CHAPTER 4

Related Work

4.1 NEED FOR GSPM (GENERALIZED SOFTWARE PROCESS MODEL)

Software development methodologies began with traditional approaches that followed the waterfall process model. They then moved toward object-oriented abstractions, which were finally supported by object-oriented methodologies. Now, component-based technology introduces abstraction and lower-level mechanisms but has yet to be organized into a comprehensive software engineering process.

The need to obtain a general model arises as the language independent process can be established and the techniques can then be designed that guarantee the conservation of model correctness during configuration, thus liberating analysts from using manual methods to check the model correctness. Model should be able to eliminate redundancies in the process and facilitate standardization and reuse. Thus it is tried to develop an integrated framework or system to manage variability in process aware information system.

Software organizations have realized that developing the software using reusable components can radically reduce the development effort, cost and accelerate delivery. The absence of a standard process model in this field results in failures of the project at times. From the past studies, it is found that very little effort has been made to identify or measure the software reuse process to imply success. Even though some...
studies have tried to formulate the reference model for reuse, but could not achieve success in their attempt [130,131].

Process modeling is important specially in the context of diversification or acquisition of the projects. The features that play prominent role in making of the model are:

- project objectives should be clearly stated
- project requirements should be stable
- progress of system should be measurable

In the generalized model all stages and activities should be well defined which help in planning and scheduling the project. Verification at each stage where ever needed are to be implemented in order to ensure early detection of errors or misunderstanding. Risk reduction mechanisms should be taken care of as this model supports iteration and reflects real-world practices. It should have a systematic approach and should be better to understand. Changes made in one module should not have any impact on another module. The model should also supports dynamically changing requirements.

Software reuse at its most basic level consists of making use of any existing information, artifact or product when designing and implementing a new system or product. Reuse of resources is dependent upon both similarities and differences between the applications in which the component is being used. [132]

To solve actual problems in an industry, software engineer or a team of engineers must incorporate a development strategy that encompasses the process, methods and tools. A process model for software engineering is chosen based on the nature of the
project and application, the methods and tools to be used, and the controls and deliverables that are required. [133]

4.2 GENERAL MODEL FOR SOFTWARE PROCESS

In the quest to obtain a general process model it is tried to explore the basic functioning of the traditional models. The work starts by studying the traditional models and flows to the analysis of these standard / traditional models.

In the study phase various existing standard models have been studied and the place where they are used beneficially is explored. The advantages and disadvantages of the available models have been studied and a comparison has been made in order to find out as to which area these models lack.

In the analysis phase the requirements of the system have been defined. Thus at the end of this phase the requirements specify what is to be built. It represents the "what" phase.

Next, implementation of the research is carried out using variations of the models in order to obtain a Generalized Software Process (GSPM) Model. In the proposed model, GSPM (Generalized Software Process Model) the users / stakeholders are actively involved whenever it seems essential during the development process. It provides a better system to the users especially considering the aspect of changing requirements specifications by the user. Since in this methodology a working model of the system is provided to the user, thereby the user gets a better understanding of the system while it is being developed. Errors are detected at the initial stages because the system components are made simultaneously. Rapid user feedback is available using this model that leads to providing better solutions. It is followed by testing the
GSPM model and the results are drawn using the graphical analysis tool, Petri-net.[134]

4.2.1 Study and comparison of process models to ensure need for obtaining a generalized model

In this phase the study of various existing standard and traditional models and where they are used was conducted. The standard software development methodologies have been compared. The advantages and disadvantages of the available models were studied and a comparison was made in order to find out the lacking area in each model. This information helps in recognizing which methodologies may be best suited for use in various situations, and presents an approach to software process modeling which is based on behavioral descriptions of software development activities. The use of behavioral descriptions makes it possible to describe the software process at any desired level of abstraction and therefore assists in complying aspects of the process that are poorly understood. It also provides the ability to reason about the way in which the software processes are conducted and is sufficiently accurate to provide a basis for structuring software environments. An attempt has been made to cover all the methodologies that are most applicable to general projects.

All of us are aware that modern software development approaches make heavy use of process models during the development process. Models are being evolved, simplified and/or extended over a longer period of time, which leads to the need of merging different models into one. Models play important role in modern software engineering processes. In the model driven development many different versions of models are created until the software is deployed successfully.

Software systems undergo various steps that account for their inception, initial development, productive operation, upkeep, and retirement from one generation to
another. We categorize and examine various methods for describing how software systems are developed. We start with the background and definitions of traditional software life cycle models that are popular and used extensively in the current software development practices. It is followed by a comprehensive review of the alternative models of software evolution that are in current use as the basis for organizing software engineering projects and technologies. [135, 136]

4.3 SOFTWARE LIFE CYCLE MODELS

A software life cycle is the sequence in which a project specification, prototyping, designing, implementation, testing, and maintenance of software takes place. A life cycle model can either be in a descriptive form or prescriptive form.

Descriptive model describes the history of how a particular software system was developed and it can be used as the basis for understanding and improving software development processes. A prescriptive model on the other hand suggests how a new software system should be developed and can be used as guidelines to organize and structure (define the order) the software development activities. It outlines the documents to produce for the delivery to client.

These two forms serve as a guideline to organize, plan, budget, schedule, staff, manage and to determine which software engineering tools and methodologies will be more appropriate for the particular project. They also help to draw a framework for analyzing or estimating patterns of resource allocation and consumption during the software life cycle (Boehm 1981). They can also serve as a basis for conducting empirical studies to determine what affects software productivity, cost, and overall quality. The software process model maybe defined as a description of a software process, presented from a user perspective. Each stage of the software process is
identified and a model is then developed to represent the intrinsic activities associated within that stage.

Suitability of the software paradigm depends on lot of factors such as:

- nature of the project
- type of application
- tools to be used
- types of controls and documentation required

4.3.1 Waterfall model

Earlier the software development code was written and then debugged. There was no formal design or analysis available. The approach to developing complex hardware systems was well understood so it provided a base for modeling software development.

Waterfall approach was the first process model to be introduced and followed widely in software engineering to ensure success of the project. In this approach, the whole process of software development is divided into separate process phases. The phases in waterfall model are, Requirement Specifications phase, Software Design, Implementation and Testing & Maintenance.

Waterfall model is an approach to development that emphasizes completion of a phase in the development before proceeding to the next phase. If a need is identified
to change, a formal process is followed for making the change. The graphical representation of these phases resemble the downward flow of a waterfall. [106, 107]

The various stages of "The Waterfall Model" are as follows:

**Requirement Analysis & Definition:** Requirements are gathered from the end-user usually by discussion. These requirements are then analyzed for their validity and the possibility of incorporating the requirements in the system under consideration is measured. Finally, a Requirement Specification document is created which serves the purpose of guideline for the next phase of the model.

**System & Software Design:** The requirement specifications from the first phase are studied in this phase and system design is prepared. System design helps to specify the hardware and the systems requirements thereby help in defining overall system architecture. The system design specifications serve as an input for the next phase of the model.

Courtesy: wikipedia.org

Figure 4.1 Waterfall Model
Implementation & Unit Testing: The system is developed in small programs called Units, which are integrated in the next phase. Each unit is developed and tested for its functionality by Unit Testing.

Integration & System Testing: The system is first divided into units that are developed and tested for their functionalities. These units are integrated into a complete system during Integration phase and are tested to check if all modules/units coordinate amongst each other. It is also tested that the system as a whole behaves as per the specifications. After successful testing of the software, it is then delivered to the customer / client.

Operations & Maintenance: Generally, problems with the developed system occur after its implementation. So the issues related to the system are solved only after the deployment of the system. Not all the problems come into picture directly but they arise gradually with time and they need to be solved; hence this process is known as Maintenance.

In waterfall model phases take place in sequential manner before proceeding to the next phase. The salient attributes of the waterfall model are:

- it is a formal method
- it has top-down development
- it is composed of independent phases that are to be done sequentially
- It is used in varied ways
4.3.1.1 Where to use the Waterfall model

The application of the waterfall model should be limited to situations where the requirements and the implementation of those requirements are very well understood and are not expected to change or evolve over the life of the project. Project risks should also be relatively low, when using Waterfall model. [108, 109, 110]

4.3.2 Incremental model or Iterative enhancement model

In this approach the model is designed, implemented and tested incrementally, that is, functionality is delivered in small increments and added till the product is completed. The product is considered to be complete when all of its requirements are fulfilled. This model combines the elements of the waterfall model along with the iterative nature of prototyping. This model uses a series of waterfall model. [213,218-219]
In this model the product is decomposed into a number of components. Each of the components are designed and built separately and delivered to the client when it is complete. This eliminates a long development time. Overall functionality of the product is delivered in small increments. Each increment uses the waterfall approach.

Iterative development segments the deliverable system functionality into iterations. Each linear sequence produces a deliverable product of the software. The first increment is often a core product in which basic requirements are addressed but the supplementary features are not delivered. The core product is used and evaluated by the customer and reported. Based upon which a plan is developed for the next increment. This process is repeated until the complete product is produced. [111, 112]
4.3.2.1 Where to use the Incremental model

When developing the complete system is too risky, then the incremental development should be considered. It is good for projects where requirements are known at the beginning, and need functionality early in the project. Each cycle produces a working system, so it is also advantageous for projects whose continued funding is not assured. It is best used on low to medium-risk programs / projects. If the risks are too high to build a successful system then spreading the development over multiple cycles may lower the risks to a more manageable level.

It has been used as **Synchronize and Stabilize Model** by **Microsoft** (introduced in 1997). The work is divided in 2 or 3 builds. The first build consists of the most critical features, the second build with next critical ones, etc. Each build is carried out by a number of small teams working in parallel. At the end of each day, all the teams synchronize by putting the partially completed components together followed by testing and debugging the resulting product. [173, 174]

4.3.3 RAD model

Rapid Application Development (RAD) is a software development methodology, which includes iterative development and the creation of prototypes. A RAD approach enables faster development and facilitating application maintenance. **James Martin** introduced it in 1991. Martin's methodology involves iterative development and the construction of prototypes. With the previous methodologies the problem was that applications took very long to build and practically the requirements changed before the system was complete, resulting in inadequate or even unusable systems.[168, 169]
RAD (Rapid Application Development) is a concept that produces products to be developed faster and of higher quality through the process of:

- gathering requirements using workshops or focus groups
- prototyping and user testing of designs
- re-use of software components
- design improvements are kept pending until the next product version
- communication is kept informal amongst team members

Courtesy: cavehill.uwi.edu

Figure 4.3 RAD Model
The iterations in RAD are usually 3 months or less, causing rapid succession of deliveries. There can be multiple teams each working on one component. Each team takes 3 months or less. The release combines the different components together.

Rapid Application Development methodology supports that development be undertaken by small, experienced teams using CASE (Computer Aided Software Engineering) tools to enhance their productivity.

Some of the largest software vendors such as Microsoft and IBM do not broadly employ RAD in the development of their core products and they still largely rely on traditional waterfall methodologies with some extent of spiraling.

4.3.3.1 Where to use the RAD model

It is used in strong collaborative environments and where dynamic gathering of the requirements is necessary. Stakeholder / client can actively participate in prototyping, writing test cases and performing unit testing. [170, 171, 172]

4.3.4 Prototyping model

It involves the activity of creating prototypes, that is, the incomplete versions of the software program are developed. Prototype simulates only a few characteristics of the features of the program, and may be completely different from the final implementation.

Purpose of a prototype is to allow users / client / stakeholders to evaluate software developer’s proposals for the design of the final product by actually using them. The major purpose is to help customers and developers understand the requirements for the system being developed. [149, 150, 151, 152]
The system is constantly refined and rebuilt, as we do not know all the future requirements. As we learn what the client wants, we build the software and this allows the team of developers to change and add features throughout the building process that were not considered in the requirements and design phase.

The beta system is sent to the customer sites. As users work with the system, they look up for any possible modifications needed. The existing features or requirement for even some new features can be considered and request can be made to the
developers for adding these features. Developers then start working to build a system to be released that fully meets the needs of the user / client / stakeholders. [153 - 156]

4.3.4.1 Where to use the Prototyping model

It works best with systems that will have more or sufficient interaction with the users. For example an ATM system in which the user is given choices and is simple and understandable for most people. More emphasis is on the user interface design (UID). This model makes greater involvement of users in the development process and reduced documentation and fast development time.

Prototyping may be used with waterfall approach and is useful to demonstrate technical feasibility when the technical risk is high. It can also be used to better understand and extract user requirements. The basic purpose is to limit cost by understanding the problem before committing more resources. It is usually used on medium to high-risk systems. The evolutionary model should be considered for systems where requirements are not all known, but are expected to evolve. It is more applicable to new systems than upgrading existing software and for software intensive systems, also to systems that have a large number of diverse users, programs that have rapidly changing software technology, developing an unprecedented system and where limited capability is needed quickly. [157 - 160]

4.3.5 Spiral model

The spiral model is a software development process, which combines elements of design and prototyping, to unite advantages of top-down and bottom-up concepts.
This model combines the features of the prototyping model and the waterfall model. The spiral model is intended for large, expensive and complicated projects.

**Barry Boehm** defined the spiral model in his 1988 article "A Spiral Model of Software Development and Enhancement". This model was the first model to explain iterative matters of development. The iterations were typically 6 months to 2 years long. Each phase starts with a design goal and ends with the client reviewing the progress. Analysis and engineering efforts are applied at each phase of the project, keeping the goal of the project in mind. [113 -116]

The steps used in the spiral model can be summarized as follows:

New system requirements are defined in detail. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system. A preliminary design is created for the new system. First prototype of the new system is constructed from the preliminary design. Second prototype is evolved as follows: [117 - 120]

1. evaluating the first prototype for its strengths, weaknesses, and risks
2. defining the requirements for the second prototype
3. planning and designing the second prototype
4. constructing and testing the second prototype
Figure 4.5 Spiral Model

In the above diagram the radial dimension is cumulative cost to complete the cycle. Angular dimension is progress within this cycle. The four quadrants represent four standard periods in each cycle as follows:

1. goal setting

2. risk minimization
4.3.5.1 Where to use the Spiral model

The spiral model is often used in the large projects. For smaller projects, the concept of agile software development is being considered as an alternative. The US military has adopted the spiral model for its Future Combat Systems program. It has become quite popular among ADE (Aerospace, Defense and Engineering) specialists, and is not so familiar among business developers. It is particularly useful in ADE projects, because they are risky in nature. Business projects are conventional and tend to use mature technology to work with applications, which require much computation as in decision support systems.

The spiral method should be considered for projects where risks are high, the requirements need to be refined, and the user needs are very important. Game development is a main area where the spiral model is used and needed, because of the size and the constantly shifting goals of the project. Online customer support system where it is not well understood what services customers want or can accomplish online is another useful area, to be considered. [164 - 167]

The spiral model is mostly used in large projects. The US military adopted the spiral model for its Future Combat Systems program. The FCS project was canceled after six years (2003 - 2009), it had a 2 year iteration (spiral). FCS should have resulted in 3 consecutive prototypes (one prototype per spiral - every 2 years). It was canceled in May, 2009. [31, 121, 122]
### 4.4 Comparison Table of the Traditional Models

<table>
<thead>
<tr>
<th>Model Name</th>
<th>Advantages of the Model</th>
<th>Shortcomings of the Model</th>
</tr>
</thead>
</table>
2. Systems have well-defined and understood requirements  
3. Reuse is common and assumes that the developers have worked on similar systems in the past and are experts in the application domain.  
4. Risk of project failure is low.  
5. Provides a template into which methods for analysis, design, coding, testing and support can be placed.  
6. Widely used procedural model for software engineering.  
7. Have clear project objectives. | 1. Once project requirements are gathered in the first phase, no formal way to make changes to the project.  
2. Poor choice for software development projects where requirements are not well known or understood by development team.  
3. Not good for complex projects or projects that take more time to complete. The project failure rate is high.  
4. Real project rarely follows the sequential flow that the model proposes.  
5. Requires all requirement explicitly, but it is often difficult for the customer to state all |
<table>
<thead>
<tr>
<th>Model Name</th>
<th>Advantages of the Model</th>
<th>Shortcomings of the Model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8. Has a stable project requirement.</td>
<td>requirement explicitly.</td>
</tr>
<tr>
<td></td>
<td>9. System progress is measurable.</td>
<td>6. Working version of the program is not available until late in the project. Hence, time consuming.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7. Little scope for iteration. Thus, not good for complex and object oriented projects.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8. Difficulty in responding to changes results to high amount of risk and uncertainty.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9. Assumes requirements be frozen before the design begins which is possible for systems that automate existing manual system. For new system, it is difficult, as user does not know the requirements.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10. Choosing the hardware during requirement specification leads to problem as a large project might take a few years to complete. Selecting hardware early, will implement hardware technology</td>
</tr>
<tr>
<td>Model Name</td>
<td>Advantages of the Model</td>
<td>Shortcomings of the Model</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Incremental</td>
<td>1. System is developed and delivered in increments. Software is generated quickly and early during the software life cycle.</td>
<td>1. Each phase is rigid and do not overlap each other.</td>
</tr>
<tr>
<td></td>
<td>2. Useful when staff is not available for complete implementation by the business deadline.</td>
<td>2. System architecture problem because not all requirements are gathered together for the entire software life cycle.</td>
</tr>
<tr>
<td></td>
<td>3. More flexible and less costly to change scope and requirements.</td>
<td>3. Each additional build has to be incorporated into the existing structure.</td>
</tr>
<tr>
<td></td>
<td>4. Easier to test and debug for smaller iteration and to manage risk.</td>
<td>4. Addition of succeeding builds must be easy.</td>
</tr>
<tr>
<td></td>
<td>5. Each iteration is easily managed.</td>
<td>5. Design errors become part of the system and are difficult to remove.</td>
</tr>
<tr>
<td></td>
<td>6. Easier to managed milestone.</td>
<td>6. Clients may want to change requirements.</td>
</tr>
<tr>
<td>Model Name</td>
<td>Advantages of the Model</td>
<td>Shortcomings of the Model</td>
</tr>
<tr>
<td>------------</td>
<td>-------------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td></td>
<td>7. Small number of programmers / developers may start working.</td>
<td>1. Cost of product is not known.</td>
</tr>
<tr>
<td></td>
<td>8. Clients can work on the system and provide feedback.</td>
<td>2. It is difficult for users to commit the time required for success of the RAD process.</td>
</tr>
<tr>
<td></td>
<td>9. It breaks down the problem into sub-problems thus dealing with reduced complexity.</td>
<td>3. It may increase the overall lifetime costs in operation, support and maintenance.</td>
</tr>
<tr>
<td>RAD</td>
<td>1. It generally incorporates short development cycles thus users see the product quickly.</td>
<td>4. More speed and lower cost may lead to lower overall system quality</td>
</tr>
<tr>
<td></td>
<td>2. It enables faster and updated version by systematically eliminating redundant steps or using the prototype methods.</td>
<td>5. May have inconsistent internal</td>
</tr>
<tr>
<td></td>
<td>3. Cost overruns and meeting the time constraints are an advantage.</td>
<td></td>
</tr>
<tr>
<td>Model Name</td>
<td>Advantages of the Model</td>
<td>Shortcomings of the Model</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>changes. Ability to rapidly change system design as demanded by users.</td>
<td>designs within and across systems.</td>
</tr>
<tr>
<td></td>
<td>5. It makes an overall reduction in project risk.</td>
<td>6. Difficult to reuse the module for future systems.</td>
</tr>
<tr>
<td></td>
<td>6. Pareto’s 80 - 20 Rule results in reducing the costs to create a custom system.</td>
<td>7. Lack of scalability.</td>
</tr>
<tr>
<td></td>
<td>7. It is useful for projects in which user requirements are uncertain or imprecise.</td>
<td>8. High cost of commitment by the user.</td>
</tr>
<tr>
<td></td>
<td>8. Errors and omissions are detected earlier in prototypes than in system models.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9. It reduces risk because technical solution is tested iteratively.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10. Where speed of development is important.</td>
<td></td>
</tr>
<tr>
<td>Prototyping</td>
<td>1. Application software is a</td>
<td>1. Process can be too fast to properly</td>
</tr>
<tr>
<td>Model Name</td>
<td>Advantages of the Model</td>
<td>Shortcomings of the Model</td>
</tr>
<tr>
<td>------------</td>
<td>-------------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td></td>
<td>&quot;throw-away.&quot; If a new version is needed, it is developed from scratch.</td>
<td>test the system.</td>
</tr>
<tr>
<td></td>
<td>2. The work will be faster and efficient if developers will collaborate.</td>
<td>2. Most of the models presented in the early stage are not complete. The prototype changes from time to time.</td>
</tr>
<tr>
<td></td>
<td>3. It is created using user feedback.</td>
<td>3. Temptation for developers to create a prototype and stick to it even though it has flaws. Possibility of a designer flaw. Flawed software could mean losses of important resources.</td>
</tr>
<tr>
<td></td>
<td>4. Early visibility of the prototype helps users to know the final system as it is being made.</td>
<td>4. Integration can be very difficult.</td>
</tr>
<tr>
<td></td>
<td>5. Cost effective (development cost is reduced)</td>
<td>5. A system inadequate for overall organization needs could be produced.</td>
</tr>
<tr>
<td></td>
<td>6. Increased system development speed</td>
<td>6. Structure of system can be damaged since many changes could be made.</td>
</tr>
<tr>
<td></td>
<td>7. Potential risks associated with the delivery of the system can be refined.</td>
<td>7. Not suitable for large applications.</td>
</tr>
<tr>
<td></td>
<td>8. Various aspects can be tested</td>
<td></td>
</tr>
<tr>
<td>Model Name</td>
<td>Advantages of the Model</td>
<td>Shortcomings of the Model</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td><strong>Spiral</strong></td>
<td>and quicker feedback can be got from the user</td>
<td>8. Over long period of time, consumer interest could be lost and cancellation of product due to a lack of a market (for commercial products) can take place.</td>
</tr>
<tr>
<td></td>
<td>1. Iterative approach allows development to begin even when all the system requirements are not known.</td>
<td>1. Risk assessment could cost more than development.</td>
</tr>
<tr>
<td></td>
<td>2. Each prototype is tested and user feedback is used to make sure the project is on track.</td>
<td>2. Risk analysis requires high expertise.</td>
</tr>
<tr>
<td></td>
<td>3. Risk analysis step ensures the project stays on track even if requirements change.</td>
<td>3. Success of project is dependent on the risk analysis phase.</td>
</tr>
<tr>
<td></td>
<td>4. This model combines the features of the prototyping model and the waterfall model.</td>
<td>4. Does not work well for smaller projects.</td>
</tr>
<tr>
<td></td>
<td>5. It is favored for large, expensive, and complicated</td>
<td>5. Highly customized (limits re-usability)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6. Made separately for each application.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7. Risk of not meeting budget or</td>
</tr>
<tr>
<td>Model Name</td>
<td>Advantages of the Model</td>
<td>Shortcomings of the Model</td>
</tr>
<tr>
<td>------------</td>
<td>-------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6. Estimates (budget, schedule, etc.) become more realistic.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7. It uses a stepwise approach.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8. Any risks are detected earlier than in other process models and measures can be taken to handle them.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8. Complex and difficult to follow strictly.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9. Applicable only to large systems.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10. Applying already developed product to the needs of a new customer by small changes, involves high risk of making the correct product.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11. Spiral model is not much used, although it has many advantages and could have even more if the risk assessment phases would be reformed to the necessary amount.</td>
</tr>
</tbody>
</table>

Table 4.1 Comparison Table

**4.5 SHORTCOMINGS OF VARIOUS SOFTWARE DEVELOPMENT MODELS**

**Waterfall Model:** Once the project requirements are gathered there is no formal way to make changes to the project. This model is not a good choice for those software development projects where requirements are not well known or understood by the development team. Also, it is not suitable for complex projects that take up more time.
to complete. The project failure rate using Waterfall model is high. In this model all requirements are needed explicitly, but it is often difficult for the customer to state all requirements explicitly. Working version of the program is not available until late in the project. Hence, it is a time consuming approach. This model assumes requirements to be frozen before the design begins which is possible only for systems that automate existing manual system.

**Incremental Model:** In this model each phase is rigid and does not overlap each other. Each additional build has to be incorporated into the existing structure. Addition of succeeding builds is easy. Design errors become part of the system and are difficult to remove.

**RAD (Rapid Application development) Model:** Using this process model, cost of product is not known in the beginning. It is difficult for the users to commit time required for success of the RAD process. It may increase the overall costs in operation, support and maintenance. High speed and lower cost may lead to reduce overall system quality; also it might have inconsistent internal designs within and across the systems. This makes it difficult to reuse the developed module for future systems. It has lack of scalability. It needs high cost of commitment by the user.

**Prototyping Model:** In this model the process can be too fast to be able to properly test the system. Most of the models presented in the early stage are not complete. The prototype changes from time to time. Developers would like to stick to the prototype even though it has flaws. Integration can turn out to be very difficult. Structure and design of the original system can be damaged or changed since many changes could be made. It is not suitable for large applications.

**Spiral Model:** In this model the risk assessment could cost more than the development. Risk analysis requires and demands high form of expertise. Success of
the project is dependent on the risk analysis phase. It is not found to be suitable for smaller projects. It is highly customized and is made separately for each application. It has risk of not meeting the budget or the schedule. It is a complex approach and is difficult to follow strictly. It is well applicable and suitable only to the large systems. Making changes to a developed product, according to the needs of a new customer, by making few changes, involves high risk of developing the correct product.

The above study helps to analyse the advantages and shortcomings prevailing to the traditional and standard models that have been used extensively in the industry for commercial use, in the defence area, in the various fields whatsoever. It strengthened my view to form a generalised model, which should overcome the existing shortcomings and help to generate the process workflow in a smooth manner with ease. In this research work we consider a new method for software life cycle and architecture of the new model has been constructed. Based on the above study new models have been designed. [134] - [148]

### 4.6 EFFICIENT PARADIGM FOR IMPLEMENTATION OF PROCESS MODELS IN PROJECTS

Software development techniques make extreme use of traditional software development models during the development process. Models have been used since a long time for many years and the study of merits and demerits studied confirm the need of merging different models into one.

We are aware that models play vital role in existing software engineering processes which help in development of different versions of models until the software is deployed. Software systems undergo various methods for initial development, productive operation, upkeep, change and maintenance.
4.6.1 Model for flexible requirements

Projects with hazy requirements can use this new model. This model integrates waterfall model, and incremental model to get advantage of the software developed and delivered in increments. It will result in flexible and less costly affair to deal with such situations.

A new process and strategy is planned and designed for catering the flexible requirements. In this proposed model for projects, the process starts with the feasibility study of the project under consideration and moves on to the requirements analysis and specification phase. After final analysis is done the confirmation is sought from the client and scope for re-discussion is kept open.

Meanwhile, a quick design is developed and communicated to the client. This sets the confirmation at this stage that nothing more or new is needed further. Next, the process for construction of the code starts followed by the deployment of the project at the client site and a feedback is sought which helps in providing further assistance or maintenance.
Figure 4.6 Proposed Model for Projects with Flexible Requirements

Feasibility Study

Requirements Analysis & Specification

Communication with Client

Making Quick Design

Communication with Client

Construction of the Code

Deployment

Feedback
Thus a systematic and disciplined approach is used here in a sequential manner for modelling the process which proves to be beneficial in projects with flexible requirements.

4.6.1.1 Implementation of proposed model for projects with flexible requirements in courseware development

We have implemented the proposed models on the activity of developing courseware at Birla Institute of Technology, Mesra, Ranchi, Extension Center Jaipur by the faculty in various departments. Courseware refers to the contents, which a student requires for his study. This material needs to be concise with the support of instructional delivery system. The structure of the courseware will include the basic architecture as follows:

Figure 4.7 Structure of a Courseware
The architecture plays a vital role and is used to denote the specific way that the components will be organized. The contents will be course or subject specific. Media will be representing the physical basis like, text, audio, video, etc. related activities should be able to make effective, successful and quality courseware. A systematic, well-defined and qualitative approach is required which shows what is to be done and in which sequence.

We have used the above proposed model which is suitable for flexible projects. While dealing with many subjects and different faculties, we need to have a flexible approach in developing a coursework. In developing courseware usually the requirements are not very well known or fixed initially hence practically, if we integrate the Waterfall model along with the Incremental model then we have the advantage of the software being developed and delivered in increments.

The application of the waterfall model should be limited to situations where the requirements and the implementation of those requirements are very well understood and are not expected to change or evolve. Project risks should also be relatively low. Integrating the waterfall and Incremental model it will be flexible and less costly to change scope and requirements. This will also help the clients to work on the system and provide feedback. If developing the complete system is too risky, then also it will be beneficial. Each cycle will produce a working system, so it is also advantageous for projects whose continued funding is not assured. If the risks are too high to build a successful system then spreading the development over multiple cycles may lower the risks to a more manageable level.

In the proposed model it has been tried to make the data flow functioning along with the consensus of the client at the stage where it is crucial to decide and then proceed. This helps in maintaining the quality assurance. Various phases and activities involved are discussed in detail within. A systematic and disciplined approach is used
in a sequential manner for designing the course. It deals with the implementation of courseware development model using an efficient paradigm developed. Each phase is carried out using the proposed model in the following manner as given below.

Initially the feasibility study is done in order to plan for a successful courseware development project. The development team should be able to understand and specify the work to be done, the resources required, the risk to be incurred, the tasks to be performed and the schedule, which is to be followed along with the cost effectiveness. This is followed by the analysis phase. It is the process that defines what needs to be learned in the course by the student. Using the feedback from the students and academicians data is gathered from the real environment and the actual need for the required material is established. Hence the contents preferred and their limitations can be defined in a better manner. This will help to design the topics and subtopics of the course. It will also help in identifying the prerequisites skills of the students to learn the course effectively. Relevant information is then gathered from textbooks and references to achieve an effective material for the course. The courseware requirement analysis is conducted using a well-defined process. This activity is repeated until an acceptable requirement is produced. Courseware design is the process of specifying how the contents are to be arranged and presented. Designing a courseware should be such that ensures competent material is produced which enhances knowledge and skills of the student. It is the most crucial design activity. The architecture should be effective.

Later the testing is performed to make sure that the courseware is developed according to the specifications and the requirements. Once various courseware components are developed they are then integrated. Unit testing can begin as soon as the module is complete. Once all the components are integrated then the system needs to be tested again using various integration-testing techniques. Later the courseware can be developed and tested and delivered to the academicians and
students. Continuous feedback is required to enhance and update the system. This feedback will help in correcting errors, perfection of courseware and adapting new contents.

After implementation and analysis of implementation of the traditional process models for their shortcomings, proposed work is based on the elimination of the shortcomings and introducing new concepts. The above implementation gives the opportunity to analyse the advantages and shortcomings prevailing to the traditional and standard models which are and have been used extensively in the industry for commercial use, in the defence area, in the various fields whatsoever; and in this path of research it strengthens the view to form a generalised model which shall overcome the shortcomings and help to generate the workflow in a smooth manner with ease.
4.6.2 Model for dynamic and collaborative projects

Projects with strong collaborative environment and with dynamic requirements gathering can use RAD model integrated with prototyping model and waterfall model so that it becomes easier to demonstrate technical feasibility when the technical risk is high. It will be beneficial for medium to high-risk systems.

Figure 4.8 Proposed Model for Dynamic and Collaborative Projects
After studying and analyzing the traditional process models for their shortcomings, proposed work is based on the elimination of the shortcomings and introducing new concepts. In projects usually the requirements are not very well known or fixed initially hence practically, if we integrate the Waterfall model along with the Incremental model then we have the advantage of the software being developed and delivered in increments. In the proposed model also it has been tried to make the data flow functioning along with the consensus of the client at the stage where it is crucial to decide and then proceed.

In projects which use strong collaborative environment and where dynamic gathering of requirements is the need we can use RAD model and integrate it with Prototyping model and with the Waterfall model as it will be useful to demonstrate technical feasibility when the technical risk is high. This model can make greater involvement of users in the development process and reduce documentation and will provide fast development time. Prototyping will be used to better understand and extract user requirements. The basic purpose is to limit cost by understanding the problem before committing more resources. It is usually used on medium to high-risk systems.

It will be useful for systems that have large number of diverse users, programs that have rapidly changing software technology, developing an unprecedented system and where limited capability is needed quickly. It will combine elements of design and prototyping, to unite advantages of top-down and bottom-up concepts. Using spiral will also help to combine the features of the prototyping model and the waterfall model. This model will be intended for large, expensive and complicated projects.
4.6.2.1 Evaluation to produce an efficient paradigm for dynamic and collaborative projects

It deals with the learning and examining various standard software engineering process models to evaluate the need for an efficient and improved form of new software development method. The benefits and shortcomings of the existing traditional models will be considered for evaluation. Here it is tried to introduce and evaluate traditional software development processes. This information will help to recognize which process may be appropriate for use in different circumstances.

Initially the feasibility study is done in order to plan for a successful courseware development project involving more than one courseware development at a time. Here also the development team should be able to understand and specify the work to be done, the resources required, the risk to be incurred, the tasks to be performed and the schedule, which is to be followed along with the cost effectiveness. This is followed by the quick planning and modeling the design to provide a prototype of the project. Here we emphasize on preparing the prototype for all the different modules in hand at the same time. All the modules are sent to the client for evaluation and feedback simultaneously.

Later the testing is performed to make sure that the developed project functions according to the specifications and the requirements. Once all the courseware components are developed they are then integrated. Unit testing is performed within the completed modules. Once all the components are integrated then the system is tested using different integration-testing techniques. Continuous feedback is expected to enhance and update the system. This feedback will help in correcting errors, perfection of courseware and adapting new contents.
Projects where all the requirements are not known, but are expected to evolve the evolutionary model can be integrated along with the spiral model and will be useful for systems that have large number of different users, where programs rapidly changing software technology, unprecedented system and where limited capability is needed quickly. Taking use of spiral model will also help to combine the features of the prototyping model and the waterfall model. This model can be intended for large, expensive and complicated projects.

This model was implemented at Birla Institute of Technology, Mesra, Ranchi, Extension Center Jaipur by the Time Table team, which always collaborates different departments and co-ordinates the complete college time table for the post graduates and under graduates.

The above study gives the opportunity to analyse the advantages and shortcomings prevailing to the traditional and standard models which are and have been used extensively in the industry for commercial use, in the defence area, in the various fields whatsoever; and in this path of research it strengthens the view to form a generalized model which shall overcome the shortcomings and help to generate the workflow in a smooth manner with ease. Next step of the work will consider a new method for software life cycle model and derivation of the architecture of the new model.