International Journal of Engineering Research in Computer Science and Engineering (IJERCSE) Vol 2, Issue 7,July 2015

Role of Project Domain and Project Type attributes for Software Development

[1] Prof. Ashwini Gharde ,[2] Dr Ashish B.Sasankar
[1] Department of Master of Computer Applications Rajiv Gandhi College of Engineering and Reaserch Nagpur, India
[2] P.G.Department of Computer Science G.H.Raisoni Institute of Information Technology Nagpure,India
[1] ashwini.gharde@gmail.com,[2] ashish.sasankar@raisoni.net

Abstract- **Software worth billions and trillions of dollars have gone waste in the past due to lack of proper techniques used for developing software resulting into software crisis. Historically , the processes of software development has played an important role in the software engineering. A number of life cycle models have been developed in last three decades. This paper is an attempt to identify attributes that are critical to software process model for a software project. The objective is to identify the project domain(area) and project type(scale).**

Key Terms: **DSS,RSD,SDLC**.

I. INTRODUCTION

In the current scenario, information systems are important part of any organization. As compared to 1970's and 1980's, they are becoming more and more complex. In the early years i.e., 1940's, software development was not an independent established discipline. Instead it was only an extension of the hardware. Earlier programs were written mostly in assembly language and were not complex. The persons/users who did programming were the one who also executed, tested and fixed the problems/errors in the software.

As the information systems became more and more complex and organizations became more dependent on software, a need was felt to develop the software in a systematic fashion. A survey was conducted by researchers in seventies and it was found that most of the software used by companies was of poor quality.. Some important characteristics of software are As compared to hardware which follows the "bathtub curve" as shown in Fig. 1.1, software does not wear out as over a period of time it will become more reliable.

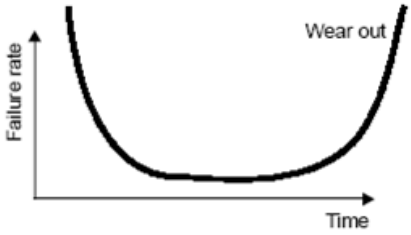

Figure 1.1 Bathtub curve (reproduce from pressman

Instead the software becomes obsolete due to new operating environment, new user requirements etc.. Hence it can only retire but not wear out. So it should follow the curve shown in Fig. 1.2.

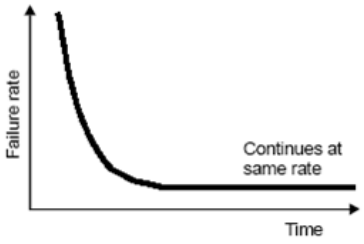

Figure 1.2 The software Curve

From 1960's to 1980's , software engineering was spurred by so called ***software crisis*.** A number of large size projects failed called ***software runaways*** because of development teams exceeding the budget, late delivery of software ,Poor quality ,user requirements not completely supported by the software, difficult maintenance, unreliable software. Statistics show that only 2% of the projects were used as they were delivered, 3% of the projects used after modifications, 47% of the software was never used only delivered, 19% of the software rejected or reworked and 29% was not even delivered. The problems increased because of increased dependence of business on software and lack of systematic approach to build the software. Developers and researchers realized that development of software was not an easy and straight forward task, instead it required lot of engineering principles.

II. SOFTWARE DEVELOPMENT PROCESS

Developers should be able to deliver good quality software to the end users using a well defined, well managed, consistent and cost-effective process. A software process framework therefore describes the different phases of the project via the activities performed in each phase without

telling about the sequence in which these phases or activities will be conducted. The different phases of software development process are shown in Fig. 1.3.

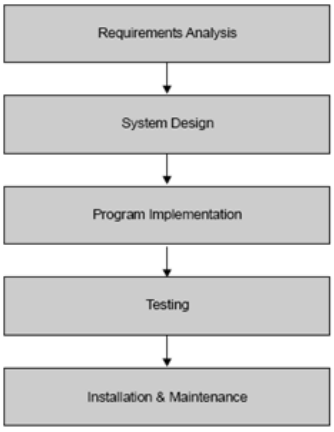

Figure. 1.3 Phases of Software Development Process

III. WHY ARE SOFTWARE LIFE CYCLE MODELS IMPORTANT?

The duration of time that begins with conceptualization of software being developed and ends after system is discarded after its usage, is denoted by Software Development Life Cycle (SDLC). A number of software life cycle models have been proposed by the researchers to organize the software engineering activities into phases. While adopting a software process for developing a product, the question which immediately comes to the mind is that whether the process is the right process to be adopted which will ensure a good quality product. It is normally seen that as complexity and size of the project increases, need for a formal process also increases.

3.1 SOFTWARE DEVELOPMENT LIFE CYCLE MODELS

Some of the widely used, well known software life cycle models are the Waterfall model, V model, Prototyping model, Incremental model, Spiral model etc. Depending upon the scope, complexity and magnitude of the project a particular software life cycle model is selected and this selection of life cycle model significantly contributes towards the successful completion of the project. In the following sections we will be briefly discussing each of these models.

3.1.1Build and Fix Model

Techniques used in the initial years of software development resulted into the term Build and Fix model. In fact the model resulted in a number of project failures because the product was not constructed using proper specification and design. Instead the product was reworked number of times in order to satisfy the clients as shown in Fig. 1.4. This model has only historical importance now.

The advantages and disadvantages of the model are:

Advantages

❖ The model is useful only for small size projects, in fact only for programming exercise 100 or 150 lines long.

Disadvantages

❖ The model is not suitable for large projects.

❖ As specifications are not defined, it results into product full of errors.

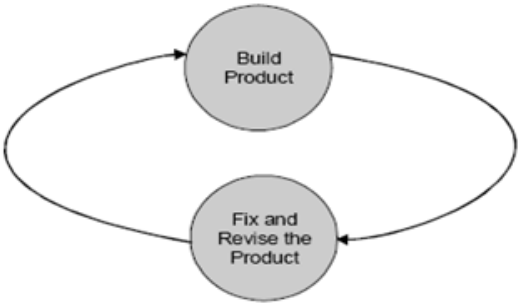

Figure. 1.4 Build and Fix Model

3.1.2The Waterfall Model

The Waterfall model is one of the most used model of 70's. It was proposed by Royce in 1970 as an alternative to Build and Fix software development method in which code was written and debugged. System was not formally designed and there was no way to check the quality criteria. Different phases of Waterfall model are shown in Figure 1.5.Given below is a brief description of different phases of Waterfall model.

❖ *Feasibility study* explores system requirements to determine project feasibility. All projects are feasible given unlimited resources and infinite time(Pressman92).

Feasibility can be categorized into

o *Economic feasibility*
o *Technical feasibility*
o *Operational feasibility*
o *Schedule feasibility*
o *Legal and contractual feasibility*
o *Political feasibility*

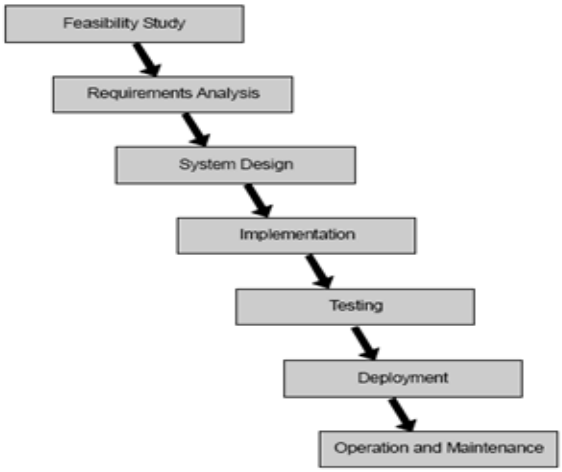

Figure 1.5 The Waterfall Model

Advantages and disadvantages of the Waterfall model are listed below:

Advantages

❖ Easy to understand even by non-technical persons i.e., customers.

❖ Each phase has well defined inputs and outputs e.g., input to system design stage is Requirement Specification Document(RSD) and output is the design document.

❖ Easy to use as software development proceeds.

❖ Each stage has well defined deliverables or milestones.

❖ Helps the project manager in proper planning of the project.

Disadvantages

❖ The biggest drawback of Waterfall model is that it does not support iteration.

Software development on the other hand is iterative i.e., while designing activities are being carried out, new requirements can come up. Similarly while product is being coded, new design and requirement problems can come up.

❖ Another disadvantage of Waterfall model is that it is sequential in nature. One cannot start with a stage till preceding stage is completed e.g., one cannot start with the system design till all the requirements are understood and represented.

❖ Users have little interaction with the project team. Their feedback is not taken during development.

❖ Customer gets opportunity to review the product very late in life cycle because the working version of product is available very late in software development life cycle.

❖ Model is very rigid because output of each phase is prerequisite for successive stage.

❖ The Waterfall model also has difficulty in accommodating changes in the product after the development process starts. Amount of documentation produced is very high.

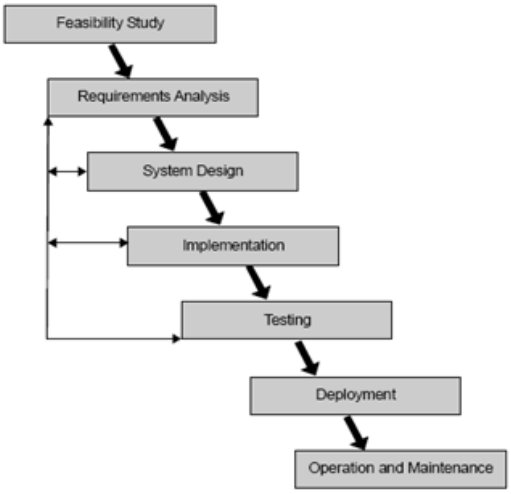

Figure 1.6 The Waterfall Model with feedback

Though Waterfall model has been used for large projects in the past, its use must be limited to projects in which requirements are well understood or the company is working on a product of similar kind which it has developed in the past. Modified version of Waterfall model shown in Fig. 1.6 allows feedback to preceding stages and hence is not very rigid. The Waterfall model is suited for well understood projects using familiar technology.It can also be used for existing projects if changes to be made are well defined.

3.1.3 *The V-Model*

This model was developed to relate the analysis and design activities with the testing activities and thus focuses on verification and validation activities of the product The advantages and disadvantages of the model are listed below.

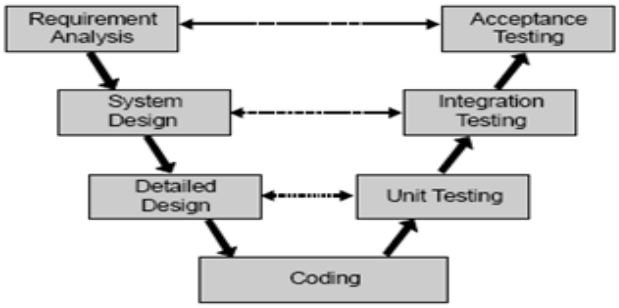

Figure 1.7 The V Model

Advantages

❖ The model is simple and easy to use.

❖ The V model focuses on testing of all intermediate products, not only the final software.
❖ The model plans for verification and validation activities early in the life cycle thereby enhancing the probability of building an error free and good quality product.

Disadvantages

❖ The model does not support iteration of phases and change in requirements throughout the life cycle.
❖ It does not take into account risk analysis.

The V model is used for systems in which reliability is very important e.g., systems developed to monitor the state of the patients, software used in radiation therapy machines.

3.1.4 The Prototype Model

The concept of prototyping is not new in various streams of engineering. A prototype is a partially developed product. Robert T. Futrell and Shafer in their book Quality Software Project Management define prototyping as a process of developing working replica of a system(Robert). This activity of prototyping now forms the basis of prototype software development life cycle model. Most of the users do not exactly know what they want until they actually see the product.

Two approaches of prototyping can be followed:

(i) Rapid Throwaway Prototyping: This approach is used for developing the systems or part of the systems where the development team does not have the understanding of the system. The quick and dirty prototypes are built, verified with the customers and thrown away. This process continues till a satisfactory prototype is built. At this stage now the full scale development of the product begins.

(ii) Evolutionary Prototyping: This approach is used when there is some understanding of the requirements. The prototypes thus built are not thrown away but evolved with time. The block diagram of the prototype model is shown in Fig. 1.8. The concept of prototyping has also led to the Rapid prototyping model and the Spiral model.

The advantages and disadvantages of the prototyping model are listed below:

Advantages

❖ A partial product is built in the initial stages. Therefore customers get a chance to see the product early in the life cycle and thus give necessary feedback.
❖ New requirements can be easily accommodated, as there is scope for refinement.
❖ Requirements become more clear resulting into an accurate product.

Disadvantages

❖ After seeing an early prototype end users demand the actual system to be delivered soon.

End users may not like to know the difference between a prototype and a well engineered fully developed system.

❖ Developers in a hurry to build prototypes may end up with sub-optimal solutions.
❖ Poor documentation.

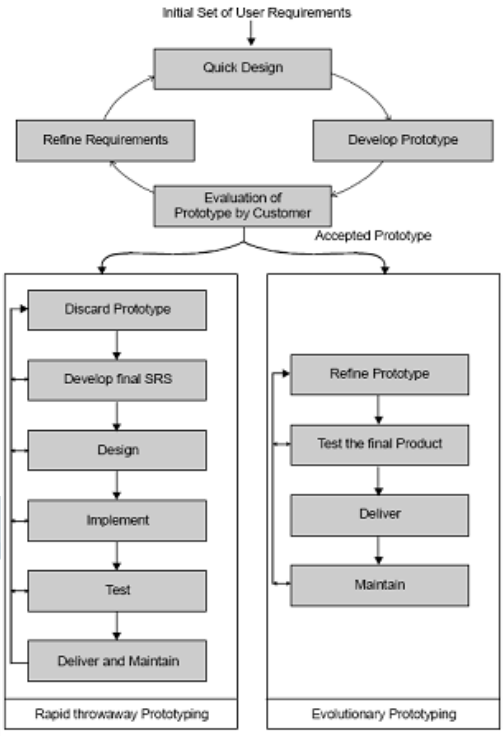

Figure 1.8 The Prototype Model

3.1.5 The Incremental Software Development Life Cycle Model

Software like all other complex systems is bound to evolve due to changing business requirements or new requirements coming up. Hence there is a need to have a model which can accommodate the changes in the product. The models discussed earlier do not take into consideration the evolutionary nature of the product. Evolutionary models are also iterative in nature. The incremental software development life cycle model is one of the popular *evolutionary software process model* used by industry.

The Fig. 1.9 shows the working of the incremental model.

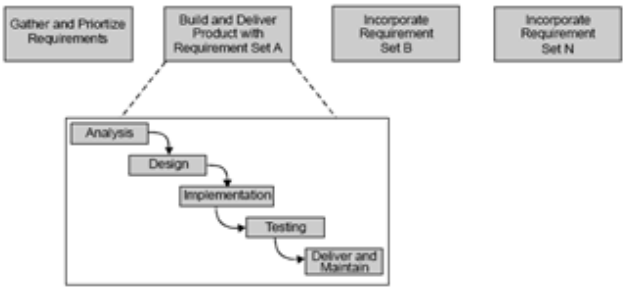

Figure 1.9 The Incremental Model

The advantages and disadvantages of incremental model are listed below:

Advantages

❖ As product is to be delivered in parts, total cost of project is distributed.

❖ Limited number of persons can be put on project because work is to be delivered in parts.

❖ As development activities for next release and use of early version of product is done simultaneously, if found errors can be corrected.

❖ Customers or end users get the chance to see the useful functionality early in the software development life cycle.

Disadvantages

❖ As product is delivered in parts, total development cost is higher.

❖ Well defined interfaces are required to connect modules developed with each phase.

3.1.6 The Spiral Model

The Spiral model is also one of the popular evolutionary process model used by the industry. The Spiral model was proposed by Boehm in 1988 and is a popular model used for large size projects. The model focuses on minimizing the risk through the use of prototype. One can view the Spiral model as a Waterfall model with each stage preceded by the risk analysis stage. A simplified view of Spiral model is shown in Fig. 1.10.

Figure 1.10 The Spiral Model

The radial coordinate in the diagram represents the total costs incurred till date. Each loop of the spiral represents one phase of the development. The model is divided into four quadrants, each with a specific purpose. Each spiral represents the progress made in the project

The advantages and disadvantages of spiral model are listed below:

Advantages

❖ The model tries to resolve all possible risks involved in the project starting with the highest risk.

❖ End users get a chance to see the product early in life cycle.

Disadvantages

❖ The model requires expertise in risk management and excellent management skills.

❖ The model is not suitable for small projects as cost of risk analysis may exceed the actual cost of the project.

❖ Different persons involved in the project may find it complex to use.

3.1.6 The Rapid Application Development (RAD) Model

The Rapid Application Development (RAD) model was proposed by IBM in 1980s and later on was introduced to software community by James Martin through his book Rapid Application development. The important feature of RAD model is increased involvement of the user/customer at all stages of life cycle through the use of powerful development tools. Block diagram of RAD model is shown in Fig. 1.11.

The RAD model consists of following four phases:

Requirements Planning – focuses on collecting requirements using elicitation techniques like brainstorming,

User Description – Requirements are detailed by taking users feedback by building prototype using development tools.

Construction – The prototype is refined to build the product and released to the customer.

Cutover – involves acceptance testing by the user and their training.

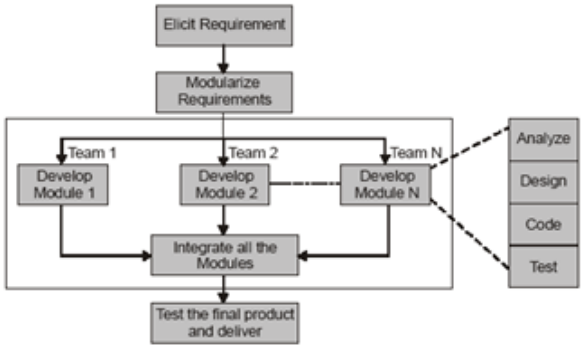

Figure 1.11 The RAD Model

The advantages and disadvantages of RAD model are discussed below:

Advantages

❖ As customer is involved at all stages of development, it leads to a product achieving customer satisfaction.

❖ Usage of powerful development tools results into reduced software development cycle time.

❖ Feed back from the customer/user is available at the initial stages.

Disadvantages

❖ The model makes use of efficient tools, to develop the prototype quickly, which calls for hiring skilled professional.

❖ Team leader must work closely with developers and customers/users to close the project in time.

Comparison of different process models in tabular form is shown in Table 1.1.

		Water fall	V Model	Incremental	Spiral	Prototype	RAD
1	Well Defined requirements	yes	yes	no	No	no	Yes
2	Domain knowledge of team members	adequate	adequate	adequate	Very Less	Very Less	Adequate
3	Expertise of users in problem domain	Very Less	Very Less	adequate	Very Less	adequate	Adequate
4	Availability	no	no	no	Yes	yes	Yes
5	Users involvement in all phases	no	no	no	No	yes	Yes
6	Complexity of System	simple	simple	Complex	Complex	Complex	Medium

IV. TYPES OF SOFTWARE

The software is being used in almost all the spheres of human life e.g., hospitals, banks, defense, finance, predicting stock rates, making pictures, running other software and so on. In fact the list is endless. Though it is somewhat difficult to categorize the software in different types but *based on their applications* the software can be categorized into following areas:

a. *System Software:* Systems software is necessary to manage the computer resources and support the execution of application programs. Software like operating systems, compilers, editors and drivers etc., come under this category. Operating systems are needed to link the machine-dependent needs of a program with the capabilities of the machine on which it runs. Compilers translate programs from high-level languages into machine languages. Without the presence of system software, a computer cannot function.

b. *Scientific Software:* Scientific and engineering software satisfies the needs of a scientific or engineering user to perform enterprise-specific tasks. Such software are written for specific applications using the principles, techniques and formulae specific to that field. Examples are software like MATLAB, AUTOCAD, PSPICE, ORCAD etc.

c. *Networking and Web Applications Software:* Networking software provides the required support necessary for computers to interact with each other, and with data storage facilities, in a situation where multiple computers are necessary to perform a task. The networking software is also used when software is running on a network of computers (such as the Internet or the World Wide Web). This category of software include all network management software, server software, security and encryption software and software to develop Web based applications like HTML, PHP, XML etc.

d. *Embedded Software:* This type of the software is embedded into the hardware normally in the Read Only memory(ROM) as a part of large system and is used to support certain functionality under the control conditions. Examples are software used in instrumentation and control applications, washing machines, satellites etc.

e. *Business Software:* This category of software is used to support the business applications and is the most widely used category of software. Examples are software for inventory management, accounts, banking, hospital, schools, stock markets etc., The software written for Enterprise Resource planning(ERP), project management , workflow management etc., also come under this category.

f. *Utilities Software:* The programs coming under this category perform specific tasks and are different from other software in terms of size cost and complexity. Examples are anti-virus software, voice recognition software, compression programs etc.

g. *Artificial Intelligence Software:* Software like expert systems, decision support systems, pattern recognition software, artificial neural networks etc., come in this category of software. Such type of software solve complex problems which are not affected by complex computations using non numerical algorithms.

4.1 Product and project characteristics as criteria for model selection

The literature on IT project SDLC models includes many factors characterizing an IT project that influence selection of an SDLC model. This paper will present the best known of these, taking contemporary knowledge as a reference point that aids the creation of a list of criteria for developing a computer-aided DSS for model selection. The criteria

were divided into two groups: product, and project criteria. The criteria concerning software, in other words: the product group, include: the type of information system, the size and complexity of the software, system architecture, modularity and level of module integrity, the variability and clarity of user requirements, or generally speaking – the quality of software

4.1.1 Type of information system

Information systems used by various institutions and companies can be divided into three categories.

a. The first category comprises commonly used systems that facilitate organization. Their main objective is to help employers in fulfilling various everyday tasks. Office software such as MS Word and MS Excel, or teamwork aiding programs such as Lotus Notes and Novell Group Wise are examples of such systems.
b. The second category of information systems comprises so called domain systems. This group can be further divided into two subgroups. The first of these consists of systems that are aimed at improving the efficiency of business processes in e.g. customer service or financial management. Examples of such systems are specialized Fixed Assets Management Systems (FAMS), or Customer Relationship Management (CRM). The second subgroup consists of systems created to aid creative activities, e.g. design or decision support processes. Elements characteristic of this subcategory are highly-advanced applications, such as systems meant for industrial designers and architects (Computer Aided Design, CAD), or systems aiding technological planning processes, designed for industrial engineers (Computer Aided Process Planning, CAPP).
c. The third category of systems is constituted by integrated management systems. This branch of systems is being steadily introduced into the field of applications that integrate the various levels on which companies function. The most advanced of such systems also cover strategic issues at the functional level of an organization. The best known examples are ERP3 integrated management systems, such as the SAP4 software family, Oracle Applications, IFS Applications5. For instance, models with a clearly distinguished maintenance stage seem to be the most appropriate for the first category, while models which emphasize the management of change and iterative cycles seem to suit the second category best. The third category seems to work best using models combining the characteristics of the first two types with the additional possibility of applying modular fragmentation in the design and implementation of software.

4.1.2 Software size and complexity

An important, though difficult activity to be done at the very beginning of an IT project is software sizing. This problem is usually solved by the use of various assessment methods based upon mathematical models, by brainstorming, or by the Delphi method. Depending on the accuracy of the outcome, the assessment data serve as an approximation of the size of the planned software6. Research shows that there is a straightforward relationship between software size and the amount of work needed for its development. Its exponential form points to the necessity of dividing large software projects into small parts, to reduce their complexity and ease project management in general.

4.1.3 Computer system modularity and the level of module integrity and complexity

The possibility of designing a modular framework for computer systems definitely simplifies work on an IT project. Problems related to the complexity of a computer system are frequently resolved by dividing the project into smaller parts, which are to be worked on by individual project teams. The integration process tends to be time consuming. However, the benefits of modularity are priceless, mainly because the complexity of a large project is dispersed over smaller tasks, each tackled individually.

CONCLUSION

Selecting an SDLC model can be compared in many ways to the specification of user requirements; the more data gathered and examined, the higher the chances for successful completion of the project. Just as the specifications of user requirements are vital in the stages of design and computer system development, so can the knowledge and regulations which constitute the basis for SDLC model selection determine the success or failure of a given project. To sum up, selecting an appropriate SDLC model is a complex and a challenging task, which requires not only broad theoretical knowledge, but also consultation with experienced expert managers. Therefore, the computer application presented should be perceived as the first step towards building a system that could be applied in practice; the possibilities for its development depend on the activity of its users. The flexible construction and permanent parameterization method used in the system makes it a multi-tasking tool for decision support, which not only gives decision-makers the opportunity to learn, but also allows them to participate in a system's development.

REFERENCES

[1] Bryant, A. (2000), "Chinese Encyclopaedias and Balinese Cockfights – Lessons for Business Process Change and Knowledge Management," In *Knowledge Engineering and Knowledge Management*.

[2] Lecture Notes in Artificial Intelligence #1937, R. Dieng and O. Corby, Eds., Springer Verlag.

[3] Bryant, A. (2001), "Metaphor, Myth and Mimicry: The Bases of Software Engineering," *Annals of Software Engineering 10*, 273-292.

[4] Curtis, B., H. Krasner, V.Y. Shen, and N. Iscoe (1987), "On Building Software Process Models under the Lamppost," In *Proceedings of the 9th International Conference on Software Engineering*, IEEE Computer Society Press, Monterey, CA., pp. 96-103.

[5] Evans M.W. and J.J. Marciniak (1987), *Software Quality Assurance and Management*, Wiley-Interscience, New York.

[6] Fayad, M.E. (1997), "Software Development Process: the Necessary Evil?" *Communications of the ACM*. *40*, 9,pp. 101-103.

[7] Gilb, T. (1988), *Principles of Software Engineering Management*, Addison-Wesley, Reading, MA.

[8] Haase, V., R. Messmarz, G. Koch, H.J. Kugler and P. Decrinis (1994), "BOOTSTRAP Fine-Tuning Process Assessment," *IEEE Software 11*, July, 25-35.

[9] Humphrey, W.S. (1988), "Characterizing the Software Process: A Maturity Framework," *IEEE Software* 5, 2, March, 73-79.

[10] Humphrey, W.S. (1995), *A Discipline for Software Engineering*, SEI Series in Software Engineering, Addison- Wesley, Reading, MA.

[11] Humphrey, W.S. and W.L. Sweet (1987), "A Method for Assessing the Software Engineering Capability of Contractors," *Technical Report CMU/SEI-87-TR-23*, Software Engineering Institute, Pittsburgh, PA.

AUTHORS PROFILE

Authors 1. Ashish B. Sasankar had done MCA,M.Phil(C/S) and M.Tech(CSE) .He had 12 yrs experience in teaching and Industry.

Authors 2. Prof. Ashwini Gharde is an assistant Professor in Rajiv Gandhi College of Engineering and Research, Nagpur. She had 9 yrs experience in teaching.