System Overview.

Day & Time:

1:30am-4:30Pm

Moderator:

Chrispus Alukwe

Introduction

S**oftware engineering** is composed of two words, software and engineering.

Software is more than just a program code. A program is an executable code, which serves some computational purpose. Software is considered to be a collection of executable programming code, associated libraries and documentations. Software, when made for a specific requirement is called **software product.**

Engineering on the other hand, is all about developing products, using well-defined, scientific principles and methods.

m can be a question looking for an answer, a situation (such as an existing information system) that isn't working properly and needs improving, or a new opportunity or idea that is worthy of further consideration. In other words, when we speak of a "problem" in systems analysis and design, we don't

Basic Systems Theory Terms:

Problem

- A problem can be a question looking for an answer, a situation (such as an existing information system) that isn't working properly and needs improving, or a new opportunity or idea that is worthy of further consideration.
- In other words, when we speak of a "problem" in systems analysis and design, we don't necessarily mean that there is something wrong. We mean that there is a situation that needs to be understood and a solution to be determined.

Basic Systems Theory Terms:

System

A system is a set of *related components* that *work together* in a particular *environment* to perform whatever *functions* are required to achieve the system's *objective*.

Feedback

A system needs a feedback mechanism that can ascertain whether the outputs of the system are what they should be. If not. a system should have the ability to adjust its inputs or processes to improve the outputs. An ideal system is self-regulating. The feedback mechanism in an information system may be automated or may be manual.

Basic Systems Theory Terms:

Entropy

Entropy is a measure of the degree of disorder in a system. It is a familiar term in thermodynamics, when considering chemical systems, and is also relevant to information systems. The concept of entropy says that any system will tend towards disorder. Knowing that, we can put checks in place to monitor the correctness of the output of a system.

Internal Environment

A system operates in an environment with both internal and external components. Its internal environment is that part of its environment over which it has some control. If some aspect of the internal environment is causing some difficulty for the system, that aspect can be altered. For example, a particular information system operates in a particular office environment. If a requirement of the information system is that its users must collect data that hasn't been collected previously, this new activity can be asked of them.

there is something wrong.

Basic Systems Theory Terms:

External Environment

A system's external environment is that part of its environment over which it has no control, but it still affects the requirements of the system. For example, in a payroll system, the provincial and federal tax laws affect the procedures in the system. The tax laws must be reflected in the system, and if the laws change, the system must change to accommodate those changes. *So an analyst must be aware of the requirements of both the internal and external environments in which an information system will work.*

Subsystem

A system is usually composed of self-contained but interrelated systems that are called subsystems. It is important to be able to recognize these subsystems, because understanding this interdependence is vital to developing a *complete* system.

Basic Systems Theory Terms:

Super system

A system composed of two or more systems may be called a super system of those systems.

System Boundary

A system boundary may be thought of as the point at which data flows (perhaps as output) from one system to another (perhaps as input).

The degree to which data is free to flow from one system to another is known as the permeability of the boundary. A permeable boundary allows data to flow freely, resulting in an open system.

Basic Systems Theory Terms:

Interdependence

One of the most important concepts in Systems Theory is the notion of interdependence between systems (or subsystems). Systems rarely exist in isolation. For example, a payroll system has to access and update a personnel system.

It is important for an analyst to identify these interdependence early. It may be the case that changes you make to one system will affect another in ways you haven't considered, or vice versa.

Introduction

We can define **software engineering** as an engineering branch associated with the development of software product using well-defined scientific principles, methods and procedures.

We can alternatively view it as a systematic collection of past experience. The experience is arranged in the form of methodologies and guidelines.

A small program can be written without using software engineering principles. But if one wants to develop a large software product, then software engineering principles are absolutely necessary to achieve a good quality software cost effectively.

NEED OF SOFTWARE ENGINEERING

The need of software engineering arises because of higher rate of change in user requirements and environment on which the software is working.

Large software - It is easier to build a wall than to a house or building, likewise, as the size of software become large engineering has to step to give it a scientific process.

Scalability- If the software process were not based on scientific and engineering concepts, it would be easier to re-create new software than to scale an existing one.

NEED OF SOFTWARE ENGINEERING

Cost- As hardware industry has shown its skills and huge manufacturing has lower down the price of computer and electronic hardware. But the cost of software remains high if proper process is not adapted.

Dynamic Nature- The always growing and adapting nature of software hugely depends upon the environment in which the user works. If the nature of software is always changing, new enhancements need to be done in the existing one. This is where software engineering plays a good role.

Quality Management- Better process of software development provides better and quality software product.

CHARACTERESTICS OF GOOD SOFTWARE

A software product can be judged by what it offers and how well it can be used. This software must satisfy on the following grounds:

1. Operational
2. Transitional
3. Maintenance

CHARACTERESTICS OF GOOD SOFTWARE

Operational

This tells us how well software works in operations. It can be measured on:

- Budget
- Usability
- Efficiency
- Correctness
- Functionality
- Dependability
- Security
- Safety

CHARACTERESTICS OF GOOD SOFTWARE

Transitional

This aspect is important when the software is moved from one platform to another:

- Portability
- Interoperability
- Reusability
- Adaptability

CHARACTERESTICS OF GOOD SOFTWARE

Maintenance

This aspect briefs about how well a software has the capabilities to maintain itself in the ever-changing environment:

- Modularity
- Maintainability
- Flexibility
- Scalability.

In short, Software engineering is a branch of computer science, which uses well-defined engineering concepts required to produce efficient, durable, scalable, in-budget and on-time software products

SOFTWARE DEVELOPMENT LIFE CYCLE

LIFE CYCLE MODEL

A software life cycle model (also called process model) is a descriptive and diagrammatic representation of the software life cycle.

A life cycle model represents all the activities required to make a software product transit through its life cycle phases.

It also captures the order in which these activities are to be undertaken.

In other words, a life cycle model maps the different activities performed on a software product from its inception to retirement.

SOFTWARE DEVELOPMENT LIFE CYCLE

The Need For A Software Life Cycle Model

The development team must identify a suitable life cycle model for the particular project and then adhere to it.

Without using of a particular life cycle model the development of a software product would not be in a systematic and disciplined manner.

When a software product is being developed by a team there must be a clear understanding among team members about when and what to do. Otherwise it would lead to chaos and project failure.

SOFTWARE DEVELOPMENT LIFE CYCLE

Different software life cycle models

Many life cycle models have been proposed so far.

Each of them has some advantages as well as some disadvantages.

A few important and commonly used life cycle models are as follows:

1) Classical Waterfall Model
2) Iterative Waterfall Model
3) Prototyping Model
4) Evolutionary Model
5) Spiral Model

Questions ?

Analyzing Presentations.

1) Interfaces - Logins at least 1- 3 Users.
2.) Scope - Users Viewing of Items.
3.) Db - Forms, Students Forms navigations,
Displays of Records.
Add & Delete of Records.
Add Records.
Viewing of Reports
Relationships - If Normalized.
Relationships dependancy.

Chapters:

1 - Introduction, Background, Objectives, Research Questions,Justificationa , Significance of study.

2. - Literature Review -

3. Methodology

4. Implementation & Testing.

5. Discussions & Findings.

Analyzing Presentations.

Chapters:

1 - Introduction, Background, Objectives, Research Questions, Justification , Significance of study.

2. - Literature Review -

3. Methodology

4. Implementation & Testing.

5. Discussions & Findings.

6. Conclusions & Recommendations.