Thakur Educational Trust's (Regd.)

THAKUR RAMNARAYAN COLLEGE OF ARTS & COMMERCE

ISO 21001:2018 Certified

PROGRAMME: B.Sc (I.T)

CLASS: S.Y.B.Sc (I.T)

SUBJECT NAME: SOFTWARE ENGINEERING

SEMESTER: IV

FACULTY NAME: Ms. SMRITI DUBEY

UNIT I

Chapter 3 – Software Processes

Concepts:

Software processes

Process and Project

Component Software process

Software processes

Software Engineering is defined as the systematic approach to the development, operation, maintenance and retirement of software. The systematic approach must help to achieve a high quality and productivity (Q&P) of software. A software process (also known as software methodology) is a set of related activities that leads to the production of the software. These activities may involve the development of the software from the scratch, or, modifying an existing system.

A software process specifies the abstract set of activities that should be performed to go from user needs to final product. The actual act of executing the activities for some user needs is a software project and all the outputs that are produced while the activities are being executed are the products.

Software process activities

Real software processes are inter-leaved sequences of technical, collaborative and managerial activities with the overall goal of specifying, designing, implementing and testing a software system.

The four basic process activities of specification, development, validation and evolution are organized differently in different development processes. In the waterfall model, they are organized in sequence, whereas in incremental development they are interleaved.

1) **Software specification**

The process of establishing what services are required and the constraints on the system's operation and development.

Requirements engineering process:

Feasibility study: is it technically and financially feasible to build the system?

Requirements elicitation and analysis: what do the system stakeholders require or expect from the system?

Requirements specification: defining the requirements in detail

Requirements validation: checking the validity of the requirements

2) **Software design and implementation**

The process of converting the system specification into an executable system.

Software design: design a software structure that realizes the specification;

Implementation: translate this structure into an executable program;

The activities of design and implementation are closely related and may be interleaved.

Design activities include:

Architectural design: identify the overall structure of the system, the principal components (sometimes called sub-systems or modules), their relationships and how they are distributed.

Interface design: define the interfaces between system components.

Component design: take each system component and design how it will operate.

Database design: design the system data structures and how these are to be represented in a database.

3) Software validation

Verification and validation (V & V) is intended to show that a system conforms to its specification and meets the requirements of the system customer.

Validation: are we building the right product (what the customer wants)?

Verification: are we building the product, right?

V & V involves checking and review processes and system testing. System testing involves executing the system with test cases that are derived from the specification of the real data to be processed by the system.

Testing is the most commonly used V & V activity and includes the following stages:

Development or component testing: individual components are tested independently; components may be functions or objects or coherent groupings of these entities.

System testing: testing of the system as a whole, testing of emergent properties is particularly important.

Acceptance testing: testing with customer data to check that the system meets the customer's needs.

4) Software evolution

Software is inherently flexible and can change. As requirements change through changing business circumstances, the software that supports the business must also evolve and change. Although there has been a demarcation between development and evolution (maintenance) this is increasingly irrelevant as fewer and fewer systems are completely new**.**

Process and Project

A **process** is a sequence of steps performed for a given purpose. As mentioned earlier, while developing (industrial strength) software, the purpose is to develop software to satisfy the needs of some users or clients, as shown in Figure 2.1. A **software project** is one instance of this problem, and the development process is what is used to achieve this purpose.

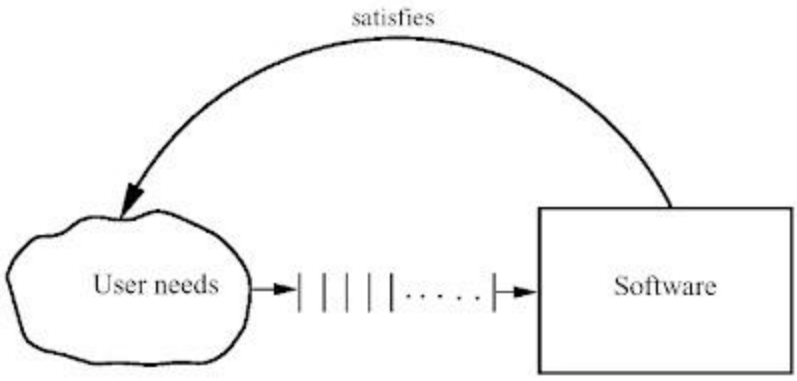

Figure 2.1: Basic problem.

So, for a **project** its development process plays a key role—it is by following the **process** the desired end goal of delivering the software is achieved. However, as discussed earlier, it is not sufficient to just reach the final goal of having the desired software, but we want that the **project** be done at low cost and in low cycle time, and deliver high-quality software. The role of **process** increases due to these additional goals, and though many processes can achieve the basic goal of developing software in Figure 2.1, to achieve high Q&P we need some "optimum" process. It is this goal that makes designing a process a challenge.

We must distinguish process specification or description from the process itself. A process is a dynamic entity which captures the actions performed. Process specification, on the other hand, is a description of process which presumably can be followed in some project to achieve the goal for which the process is designed.

In a **project**, a process specification may be used as the process the project plans to follow. The actual **process** is what is actually done in the project. Note that the actual process can be different from the planned process, and ensuring that the specified process is being followed is a nontrivial problem. We will assume that the planned and actual processes are the same and will not distinguish between the two and will use the term process to refer to both.

A process model specifies a general process, which is "optimum" for a class of projects. A process model is essentially a compilation of best practices into a "recipe" for success in the project. In other words, a **process** is a means to reach the goals of high quality, low cost, and low cycle time, and a process model provides a process structure that is well suited for a class of projects.

A process is often specified at a high level as a sequence of stages. The sequence of steps for a stage is the process for that stage, and is often referred to as a subprocess of the process.

Component software process

A process is the sequence of steps executed to achieve a goal. Since many different goals may have to be satisfied while developing software, multiple processes are needed. Many of these do not concern software engineering, though they do impact software development. These could be considered non-software process. Business processes, social processes, and training processes are all examples of processes that come under this. These processes also affect the software development activity but are beyond the purview of software engineering.

There are clearly two major components in a software process—a development process and a project management process. **The development process** specifies all the engineering activities that need to be performed, whereas **the management process** specifies how to plan and control these activities so that cost, schedule, quality, and other objectives are met. Effective development and project management processes are the key to achieving the objectives of delivering the desired software satisfying the user needs, while ensuring high productivity and quality.

During the project many products are produced which are typically composed of many items (for example, the final source code may be composed of many source files). These items keep evolving as the project proceeds, creating many versions on the way. As **development processes generally do not focus on evolution and changes, to handle them another process called software configuration control process is often used. The objective of this component process is to primarily deal with managing change, so that the integrity of the products is not violated despite changes.**

These three constituent processes focus on the projects and the products and can be considered as comprising the product engineering processes, as their main objective is to produce the desired product. If the software process can be viewed as a static entity, then these three component processes will suffice. However, a software process itself is a dynamic entity, as it must change to adapt to our increased understanding about software development and availability of newer technologies and tools. Due to this, a process to manage the software process is needed.

The relationship between these major component processes is shown in Figure 2.2. These component processes are distinct not only in the type of activities performed in them, but typically also in the people who perform the activities specified by the process. In a typical project, development activities are performed by programmers, designers, testers, etc.; the project management process activities are performed by the project management; configuration control process activities are performed by a group generally called the configuration controller; and the process management process activities are performed by the software engineering process group (SEPG).