Submitted By,
Adesh Oak

LECTURE SUMMARY
LECTURE #1

The best example of sophisticated-ness causing problems in a real-life scenario can be the fire at the Notre-Dame de Paris. The fire-fighting system at Notre Dame, basically, was a very well planned and implemented system, but there was a very important utility problem in it. The fire-fighting system, basically just displayed a code for the sensor where the fire was detected and nothing more. And, locating the fire just from this code wasn't a part of the security guards training. Hence, the fire spread vigorously and the church was destroyed.
From this, we can learn that a very well designed, systematically implemented software system can also have major drawbacks and might fail if it doesn't cater to the needs and knowledge level of the end-user.

What is Software?

- Software is basically a set of programs(computer instructions/code/logic statements) which make the computer do our desired work and yield us the required output.
- Software isn't physical and it is skillfully developed.
- Most importantly, software DOESN'T wear out.

Failure curve for software : -
There's a difference between the ideal failure curve and actual failure curve of software even though it is true that software doesn't wear out.
That is because, over time, the requirements of stakeholders, system performance and capacity etc.. keeps changing.

Types of Software Applications : -
1. System Software : e.g. Compilers, Operating System
2. Application Software : e.g. MS Word, Chrome
3. Engineering/Scientific Software : e.g. AutoCAD
4. Embedded Software : e.g. Softwares on sensors
5. Product-line Software : e.g. Inventory control products
6. Web/Mobile Applications : e.g. Instagram
7. AI Software

Legacy Software must change because it should adapt to changing requirements, to be interoperable across systems, it must be enhanced for more profitable business requirements.

LECTURE #2

Software Engineering

Software engineering, in simple words is planned, skillful and sophisticated development of software programs using a set of rules and technologies

More formally, (According to IEEE)
The application of a systematic, disciplined,
quantifiable approach to the development,
operation, and maintenance of software; that is,
the application of engineering to software.

Software engineering is a layered technology with tools stacked upon methods stacked upon process model stacked upon the base of "quality" focus.

Software Process

- Software process is collection of activities, actions and tasks to create product.
- Activities achieve broad objectives, actions achieve the major chunk of work and task does the small objectives with tangible outcome.

A software process framework consists of broadly two types of activities:

1. Framework Activities
   - ➔ Communication
   - ➔ Planning
   - ➔ Modeling
   - ➔ Constructions
   - ➔ Deployment

2. Umbrella Activities
   - ➔ Software Project tracking and control
   - ➔ Risk management
   - ➔ Software Quality Assurance
   - ➔ Technical reviews
   - ➔ Measurement
   - ➔ Software Configuration Management
   - ➔ Reusability Management

➔ Work Product preparation and production

Hooker has given some principles for software development:

1. The Reason it all exists
   Always be conscious of whether a feature/increment adds value to system or not.

2. KISS(Keep It Simple, Stupid!)
   The simpler designs are easier to understand, use and more often than not, elegant.

3. Maintain the vision
   Throughout the process, we should ensure the vision/motivation behind the development of the system should not be forgotten

4. What you produce, others will consume
   We should be mindful of what we are developing and develop appropriate stuff

5. Be Open to the Future
   We must develop adaptable systems

6. Plan ahead for reuse
   If not the whole system, atleast parts of code from a software should be reusable

7. Think!
   We shouldn't stop thinking and make sure the thought contributes towards effective software development.