
SOFTWARE ENGINEERING

UNIT-I: Introduction - Evolving Role of Software - Changing Nature of Software – Software Myths; A Generic View of Process: Layered Technology - Process Models: Waterfall Model - Evolutionary Process Models.

The software is instruction or computer program that when executed provide desired features, function, and performance.

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



Software engineering is an engineering branch associated with development of software product using well-defined scientific principles, methods and procedures. The outcome of software engineering is an efficient and reliable software product.

Software Characteristics

Different individuals judge software on different basis. This is because they are involved with the software in different ways. For example, users want the software to perform according to their requirements. Similarly, developers involved in designing, coding, and maintenance of the software evaluate the software by looking at its internal characteristics, before delivering it to the user.

Software characteristics are classified into six major components.

Functionality: Refers to the degree of performance of the software against its intended purpose.

- **Reliability:** Refers to the ability of the software to provide desired functionality under the given conditions.
- **Usability:** Refers to the extent to which the software can be used with ease.
- **Efficiency:** Refers to the ability of the software to use system resources in the most effective and efficient manner.

Maintainability: Refers to the ease with which the modifications can be made in a software system to extend its functionality, improve its performance, or correct errors.

• **Portability:** Refers to the ease with which software developers can transfer software from one platform to another, without (or with minimum) changes. In simple terms, it refers to the ability of software to function properly on different hardware and software platforms without making any changes in it.

In addition to the above mentioned characteristics, robustness and integrity are also important. **Robustness** refers to the degree to which the software can keep on functioning in spite of being provided with invalid data while **integrity** refers to the degree to which unauthorized access to the software or data can be prevented.

Classification of Software

Software can be applied in countless fields such as business, education, social sector, and other fields. It is designed to suit some specific goals such as data processing, information sharing, communication, and so on. It is classified according to the range of potential of applications.

These classifications are listed below.

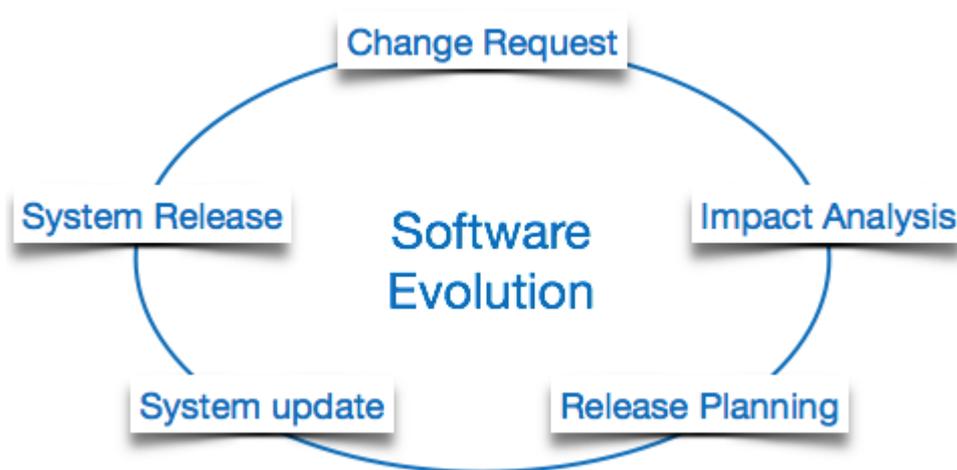
- **System software:** This class of software manages and controls the internal operations of a computer system. It is a group of programs, which is responsible for using computer resources efficiently and effectively. For example, an operating system is a system software, which controls the hardware, manages memory and multitasking functions, and acts as an interface between application programs and the computer.
- **Real-time software:** This class of software observes, analyzes, and controls real world events as they occur. Generally, a real-time system guarantees a response to an external event within a specified period of time. An example of real-time software is the software used for weather forecasting that collects and processes parameters like temperature and humidity from the external environment to forecast the weather. Most of the defence organizations all over the world use real-time software to control their military hardware.
- **Business software:** This class of software is widely used in areas where management and control of financial activities is of utmost importance. The fundamental component of a business system comprises payroll, inventory, and accounting software that permit the user to access relevant data from the [database](#). These activities are usually performed with the help of specialized business software that facilitates efficient framework in business operations and in management decisions.
- **Engineering and scientific software:** This class of software has emerged as a powerful tool in the research and development of next generation technology. Applications such as the

study of celestial bodies, under-surface activities, and programming of an orbital path for space shuttles are heavily dependent on engineering and scientific software. This software is designed to perform precise calculations on complex numerical data that are obtained during real time environment.

- **Artificial intelligence (AI) software:** This class of software is used where the problem-solving technique is non-algorithmic in nature. The solutions of such problems are generally non-agreeable to computation or straightforward analysis. Instead, these problems require specific problem-solving strategies that include expert system, pattern recognition, and game-playing techniques. In addition, they involve different kinds of search techniques which include the use of heuristics. The role of artificial intelligence software is to add certain degrees of intelligence to the mechanical hardware in order to get the desired work done in an agile manner.
- **Web-based software:** This class of software acts as an interface between the user and the Internet. Data on the Internet is in the form of text, audio, or video format, linked with hyperlinks. Web browser is a software that retrieves web pages from the Internet. The software incorporates executable instructions written in special scripting languages such as CGI or ASP. Apart from providing navigation on the Web, this software also supports additional features that are useful while surfing the Internet.
- **Personal computer (PC) software:** This class of software is used for both official and personal use. The personal computer software market has grown over in the last two decades from normal text editor to word processor and from simple paintbrush to advanced image-editing software. This software is used predominantly in almost every field, whether it is database management system, financial accounting package, or multimedia-based software. It has emerged as a versatile tool for routine applications.

Software Evolution

The process of developing a software product using software engineering principles and methods is referred to as **software evolution**. This includes the initial development of software and its maintenance and updates, till desired software product is developed, which satisfies the expected requirements.



Evolution starts from the requirement gathering process. After which developers create a prototype of the intended software and show it to the users to get their feedback at the early stage of software product development. The users suggest changes, on which several consecutive updates and maintenance keep on changing too. This process changes to the original software, till the desired software is accomplished.

Even after the user has desired software in hand, the advancing technology and the changing requirements force the software product to change accordingly. Re-creating software from scratch and to go one-on-one with requirement is not feasible. The only feasible and economical solution is to update the existing software so that it matches the latest requirements.

Software Evolution Laws

Lehman has given laws for software evolution. He divided the software into three different categories:

- **S-type (static-type)** - This is a software, which works strictly according to defined specifications and solutions. The solution and the method to achieve it, both are immediately understood before coding. The s-type software is least subjected to changes hence this is the simplest of all. For example, calculator program for mathematical computation.
- **P-type (practical-type)** - This is a software with a collection of procedures. This is defined by exactly what procedures can do. In this software, the specifications can be described but the solution is not obvious instantly. For example, gaming software.
- **E-type (embedded-type)** - This software works closely as the requirement of real-world environment. This software has a high degree of evolution as there are various changes in laws, taxes etc. in the real world situations. For example, Online trading software.

Changing Nature of Software:

Nowadays, seven broad categories of computer software present continuing challenges for software engineers .which is given below:

1. System Software:

System software is a collection of programs which are written to service other programs. Some system software processes complex but determinate, information

structures. Other system application process largely indeterminate data. Sometimes when, the system software area is characterized by the heavy interaction with computer hardware that requires scheduling, resource sharing, and sophisticated process management.

2. **Application Software:**

Application software is defined as programs that solve a specific business need.

Application in this area process business or technical data in a way that facilitates business operation or management technical decision making. In addition to convention data processing application, application software is used to control business function in real time.

3. **Engineering and Scientific Software:**

This software is used to facilitate the engineering function and task. however modern application within the engineering and scientific area are moving away from the conventional numerical algorithms. Computer-aided design, system simulation, and other interactive applications have begun to take a real-time and even system software characteristic.

4. **Embedded Software:**

Embedded software resides within the system or product and is used to implement and control feature and function for the end-user and for the system itself. Embedded software can perform the limited and esoteric function or provided significant function and control capability.

5. **Product-line Software:**

Designed to provide a specific capability for use by many different customers, product line software can focus on the limited and esoteric marketplace or address the mass consumer market.

6. **Web Application:**

It is a client-server computer program which the client runs on the web browser. In their simplest form, Web apps can be little more than a set of linked hypertext files that present information using text and limited graphics. However, as e-commerce and B2B application grow in importance. Web apps are evolving into a sophisticate computing environment that not only provides a standalone feature, computing function, and content to the end user.

7. Artificial Intelligence Software:

Artificial intelligence software makes use of a nonnumerical algorithm to solve a complex problem that is not amenable to computation or straightforward analysis. Application within this area includes robotics, expert system, pattern recognition, artificial neural network, theorem proving and game playing.

Software Myth:

The development of software requires dedication and understanding on the developers' part. Many software problems arise due to myths that are formed during the initial stages of software development. Unlike ancient folklore that often provides valuable lessons, software myths propagate false beliefs and confusion in the minds of management, users and developers.

Management Myth:

<p>The members of an organization can acquire all-the information, they require from a manual, which contains standards, procedures, and principles;</p>	<p>Standards are often incomplete, inadaptable, and outdated.</p> <p>Developers are often unaware of all the established standards.</p> <p>Developers rarely follow all the known standards because not all the standards tend to decrease the delivery time of software while maintaining its quality.</p>
<p>If the project is behind schedule, increasing the number of programmers can reduce the time gap.</p>	<p>Adding more manpower to the project, which is already behind schedule, further delays the project.</p> <p>New workers take longer to learn about the project as compared to those already working on the project.</p>
<p>If the project is outsourced to a third party, the management can relax and let the other firm develop software for them.</p>	<p>Outsourcing software to a third party does not help the organization, which is incompetent in managing and controlling the software project internally. The</p>

	organization invariably suffers when it out sources the software project.
--	---

User Myth:

Brief requirement stated in the initial process is enough to start development; detailed requirements can be added at the later stages.	Starting development with incomplete and ambiguous requirements often lead to software failure. Instead, a complete and formal description of requirements is essential before starting development. Adding requirements at a later stage often requires repeating the entire development process.
Software is flexible; hence software requirement changes can be added during any phase of the development process.	Incorporating change requests earlier in the development process costs lesser than those that occurs at later stages. This is because incorporating changes later may require redesigning and extra resources.

Developer Myth:

Software development is considered complete when the code is delivered.	50% to 70% of all the efforts are expended after the software is delivered to the user.
The success of a software project depends on the quality of the product produced.	The quality of programs is not the only factor that makes the project successful instead the documentation and software configuration also play a crucial role.
Software engineering requires unnecessary documentation, which slows	Software engineering is about creating quality at every level of the software

down the project.	project. Proper documentation enhances quality which results in reducing the amount of rework.
The only product that is delivered after the completion of a project is the working program(s).	The deliverables of a successful project includes not only the working program but also the documentation to guide the users for using the software.
Software quality can be assessed only after the program is executed.	The quality of software can be measured during any phase of development process by applying some quality assurance mechanism. One such mechanism is formal technical review that can be effectively used during each phase of development to uncover certain errors.

Software engineering - Layered technology

- Software engineering is a fully layered technology.
- To develop a software, we need to go from one layer to another.
- All these layers are related to each other and each layer demands the fulfillment of the previous layer.

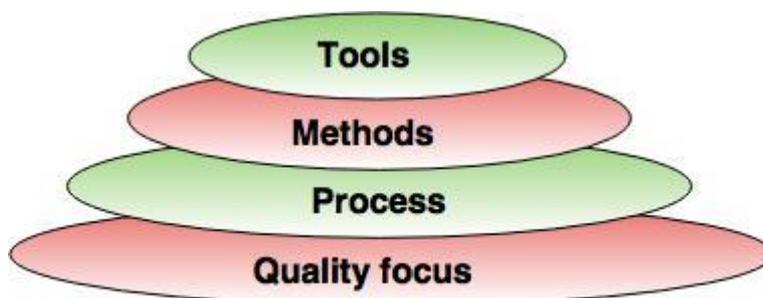


Fig. - Software Engineering Layers

The layered technology consists of:

1. Quality focus

The characteristics of good quality software are:

- Correctness of the functions required to be performed by the software.
- Maintainability of the software
- Integrity i.e. providing security so that the unauthorized user cannot access information or data.
- Usability i.e. the efforts required to use or operate the software.

2. Process

- It is the base layer or foundation layer for the software engineering.
- The software process is the key to keep all levels together.
- It defines a framework that includes different activities and tasks.
- In short, it covers all activities, actions and tasks required to be carried out for software development.

3. Methods

- The method provides the answers of all 'how-to' that are asked during the process.
- It provides the technical way to implement the software.
- It includes collection of tasks starting from communication, requirement analysis, analysis and design modelling, program construction, testing and support.

4. Tools

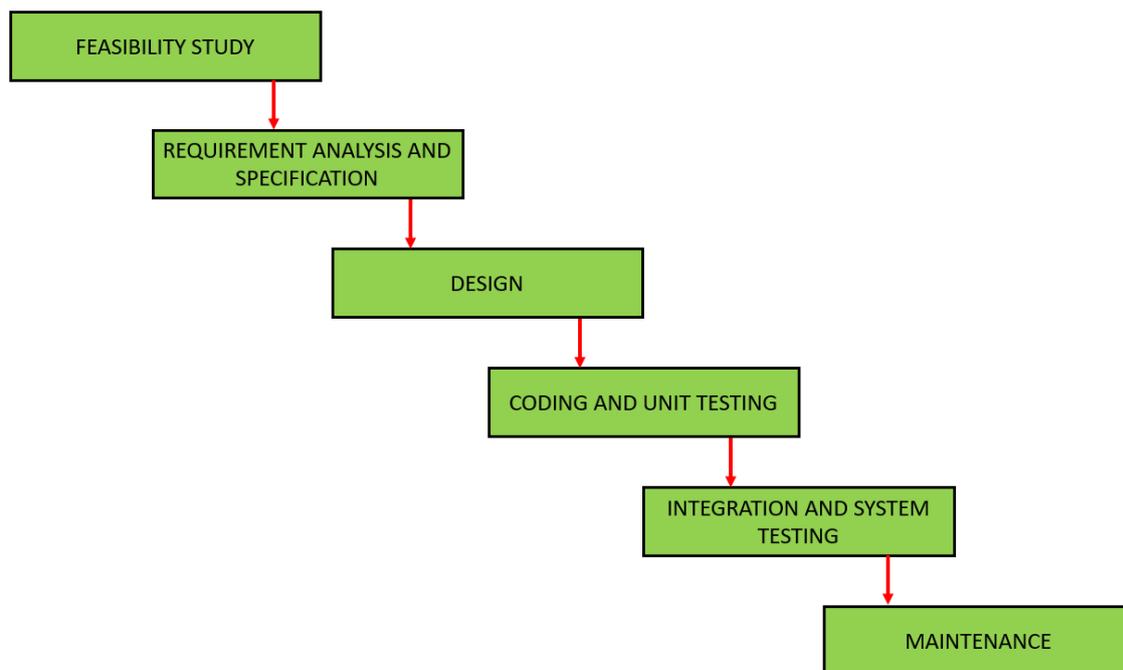
- The software engineering tool is an automated support for the software development.
- The tools are integrated i.e the information created by one tool can be used by the other tool.

For example: The Microsoft publisher can be used as a web designing tool.

Classical Waterfall Model:

Classical waterfall model is the basic **software development life cycle** model. It is very simple but idealistic. Earlier this model was very popular but nowadays it is not used. But it is very important because all the other software development life cycle models are based on the classical waterfall model.

Classical waterfall model divides the life cycle into a set of phases. This model considers that one phase can be started after completion of the previous phase. That is the output of one phase will be the input to the next phase. Thus the development process can be considered as a sequential flow in the waterfall. Here the phases do not overlap with each other. The different sequential phases of the classical waterfall model are shown in the below figure:



Let us now learn about each of these phases in brief details:

1. **Feasibility Study:** The main goal of this phase is to determine whether it would be financially and technically feasible to develop the software. The feasibility study involves understanding the problem and then determine the various possible strategies to solve the problem. These different identified solutions are analyzed based on their benefits and drawbacks, The best solution is chosen and all the other phases are carried out as per this solution strategy.

-
2. **Requirements analysis and specification:** The aim of the requirement analysis and specification phase is to understand the exact requirements of the customer and document them properly. This phase consists of two different activities.
 - **Requirement gathering and analysis:** Firstly all the requirements regarding the software are gathered from the customer and then the gathered requirements are analyzed. The goal of the analysis part is to remove incompleteness (an incomplete requirement is one in which some parts of the actual requirements have been omitted) and inconsistencies (inconsistent requirement is one in which some part of the requirement contradicts with some other part).
 - **Requirement specification:** These analyzed requirements are documented in a software requirement specification (SRS) document. SRS document serves as a contract between development team and customers. Any future dispute between the customers and the developers can be settled by examining the SRS document.
 3. **Design:** The aim of the design phase is to transform the requirements specified in the SRS document into a structure that is suitable for implementation in some programming language.
 4. **Coding and Unit testing:** In coding phase software design is translated into source code using any suitable programming language. Thus each designed module is coded. The aim of the unit testing phase is to check whether each module is working properly or not.
 5. **Integration and System testing:** Integration of different modules are undertaken soon after they have been coded and unit tested. Integration of various modules is carried out incrementally over a number of steps. During each integration step, previously planned modules are added to the partially integrated system and the resultant system is tested. Finally, after all the modules have been successfully integrated and tested, the full working system is obtained and system testing is carried out on this.

System testing consists three different kinds of testing activities as described below :

- **Alpha testing:** Alpha testing is the system testing performed by the development team.
- **Beta testing:** Beta testing is the system testing performed by a friendly set of customers.

-
- **Acceptance testing:** After the software has been delivered, the customer performed the acceptance testing to determine whether to accept the delivered software or to reject it.

6. **Maintainence:** Maintenance is the most important phase of a software life cycle. The effort spent on maintenance is the 60% of the total effort spent to develop a full software. There are basically three types of maintenance :

- **Corrective Maintenance:** This type of maintenance is carried out to correct errors that were not discovered during the product development phase.
- **Perfective Maintenance:** This type of maintenance is carried out to enhance the functionalities of the system based on the customer's request.
- **Adaptive Maintenance:** Adaptive maintenance is usually required for porting the software to work in a new environment such as work on a new computer platform or with a new operating system.

Advantages of Classical Waterfall Model

Classical waterfall model is an idealistic model for software development. It is very simple, so it can be considered as the basis for other software development life cycle models.

Below are some of the major advantages of this SDLC model:

- This model is very simple and is easy to understand.
- Phases in this model are processed one at a time.
- Each stage in the model is clearly defined.
- This model has very clear and well understood milestones.
- Process, actions and results are very well documented.
- Reinforces good habits: define-before- design, design-before-code.
- This model works well for smaller projects and projects where requirements are well understood.

Drawbacks of Classical Waterfall Model

Classical waterfall model suffers from various shortcomings, basically we can't use it in real projects, but we use other software development lifecycle models which are based on the classical waterfall model. Below are some major drawbacks of this model:

-
- **No feedback path:** In classical waterfall model evolution of a software from one phase to another phase is like a waterfall. It assumes that no error is ever committed by developers during any phases. Therefore, it does not incorporate any mechanism for error correction.
 - **Difficult to accommodate change requests:** This model assumes that all the customer requirements can be completely and correctly defined at the beginning of the project, but actually customers' requirements keep on changing with time. It is difficult to accommodate any change requests after the requirements specification phase is complete.
 - **No overlapping of phases:** This model recommends that new phase can start only after the completion of the previous phase. But in real projects, this can't be maintained. To increase the efficiency and reduce the cost, phases may overlap.

Evolutionary Process Models:

- Evolutionary models are iterative type models.
- They allow to develop more complete versions of the software.

Following are the evolutionary process models.

1. The prototyping model
2. The spiral model
3. Concurrent development model

1. The Prototyping model

- Prototype is defined as first or preliminary form using which other forms are copied or derived.
- Prototype model is a set of general objectives for software.
- It does not identify the requirements like detailed input, output.
- It is software working model of limited functionality.
- In this model, working programs are quickly produced.

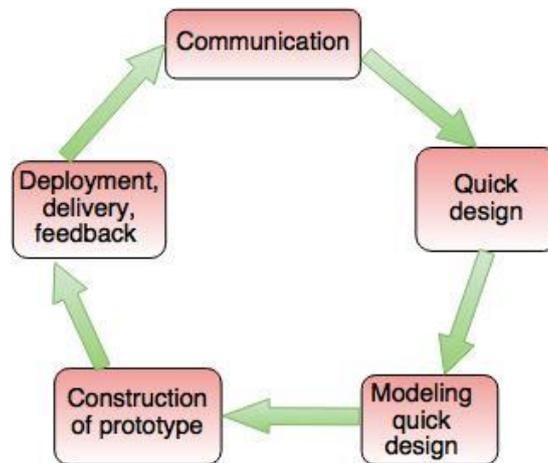


Fig. - The Prototyping Model

The different phases of Prototyping model are:

1. Communication

In this phase, developer and customer meet and discuss the overall objectives of the software.

2. Quick design

- Quick design is implemented when requirements are known.
- It includes only the important aspects like input and output format of the software.
- It focuses on those aspects which are visible to the user rather than the detailed plan.
- It helps to construct a prototype.

3. Modeling quick design

- This phase gives the clear idea about the development of software because the software is now built.
- It allows the developer to better understand the exact requirements.

4. Construction of prototype

The prototype is evaluated by the customer itself.

5. Deployment, delivery, feedback

- If the user is not satisfied with current prototype then it refines according to the requirements of the user.

-
- The process of refining the prototype is repeated until all the requirements of users are met.
 - When the users are satisfied with the developed prototype then the system is developed on the basis of final prototype.

Advantages of Prototyping Model

- Prototype model need not know the detailed input, output, processes, adaptability of operating system and full machine interaction.
- In the development process of this model users are actively involved.
- The development process is the best platform to understand the system by the user.
- Errors are detected much earlier.
- Gives quick user feedback for better solutions.
- It identifies the missing functionality easily. It also identifies the confusing or difficult functions.

Disadvantages of Prototyping Model:

- The client involvement is more and it is not always considered by the developer.
- It is a slow process because it takes more time for development.
- Many changes can disturb the rhythm of the development team.
- It is a thrown away prototype when the users are confused with it.

2. The Spiral model

- Spiral model is a risk driven process model.
- It is used for generating the software projects.
- In spiral model, an alternate solution is provided if the risk is found in the risk analysis, then alternate solutions are suggested and implemented.
- It is a combination of prototype and sequential model or waterfall model.
- In one iteration all activities are done, for large project's the output is small.

The framework activities of the spiral model are as shown in the following figure.

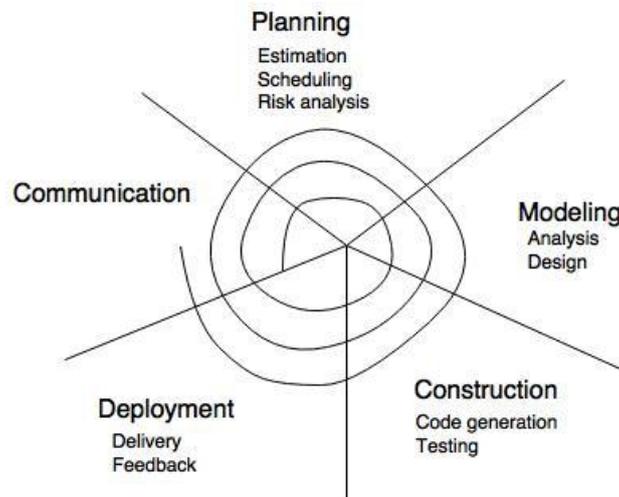


Fig. - The Spiral Model

NOTE: The description of the phases of the spiral model is same as that of the process model.

Advantages of Spiral Model

- It reduces high amount of risk.
- It is good for large and critical projects.
- It gives strong approval and documentation control.
- In spiral model, the software is produced early in the life cycle process.

Disadvantages of Spiral Model

- It can be costly to develop a software model.
- It is not used for small projects.

3. The concurrent development model

- The concurrent development model is called as concurrent model.
- The communication activity has completed in the first iteration and exits in the awaiting changes state.

- The modeling activity completed its initial communication and then go to the underdevelopment state.
- If the customer specifies the change in the requirement, then the modeling activity moves from the under development state into the awaiting change state.
- The concurrent process model activities moving from one state to another state.

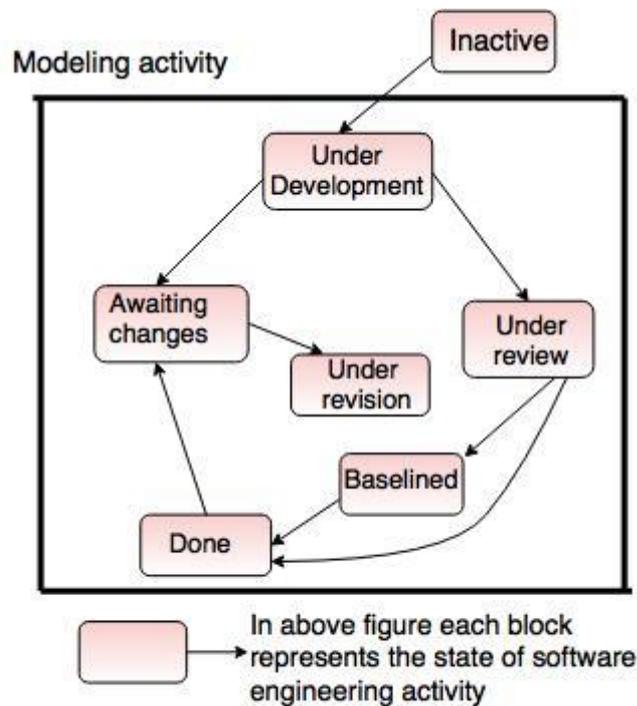


Fig. - One element of the concurrent process model

Advantages of the concurrent development model

- This model is applicable to all types of software development processes.
- It is easy for understanding and use.
- It gives immediate feedback from testing.
- It provides an accurate picture of the current state of a project.

Disadvantages of the concurrent development model

- It needs better communication between the team members. This may not be achieved all the time.
- It requires to remember the status of the different activities.



REFERENCES

TEXT BOOK: Roger S Pressman, “Software Engineering - A Practitioner’s Approach”, Sixth Edition, McGraw Hill International Edition, New York: 2005.

- 1. www.ecomputers.com**
- 2. www.tutorialride.com**
- 3. www.javapoint.com**