LECTURE-01:(introduction SE)

-set of programme
-configuration life

Benefit of Configurations life :

1.set of Program
2.system documentation (structure file)
3.user documentation (how to use)

Type of Software :

1.generic (free & available)
2.Customised (paid & not for all)

Software Engineering is the application of a systematic, discipline, cost-effective, techniques which is an engineering approach for the development operation and maintenance of software.

CASE: COMPUTER AIDD,SOFTWARE, ENGINEERING.This a tools.

Computer Science :
Computer science deals with the science behind the interaction Between hardware & software.

Good software :

1 Functionality.
2.usability.
3.Effectively
4.Maintainable.
5.security.
6 Reliability.

SOFTWARE Process :

Fundamental process

1.software specification (SRs)
2.Software Development (Design & programmed)
3.software validation
4.software Evaluation

Challenges of software Engineering
1.Legacy challenge
2.Heterogeneity challenge
3.Delivery challenge

LECTURE-02(SDCL)

SDCL Phases:

1.requirement collection and Analysis
2.Feasibility study
3.Design
4.coding/Implementation
5.Integration and Testing
6.Installation /development
7.Maintenance.

What is SDLC?

Stands for fhe software development life cycle also referred to as the application life cycle.

Systematic process for building software.Every phase of the SDLC has its own process and deliverables that feed into the next process.
Increased entrance development speed. So improved client relations.

Helps us to decrease project risk and project management plan overhead.

LECTURE-03:(SDLC PHASES)

1.Economic
2.Legal
3. Operation Feasibility
4.Technical
5.Schedule

SRS - Software Requirement Specific specification.

HLD:
1.Brief description
2. An outline.
3.Interface Relationships.
4.E-R model

LLD:
1.Functional Logic
2.Db table-type
3.complete details.
4.complete input & output.

LECTURE-04 (Waterfall Model)

1.requirement are not changing frequently
2.application js not complicated and big
3. Project is short
4. Requirement is clear
5. Environment is stable

(SRs)
Requirement Analysis & specification
  SDD
  Design
    implementation unit Testing
      implementation & system testing
        Operation & maintenance

LECTURE-05(Iterative Waterfall Model)

In practice, it is not possible to strictly follow the classical waterfall model for software development work. In this context, we can view the iterative waterfall model as making necessary changes to the classical waterfall model so that it becomes applicable to practical software development projects.
The feedback paths allow for correction of the errors committed during a phase, as and when these are detected in a later phase.

When to use Iterative Waterfall Model
The requirement of the defined and clearly understood.
New technology is being learned by the development team.
There are some high risk features and goals which might in the future.
Application of Iterative Waterfall Model
Major requirements are defined but the minor details might involve when time goes.
New technologies are being used and there is a learning curve for the programmers to learn.
Resource are limited to do a huge project as if a small project automates are in contact rather than permanent.
Very high risk as a goal of the project might change from time to time.

Advantages of Iterative Waterfall Model
Feedback Path: iterative waterfall allows the mechanism of error connection because there is a feedback path from one phase to its preceding phase which it lacks in the Waterfall Model.
Simple: iterative waterfall model is simple to understand and use. It is the most widely used software development model evolved so far.

Disadvantage of Iterative Waterfall Model
More resources: may be required to implement the iterative waterfall model.

LECTURE -06(Incremental Model/ Evolutation)

Outline Description
- Concurrent Activities
  - specification
    - Initial version
- Development
  - intermediate Version
- Validation
  - Final Version

LECTURE-08(Spiral Model/Meta-Model)

1.Determining objectives alternative & constraints
2. Risk assessment & Reduction
3.Develop d validation
4Review & planning

LECTURE-10(Requirement Engineering)

Requirements engineering) refers to the process of defining, documenting, and maintaining requirements in the engineering design process. Requirement engineering provides the appropriate mechanism to understand what the customer desires, analyzing the need, and assessing feasibility, negotiating a reasonable solution, specifying the solution clearly, validating the specifications and managing the requirements as they are transformed into a working system. Thus, requirement engineering is the disciplined application of proven principles, methods, tools, and notation to describe a proposed system's intended behavior and its associated constraints.

1.what the customer desires.
2.Analyzing The need
3.Assessing feasibility
4.Negotiating a reasonable solution.
5.Specifying the solution clearly
6.validation the specification.
7.Managing the requirements.

LECTURE-11(Types of Requirement)

1. user Requirements

User requirements are statements, in a natural language plus diagrams, of what services the system is expected to provide and the constraints under which it must operate.

2. SysTem requirements

System requirements set out the system's functions, services and operational constraints in detail. The system requirements document (sometimes called a functional specification) should be precise. It should define exactly what is to be implemented. It may be part of the contract between the system buyer and the software developers.

LECTURE-12(Functional Requirement VS NonFunctional Requirement)

Software system requirements are often classified as FunctionalRequirements , NonfuntionalRequirements ,

I. Functional requirements These are statements of services the system should provide, how the system should! react to particular inputs and how the system should behave in particular sinlations. In sOMie cases, the functional requirements may also explicitly state what the system should not do.

2. Non:functional requirements Thesc;: are constraints on the services or functions offered by the system. They include tim~ng constraints, constraints on the development process and standards. Non-funcltional requireme,nts often apply to the system as a whole. They do not usually just apply to individual system features or services.

LECTURE-13(Types of Non-functional Requirements)

The types of non-functional requirements are:

ProductRequrements These requirements specify product behaviour. Examples include performance requirements on how fast the system must execute and how much memory it requires; reliability requirements that set out the acceptable failure rate; portability requirements; and usability requirements.

OrganisationalRequirements These requirements are derived from policies and procedures in the customer s and developer s organisation. Examples include process standards that must be used; implementation requirements such as the programming language or design method usedl; and delivery requirements that spedfy when the product and :Its documentation are to be delivered.

ExternalRequirements This broad heading covers all requirements that are derived from factors external to the system and its development process. These may include interoperability requirements that define how the system interacts with systems in other organisations: legislative requirements that must be followed to ensure that the system operates within the law; and ethical requirements. Ethical requirements are requirements placed on a system to ensure that it will be acceptable
to its users and the general public.

LECTURE-14(Requirement Engineering Process)

It is a four-step process, which includes -

1. Feasibility Study
2. Requirement Elicitation and Analysis
3. Software Requirement Specification
4. Software Requirement Validation

LECTURE-17(System Modeling)

1. Models of the existing system are used during requirements engineering. They help clarify what the existing system does, and they can be used to focus a stakeholder discussion on its strengths and weaknesses.

2. Models of the new system are used during requirements engineering to help explain the proposed requirements to other system stakeholders. Engineers use these models to discuss design proposals and to document the system for implementation. If you use a model-driven engineering process (Brambilla, Cabot, and Wimmer 2012), you can generate a complete or partial system implementation from system models.