of effort putted into it. Here the point that we want to emphasize is that, though coding is very important but it is not more than 13-14% of the whole effort of software development.

Fred Brook is a renowned software engineer; he wrote a great book related to software engineering named “A Mythical Man Month”. He combined all his articles in this book. Here we will discuss one of his articles named “No Silver Bullet” which he included in the book.

Therefore, *Software Engineering is nothing but a disciplined and systematic approach to software development*.

Now we will look at some of the activities involved in the course of software development. The activities involved in software development can broadly be divided into two major categories first is construction and second is management.

**Software Development**

The construction activities are those that are directly related to the construction or development of the software. While the management activities are those that complement the process of construction in order to perform construction activities smoothly and effectively. A greater detail of the activities involved in the construction and management categories is presented below.

**Construction**

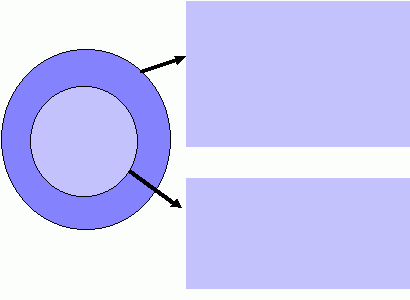
The construction activities are those that directly related to the development of software, e.g. gathering the requirements of the software, develop design, implement and test the software etc. Some of the major construction activities are listed below.

* Requirement Gathering
* Design Development
* Coding
* Testing

**Management**

Management activities are kind of umbrella activities that are used to smoothly and successfully perform the construction activities e.g. project planning, software quality assurance etc. Some of the major management activities are listed below.

* Project Planning and Management
* Configuration Management
* Software Quality Assurance
* Installation and Training



Management

Construction

* Project Planning and Management
* Configuration Management
* Quality Assurance
* Installation and Training
* Requirements
* Design
* Coding
* Testing
* Maintenance

Figure1

Development Activities

As we have said earlier that management activities are kind of umbrella activities that surround the construction activities so that the construction process may proceed smoothly. This fact is empathized in the Figure1. The figure shows that construction is surrounded by management activities. That is, certain processes and rules govern all construction activities. These processes and rules are related to the management of the construction activities and not the construction itself.

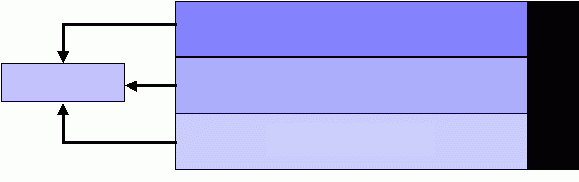
The software development organization must have special focus on quality while performing the software engineering activities. Based on this commitment to quality by the organization, a software engineering framework is proposed that is shown in Figure 2. The major components of this framework are described below.

**Quality Focus:**As we have said earlier, the given framework is based on the organizational commitment to quality. The quality focus demands that processes be defined for rational and timely development of software. And quality should be emphasized while executing these processes.

**Processes:**The processes are set of key process areas (KPAs) for effectively manage and deliver quality software in a cost effective manner. The processes define the tasks to be performed and the order in which they are to be performed. Every task has some deliverables and every deliverable should be delivered at a particular milestone.

**Methods:**Methods provide the technical “how-to’s” to carry out these tasks. There could be more than one technique to perform a task and different techniques could be used in different situations.

**Tools:**Tools provide automated or semi-automated support for software processes, methods, and quality control.



Method

Process

Quality Focus

Task Set

**TOO**

**L**

**S**

Figure 2  
 Software Engineering Framework

**Software Development Loop**

Let’s now look at software engineering activities from a different perspective. Software development activities could be performed in a cyclic and that cycle is called software development loop which is shown in Figure3. The major stages of software development loop are described below.

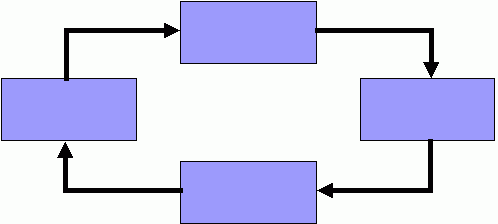
**Problem Definition:** In this stage we determine what is the problem against which we are going to develop software. Here we try to completely comprehend the issues and requirements of the software system to build.

**Technical Development:**In this stage we try to find the solution of the problem on technical grounds and base our actual implementation on it. This is the stage where a new system is actually developed that solves the problem defined in the first stage.

**Solution Integration:**If there are already developed system(s) available with which our new system has to interact then those systems should also be the part of our new system. All those existing system(s) integrate with our new system at this stage.

**Status Quo:** After going through the previous three stages successfully, when we actually deployed the new system at the user site then that situation is called status quo. But once we get new requirements then we need to change the status quo.

After getting new requirements we perform all the steps in the software development loop again. The software developed through this process has the property that this could be evolved and integrated easily with the existing systems.



Status Quo

Problem Definition

Technical Development

Solution Integration

Figure3  
Software Development Loop

**Overview of the course contents**

In the first course we studied the technical processes of software development to build industrial strength software. That includes requirement gathering and analysis, software design, coding, testing, and debugging. In this course our focus will be on the second part of Software Engineering, that is, the activities related to managing the technical development. This course will therefore include the following topics:

1. Introduction to Computer-based System Engineering
2. Software development process
3. Software process models
4. Software Specification
5. Project Management Concepts
6. Software Project Planning
7. Analysis and Management
8. Requirement Engineering Processes
9. Software Design
10. Architectural Design
11. Object-Oriented Design
12. Function-Oriented Design
13. User Interface Design
14. Quality Assurance
15. Processes & Configuration Management
16. Software Reuse Change
17. Software Re-engineering
18. Patterns
19. Assignments and projects on various stages and deliverables of SDLC.

**Course Grading Plan (CIIT)**

|  |  |
| --- | --- |
| Quizzes + Class Participation | 15% |
| Assignments (Projects) | 10% |
| 1st sessional | 10 % |
| 2nd sessional | 15 % |
| Final Exam | 50 % |
| Total | 100 % |

**Recommended Books**

1. Software Engineering by Ian Somerville
2. Software Engineering by R.S. Pressman
3. Software Engineering James F. Peters, Witold Pedrycz