SOFTWARE PROCESS

The concept of process is at the heart of the software engineering approach. According to Webster, the term 'process' means "a particular method of doing something, generally involving a number of steps or operations." In Software Engineering, the phrase software Process refers to the method of developing software.

A Software process is a set of activities, together with ordering constraints among them, such that if the activities are performed properly and in accordance with the ordering constraints, the desired result is produced. The desired result is, as stated earlier, high-quality software at low cost. Clearly, a process that does not scale up (*i.e.,* can not handle large software Projects) or can not produce good-quality software (*i.e.,* good-quality software is not the outcome) is not a suitable process.

2.1 SOFTWARE PROCESS

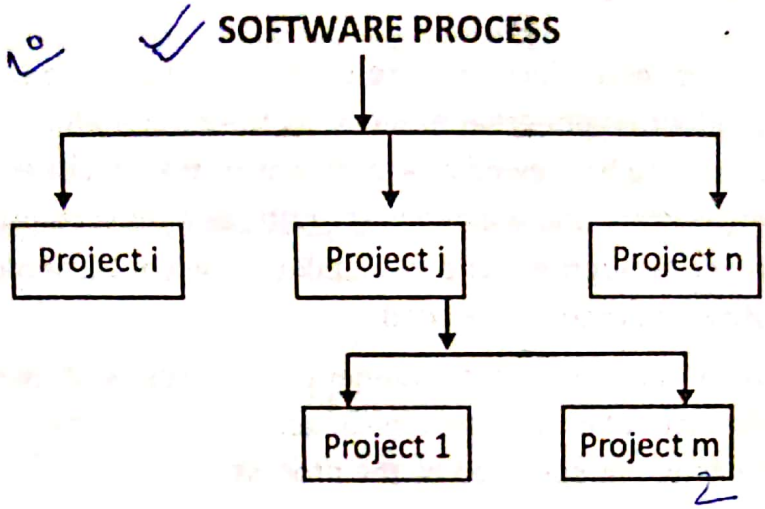
In an organization whose major business is software development, there are typically many processes simultaneously executing. Many of these do not concern software engineering, though they do impact software development. These could be considered non software-engineering process models. Business process models, social process models, and training models 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.

The process that deals with the technical and management issues of software development is called a software process. Clearly, many different types of activities need to be performed to develop software. As we have seen earlier, a software development project must have at least development activities and project management activities. All these activities together comprise the software process. As different type of activities are being performed, which are frequently done by different people, it is better to view the software process as consisting of many component processes, each consisting of a certain type of activity. Each of these component processes typically has a different objective, though these processes obviously cooperate with each other to satisfy the overall software engineering objective.

Scanned with CamSca

2.1.1 Processes, Projects and Products

A Software Process, as mentioned earlier, specifies a method of development software. A Software Project, on the other hand, is a development project in which a software process is used. And Software Products are the outcomes of a software Project. Each software development projects starts with some needs and ends with some software that satisfies those needs. Software Process Specifies the abstract set of activities that should be performed to go from user needs to final products. The actual act of executing the activities for some specific user needs is a software project. And all the outputs that are produced while the activities are being executed are the products.



Overall, the process specifies activities at an abstract level that are not project specific. It is a generic set of activities that does not provide a detailed roadmap for a particular project. The detailed roadmap for a particular project is the project plan that specifies what specific activities to perform for this particular object, when, and how to ensure that the project progresses smoothly. It should be clear that it is the process that derives a project. A process limits the degrees of freedom for a project by specifying what types of activities must be done and in what order. Further restriction on the degrees of freedom for a particular project is specified by the project plan, which, in itself is within the boundaries established by the process.

As each project is an instance of the process it follows, it is essentially the process that determines the expected outcomes of a project. Due to this, the focus of software engineering is mainly on the process.

2.1.2 Component Software Processes

The three basic types of entities that software engineering deals with-processes, Project, and products-require different processes. The major process dealing with products is the development process responsible for producing the desire product (*i.e.*, the software) and other products (*e.g.*, user manual, and requirement specification). The basic goal of this process is to develop a product that will satisfy the customer. A software project is clearly a dynamic entity in which activities are performed, and project management process is needed to properly control this dynamic activity, so that the activities do not take the project astray but all activities are general toward reaching the project goal. For large projects, project management is more important than technical methods for the success of the project. Hence we can clearly identify that there are two major components in a software process- a development process and a project management process. The development process specifies the development and

Scanned with CamScanner

- Provides management information for in-site control of the process.

2.3 WATERFALL MODEL

The waterfall model is a **sequential software development model** (a process for the creation of software) in which development is seen as flowing steadily downwards (like a **waterfall**) through the phases of **requirements analysis, design, implementation, testing** (validation), **integration,** and **maintenance.** The origin of the term 'waterfall' is often cited to be an article published in **1970** by **Winston W. Royce** (1929-1995), although Royce did not use the term 'waterfall' in this article. Ironically, Royce was presenting this model as an example of a flawed, non-working model **(Royce 1970).**

To follow the waterfall model, one proceeds from one phase to the next in a purely sequential manner. For example, one first completes requirements specifications, which are set in stone. When the requirements are fully completed, one proceeds to design. The software in question is designed and a blueprint is drawn for implementers (coders) to follow - this design should be a plan for implementing the requirements given. When the design is fully completed, an implementation of that design is made by coders. Towards the later stages of this implementation phase, disparate software components produced by different teams are integrated. After the implementation and integration phases are complete, the software product is tested and debugged; any faults introduced in earlier phases are removed here. Then the software product is installed, and later maintained to introduce new functionality and remove bugs.

Thus, the waterfall model maintains that one should move to a phase only when its preceding phase is completed and perfected. However, there are various **modified waterfall models** (including Royce's final model) that may include slight or major variations upon this process.

Scanned with CamScanner

Requirements : In **systems engineering** and **software engineering, requirements** analysis encompasses those tasks that go into determining the needs or conditions to meet for a new or altered product, taking account of the possibly conflicting requirements of the various stakeholders, such as beneficiaries or users. Systematic requirements analysis is also known as *requirements engineering*. It is sometimes referred to loosely by names such as *requirements gathering, requirements capture,* or *requirements specification*. The term *requirements analysis* can also be applied specifically to the analysis proper (as opposed to elicitation or documentation of the requirements, for instance).

Requirements analysis is critical to the success of a development project.

Requirements must be actionable, measurable, testable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.

Design

Software design is a process of problem-solving and planning for a software solution. After the purpose and specifications of software is determined, software developers will design or employ designers to develop a plan for a solution. It includes low-level component and algorithm implementation issues as well as the architectural view.

The **software requirements analysis** (SRA) step of a software development process yields specifications that are used in software engineering. If the software is "semiautomated" or user centered, software design may involve user experience design giving a story board to help determine those specifications. If the software is completely automated (meaning no user or user interface), a software design may be as simple as a flow chart or text describing a planned sequence of events. There are also semi-standard methods like Unified Modeling Language and Fundamental modeling concepts. In either case some documentation of the plan is usually the product of the design.

A software design may be platform-independent or platform-specific, depending on the availability of the technology called for by the design.

Implementation

Computer programming (often shortened to **programming or coding**) is the process of writing, testing, debugging/troubleshooting, and maintaining the source code of computer programs. This source code is written in a programming language. The code may be a modification of an existing source or something completely new. The purpose of programming is to create a program that exhibits a certain desired behaviour (customization). The process of writing source codes requires expertise in many different subjects, including knowledge of the application domain, specialized algorithms and formal logic.

Within software engineering, programming (the *implementation*) is regarded as one phase in a software development process.

There is an ongoing debate on the extent to which the writing of programs is an art, a craft or an engineering discipline. Good programming is generally considered to be the measured application of all three, with the goal of producing an efficient and maintainable software solution (the criteria for 'efficient' and 'maintainable' vary considerably). The discipline differs from many other technical professions in that programmers generally do not need to be licensed or pass any standardized (or

Scanned with CamSca

governmentally regulated) certification tests in order to call themselves 'programmers' or even 'software engineers'.

Another ongoing debate is the extent to which the programming language used in writing programs affects the form that the final program takes. This debate is analogous to that surrounding the **Sapir-Whorf hypothesis** in **linguistics** that postulates that a particular language's nature influences the habitual thought of its speakers. Different language patterns yield different patterns of thought. This idea challenges the possibility of representing the world perfectly with language, because it acknowledges that the mechanisms of any language condition the thoughts of its speaker community.

Verification

Software testing is the process used to assess the *quality of computer software*. Software testing is an empirical technical investigation conducted to provide stakeholders with information about the quality of the product or service under test, with respect to the context in which it is intended to operate. This includes, but is not limited to, the process of executing a program or application with the intent of finding **software bugs**. Quality is not an absolute; it is value to some person. With that in mind, testing can never completely establish the correctness of arbitrary computer software; testing furnishes a *criticism* or *comparison* that compares the state and behaviour of the product against a specification. An important point is that *software testing* should be distinguished from the separate discipline of **Software Quality Assurance** (SQA), which encompasses all business process areas, not just testing.

Over its existence, computer software has continued to grow in complexity and size. Every software product has a target audience. For example, a video game software has its audience completely different from banking software. Therefore, when an organization develops or otherwise invests in a software product, it presumably must assess whether the software product will be acceptable to its end users, its target audience, its purchasers, and other stakeholders. Software testing is the process of attempting to make this assessment.

A study conducted by **NIST** in 2002 reports that software bugs cost the U.S. economy $59.5 billion annually. More than a third of this cost could be avoided if better software testing was performed.

Maintinance

Software maintinance is all of the activities that make a software system available for use. The general deployment process consists of several interrelated activities with possible transitions between them. These activities can occur at the producer site or at the consumer site or both. Because every software system is unique, the precise processes or procedures within each activity can hardly be defined. Therefore, 'deployment' should be interpreted as a *general process* that has to be customized according to specific requirements or characteristics.

Software maintinance is a task that every development group has to face, when the software is delivered to the customer's site, installed and is made operational. Therefore, release of software inaugurates the operation and maintinance phase of the life cycle. The time spent and effort required to keep the software operational after release is very significant.

Scanned with CamSca