Factors Affecting the Choice of Software Life Cycle Models in the Software Industry-An Empirical Study

Article *in* Journal of Computer Science · January 2012

DOI: 10.3844/jcssp.2012.1253.1262

Rupa Mahanti
42 PUBLICATIONS **200** CITATIONS
SEE PROFILE

Madhumita Singha
Xavier Institute of Social Service
9 PUBLICATIONS **28** CITATIONS
SEE PROFILE

Vandana Bhattacharjee
Birla Institute of Technology, Mesra
102 PUBLICATIONS **511** CITATIONS
SEE PROFILE

Journal of Computer Science 8 (8): 1253-1262, 2012
ISSN 1549-3636
© 2012 Science Publications

Factors Affecting the Choice of Software Life Cycle Models in the Software Industry-An Empirical Study

[1]Rupa Mahanti, [2]M.S. Neogi and [3]Vandana Bhattacherjee
[1]Department of Computer Consultancy, Tata Consultancy Services, USA
[2]Department of I and M, Xavier Institute of Social Service Ranchi, India
[3]Department of CSE, Birla Institute of Technology, Ranchi, India

Abstract: Problem statement: The aim of this study was to present the results of the survey conducted with software professionals in a few Indian software companies. **Approach:** The study initially presents an overview of the common software life cycle models used in the software development. **Results and Conclusion:** The survey results revealed that the level of understanding of the user requirements is the most important fact in the choice of the life cycle model used in the software project. Project Complexity is the second most important factor. Man-machine Interaction is the least important factor in the choice of the life cycle model used in the software project. This study will be valuable for developers, analysts and project leaders in software organizations. This study was carried out with some boundaries like the number of companies, available resources, time constraints and so on.

Key words: Incremental model, cycle process, important factor, agile methodologies, software project, employee strength, software development, software life cycle models

INTRODUCTION

Large software systems, developed over several years, are the backbone of industries such as banking, retail, transportation, defense, healthcare and telecommunications. In other words, software has become an integral part of our life.

Developing software, which is within cost and time schedule, fulfils customer requirements and is reliable, seems to be the ultimate challenge for today's software professionals and calls for a systematic approach to software development. Once upon a time, software development consisted of a programmer writing code to solve a problem or automate a procedure. In those days, whenever a developer was tasked to perform programming or coding, he immediately would jump to it, start programming with or without full knowledge of what the system would look like, how the features were arranged. This was feasible because systems were very simple. However, nowadays, systems are so big and complex that teams of architects, analysts, programmers, testers and users must work together to create the millions of lines of custom-written code that drive our enterprises. In the absence of a proper software development plan, the developer is full of ideas, he/she wants to implement them all, but tends to forget about them because other features need to be prioritized. To manage this, a number of System Development Life Cycle (SDLC) models have been created. The advantage of adhering to a life cycle model is that it follows a systematic and discipline manner. It saves time, features of the system are well documented and above all, there is proper management and execution of plans. Without a life cycle model in place, the probability of chaos and project failure would have been very high (Russell, 2002; Ghezzi *et al*., 2002).

A lot of research has been reported on the evolution of software life cycle models. Agile methodologies are emerging and gaining popularity in industry (Manhart and Schneider, 2004; Cockburn and Highsmith, 2001; Coram and Bohner, 2005; Huo *et al*., 2004; Boehm, 2002; Highsmith, 2002, Kadary *et al*., 1989; Konito *et al*., 2004). Research has been reported regarding the suitability of different life cycle models and comparison of different software life cycle models (Davis *et al*., 1988). Some research has been reported on the relationship between project categories and life cycle models (Archibald and Vladmir, 2003; 2004; Archibald *et al*., 2003; Archibald, 2004b; Desaulniers *et al*., 2001). However, no empirical study has been reported regarding the importance of the factors affecting the choice of software life cycle models in the software industry. This study begins with an overview of the common software life cycle models used in the software development. This is followed by the research methodology. A survey was conducted in the Indian software organizations to study the factors affecting the

Corresponding Author: Rupa Mahanti, Computer Consultancy, Tata Consultancy Services, USA choice of software life cycle models in the Indian software industry. This is followed by the presentation of the results the survey.

However, there is no universal life cycle model, which is considered adequate in all situations in the development environment . Plan driven approaches like the waterfall model assume that requirements are static. Other iterative methods like spiral model and evolutionary model count on change. Agile methodologies consider software development as an empirical process and that people play the most important role in it. It has also been observed that agile practices do not compromise the quality of software products (Huo *et al*., 2004). A detailed description of traditional and agile methodologies can be found in literature (Brooks, 1995; McConnell, 1996; Szyperski, 1998; Pressman, 2004; Ghezzi *et al*., 2002; Jalote, 2005; Beck, 2000; Cockburn, 2001; Jeffries *et al*., 2000; Martin, 1991; Salo, 2004; Siponen *et al*., 2005; Scacchi, 2002; Neogi *et al*., 2007).

Overview of software life cycle models: The fundamental principle of software engineering is to design software products that minimize the intellectual distance between the problem and solution. Today methodical approaches to software design have evolved and design notations have proliferated. Many steps are involved in the successful development and deployment of computer software. Taken together, all these steps are referred to as the software life cycle.

From the IEEE Standard Glossary of Software Engineering Terminology, 1983, software life cycle is defined as follows.

‘The period of time that starts when a software product is conceived and ends when the product is no longer available for use. The software life-cycle typically includes a requirements phase, design phase, implementation phase, test phase, installation and checkout phase, operation and maintenance phase and sometimes, retirement phase’

The purposes of designing and documenting the overall project life cycle process for each project category are to:

- Enable all persons concerned with creating, planning and executing projects to understand the process to be followed during the life of the project
- Capture the best experience within the organization so that the life cycle process can be
  improved continually and duplicated on future projects
- Enable all the project roles and responsibilities and the project planning, estimating, scheduling, monitoring and control methods and tools, to be appropriately related to the overall project life cycle management process (Archibald, 2003; Archibald, 2004a)

Life cycle models for software development provide the basic guidelines for developing software using engineering technique. The first task of a software life cycle model is to determine the sequence of stages in software development and evolution and to establish the transition criteria for progression from one stage to the next. There are several life cycle models and many companies adopt their own models, but all have very similar patterns. Different life cycle models are the Code and Fix model, Classical Waterfall model, Iterative Waterfall model, Incremental model, Throwaway Prototyping Model, Evolutionary Prototyping model, Spiral model, Agile model, Extreme programming:

- Code and Fix Model: In the beginning of the software era, software process models included simply writing some code and trying to fix the problem. This is called code and fix model. It is a two-phase model. The first phase is to write the code and next phase is to fix it (Connell *et al*., 1993)
- Classical Waterfall Model: The Classical Waterfall Model was popularized in 1970 and is the backbone of many other software life cycle models. This process model is structured as a cascade of phases, where output of one phase acts as the input to the next phase. The classical waterfall model is an unrealistic one since there is no provision of detecting and rectifying the error at any stage of the life cycle. However in practical developments, there is always chance of errors, due to various reasons, in almost every phase of the life cycle. Therefore in any practical software development work, it is not possible to strictly follow the classical waterfall model (Royce, 1987)
- Iterative Waterfall Model: Iterative waterfall model suggests feedback paths in the classical waterfall model from every phase to its preceding phases. It allow for the correction of the errors committed during a phase that are detected in later phases. After detecting the error in later phases, it would be necessary not only to rework the design, but also to appropriately redo the coding and the system testing, thus incurring higher cost (Ghezzi *et al*., 2002)
- V-Shaped Model: Like the waterfall model, the V-Shaped model is sequential path of execution of processes i.e., linear in nature. Each phase must be completed before the next phase starts. However,

emphasis on testing in this model is more than that in the waterfall model (Raymond, 2005)

- Throwaway prototyping model: It was advocated by Brooks. It is useful in in situations where requirements and user's needs are unclear or poorly specified. The approach is to construct a quick and dirty partial implementation of the system during or before the requirements phase (Brooks, 1995; Gomma and Scott, 1981; Jalote, 2005)
- Evolutionary prototyping model: This is kind of mix of Waterfall model and prototyping. Presuppose gradual refinement of the prototype until a usable product emerges. Might be suitable in projects where the main problem is user interface requirements, but internal architecture is relatively well established and static (Jalote, 2005)
- Rapid Application Development (RAD): Rapid Application Development model was proposed in 1980 by IBM. This model is based on an evolving prototype that is not thrown away. . Rapid Application Development model is the first model, which emphasizes a short development cycle e.g., 60 to 90 days. It is a "high-speed" adaptation of the waterfall model, in which rapid development is achieved by using component based construction approach (Butler, 1994; Martin, 1991)
- Unified Process Model: During late 1980's and early 1990's, James Rumbaugh, Grady Booch and Ivar Jacobson developed the Unified Process, a framework, which is "use-case driven, architecture-centric, iterative and incremental" (Jacobson *et al*., 1999). The Unified Process Model consists of five phases
- Inception phase incorporates both customer communication and planning activities and emphasizes on refinement and development of use-cases as primary model
- Elaboration phase consist of customer communication and design activity
- Construction phase produces an implementation model that translates design classes produced during elaboration phase into software components that will be built to realize the system.
- Transition phase transfers the software from the developer to the end-user for beta testing and acceptance.
- Production phase in which on-going use of software is monitored and infrastructure support is provided (Jacobson *et al*., 1999)
- Incremental Model: It is decomposition of a large development effort into a succession of smaller components. The life cycle is also referred to as the successive versions or evolutionary model. Incremental model is an intuitive approach to the waterfall model. Multiple development cycles take place here, making the life cycle a "multi-waterfall" cycle. Cycles are divided up into smaller, more easily managed iterations. Each iteration passes through the requirements, design, implementation and testing phases. A working version of software is produced during the first iteration, so you have working software early on during the software life cycle. Subsequent iterations build on the initial software produced during the first iteration (Jalote, 2005; Pressman, 2004; McDermid, 1993)
- Spiral Model: Boehm proposed the Spiral model in 1988 (Boehm, 1988). It involves repetition of the same set of life-cycle phases such as plan, develop, build and evaluate until development is complete. The main emphasis is given on risk analysis. It encounters almost all the different types of risks such as cost overruns, change in requirements, loss of intelligent project personnel, unavailability of necessary hardware, competition from other software developers, technological drawbacks which obsolete the project and many more (Boehm *et al*., 1998; Boehm *et al*., 2000; Boehm and Hansen, 2001)
- Agile Software Development: In 2001, Kent Beck and 16 other noted software developers proposed an agile view of process. Agile software engineering combines a philosophy and a set of development guidelines. The philosophy encourages customer satisfaction and early incremental delivery of software, small, highly motivated project teams, informal methods, minimal software engineering work products and overall development simplicity. The development guidelines stress on delivery over analysis and design and active and continuous communication between developers and customers (Beck, 2000). The term 'agile' refers to a philosophy of software development. Extreme Programming, Scrum, Crystal, Adaptive Software Development (ASD) are agile methodologies (Boehm, 2002; Cockburn, 2001; Highsmith, 2002)

Research methodology: Research methodology can be viewed as the process taken to accomplish the key objectives of the research undertaken. The objectives of this research project were:

- To study the awareness/importance of software life cycle models in the Indian software Industry
- To identify the factors affecting the choice of the software life cycle models in the software/IT industry
- To determine the importance of factors affecting the choice of the software life cycle models

Authors have undertaken a survey-based approach to assess use software life cycle models in Indian Software Industry. In a survey based approach the usual proceeding to gather information is the usage of questionnaires or interviews. These are applied to a representative sample group and the outcomes are then analyzed. The aim is to derive conclusions that are descriptive, exploratory or explanatory. With the use of generalization the result from the sample is mapped to the whole group. It is, however, not possible to manipulate or control the samples. Nevertheless it is practicable to compare the result with similar outcomes of other surveys. Both qualitative as well as quantitative data can be derived from this strategy. Which one it is depends on the data that is being collected through the questionnaires or interviews and whether statistical analysis methods are applicable or not. Questionnaire survey methodology was preferred for this research since it is a reliable and economical method for data collection. In addition to the questionnaires, telephonic interviews were conducted to understand the relation between the factors affecting the choice of software life cycle models and the each individual software life cycle model.

Questionnaire survey methodology was preferred for this research since it is a reliable and economical method for data collection. An email survey was used to gather survey data. The advantages of the email survey approach to data collection are (Neuman, 2003; Sarantakos, 1998):

- Inexpensive
- Results are produced quickly
- Questionnaires are completed in the respondents' convenience
- Anonymity is greatly assured; and
- Respondents are at liberty to provide objective views on sensitive issues,

The questionnaire used in this study consisted of three parts:

- The software personnel information
- The background of the company
- The software process information

Table 1: Survey questionnaire

Notation	Factors
F1	Nature/type of project
F2	Project size
F3	Project duration
F4	Project complexity
F5	Level and type of expected risk
F6	Level of understanding of user requirements
F7	Level of understanding of the application area
F8	Customer involvement
F9	Experience of developers
F10	Team size
F11	Man-machine interaction
F12	Availability of tools and technology
F13	Versions of the product
F14	Level of reliability required

The first part dealt with the software personnel information such as experience and his/her designation. The second part was primarily aimed to understand some of the fundamental issues such as the size of the company and service areas. Third part of the questionnaire dealt with understanding the type of projects, processes and life cycle models. 14 factors affecting the choice of software life cycle models were derived mainly from the literature (Pressman, 2004; Ghezzi *et al*., 2002; Jalote, 2005; Martin, 1991; Archibald and Vladmir, 2004) and discussions with software quality professionals as shown in Table 1 below.

All factors were ranked on a five-point scale (1 = not very important, 2 = not important, 3 = important, 4 = very important and 5 = critical). The list of companies was obtained from National Association of Software and Services Companies (NASSCOM) database as well as using search engines (www.google.com). In this study, a total of 100 questionnaires were sent by email to software companies. The response rate from the companies was 51% (i.e., 51companies).

Results of the empirical investigation: The service areas of the companies participating in the survey comprised of Internet, software consultancy and services, data warehousing, IT enabled services, data mining, embedded technology, training and education, advanced databases, software vendor, telecommunication, ERP, mainframe technology, engineering design services and transportation sector services. 60% of the companies participating in the survey had multiple service areas. The rest 40% had only one service area. 50% of the companies had software consultancy and services as one of their service area. As shown in Fig. 1, 52% of the participants of the Software Development Life Cycle model survey were big companies with more than 1000 employees; 18% of the respondent hailed from companies with employee strength between 501 and 1000; 12% of the respondents were companies with 301-500 employees. 12% of the respondents were companies with 301-500 employees. The remaining 6% of the companies had employee strength of less than 100. Figure 2 shows the total work experience of the participants of survey. Figure 3 shows the designations of the individuals participating in the survey. All the respondents had worked in more than multiple projects in different technologies and business domains in the software industry. 50% of the respondents had worked on Commercial software projects; 30% had worked on open source software projects and web applications respectively; 25% had worked on ERP projects; 15% had worked on mission critical software projects; 10% of the respondents had worked on embedded software projects as shown in Table 2.

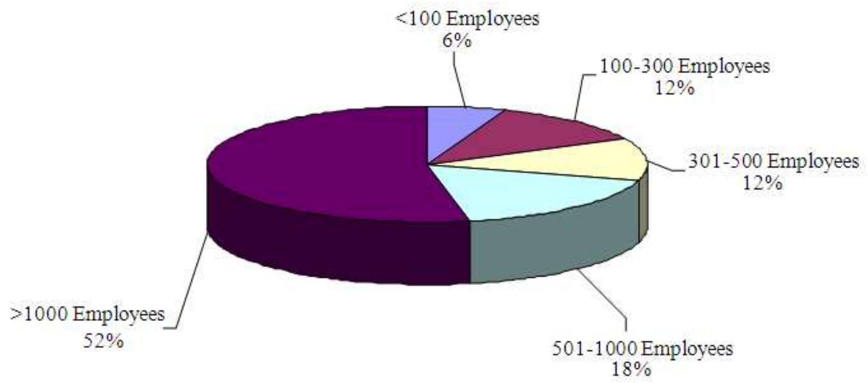

Fig. 1: Distribution of the employee strength of the companies

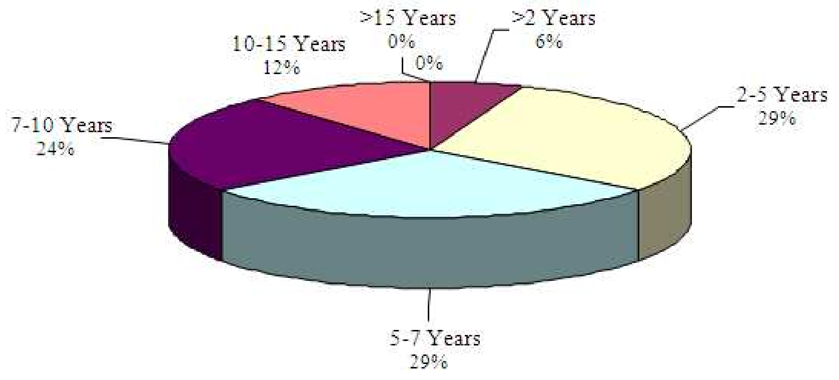

Fig. 2: Percentage distribution of experience of employees participating in the study

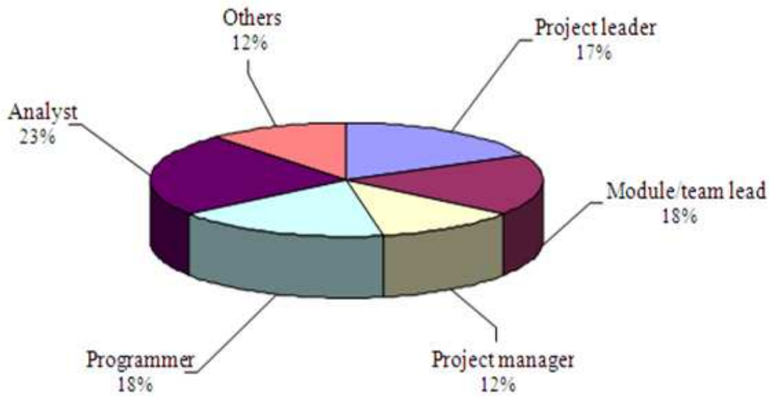

Fig. 3: Percentage distribution of employees participating in the study

Table 2: Different types of projects tackled by the respondents

Different types of projects	Percentage of respondents
Commercial software projects	50
Open source software projects	30
Web application projects	30
ERP projects	25
Mission critical software projects	15
Embedded software projects	10

Figure 4 shows the usage of the software life models. As shown in Fig. 4: Agile methodologies have a high popularity with 40% of the projects using Agile methodologies. Classical Waterfall Model and Code and Fix Model were each used by 2% of projects. Classical Waterfall Model and Code and Fix Model were each used by 2% of projects. The survey revealed that software professionals were most comfortable working with iterative life cycle model because it was easy to follow. In terms of rigidity of entry-exit criteria of phases in life cycle models the Classical Waterfall Model was most rigid with a rating of 5 the scale being 1-5 with 1 being least rigid and 5 being most rigid. In all the other models, the rigidity of entry-exit criteria of phases in life cycle models was less rigid with a rating of 3.

The participants were also asked to prioritize the 14 key attributes which are important in the software development process. These attributes were derived from the literature (Pressman, 2004; Ghezzi *et al*., 2002; Jalote, 2005) and through interactions with professionals in the software industry. The participants were asked to assign a rank in the range of 1-14 with 1 being the most important and 13 being the least). The average scores are as follows:

- Functionality-1.4
- Correctness, reliability-1.6
- Consistency-1.8
- Cost, timeliness-2
- Efficiency-2.4
- Integrity-2.5
- Maintainability-3
- Usability-3.2
- Complexity, reusability-4
- Portability-6

The respondents were asked to rate each factor on a Likert scale of 1-5 (1 = least important and 5 = crucial). The scores were added together and then divided by the number of observations per factor to determine the mean score of each factor. The results of the analysis showing the mean scores and standard deviation of each essential factor affecting the choice of software life cycle models is shown in Table 3. Fig. 5: shows the mean score of each factor, the higher the score, the greater the importance of the factor. F6 has the highest score. F1, F4, F6 and F8 have mean scores of more than 4. Factor F5 has a mean score of 4. It was observed from the analysis that the following factors were critical to choice of software life cycle models within software industry:

Factor 1 - Nature of Project
Factor 4 - Project Complexity
Factor 5 - Level and type of expected risk
Factor 6 - Level of understanding of user requirements
Factor 8 - Customer Involvement

5. Results of the telephonic interviews: Telephonic interviews were conducted with project managers with approximately 10 years of software development and project management experience to understand the relation between the factors affecting the choice of software life cycle models and the each individual software life cycle model. Classical Waterfall Model is suitable for complicated mainframe projects where requirements are clear and stable, the team members are moderately experienced and have a fairly good understanding of the application area, customer involvement is low, project risk is low and high software reliability is required. Iterative Waterfall Model is suitable for large projects where the requirements are not very clear and customer is involved during the development. RAD is apt for small or medium sized less complex projects where the project duration was short, team is small, high level of reliability is not required, active customer involvement is required and developers are highly skilled with good knowledge of CASE tools, DBMS, GUI tools, Object Oriented Techniques as well as the application area. The V Process Model works well with medium sized, moderately complex projects with moderate risk involved and the customer is available for giving feedback. The Unified Process model works for small, medium as well as large projects with varied degrees of technical complexity, where changing requirements are involved and considerable risk is involved. The Unified Process Model works well with all team sizes. Agile methodologies are apposite for small to medium sized projects where requirements are changing and unstable, high reliability is not required, team is small, developers are experienced and customer involvement and interaction is high. Agile is good at dealing with project risks. Evolutionary prototyping model is appropriate for projects where the project deadlines are not rigid, requirements are not well understood, level of reliability required is moderate, project risk is high, the developers are skilled and experienced, customer involvement is necessary and the system is implemented via number of versions. Throwaway Prototyping Model is suitable for projects where the project deadlines are not rigid, requirements are not well understood, level of reliability required is moderate, project risk is high, the developers are skilled and experienced and customer involvement is necessary. Code and fix model works with very small simple projects where the requirements are well understood, risks are minimum or absent and low level of reliability is required. Incremental models are suitable for large event driven projects, where requirements are not clearly understood, project risk is high. Spiral Model is apt for large complex real time application projects (mostly in house projects) requiring high reliability, where the requirements are unstable and not well understood and project risk is high, where their no restriction on the team size and the project manager is highly experienced and project deadlines are not rigid.

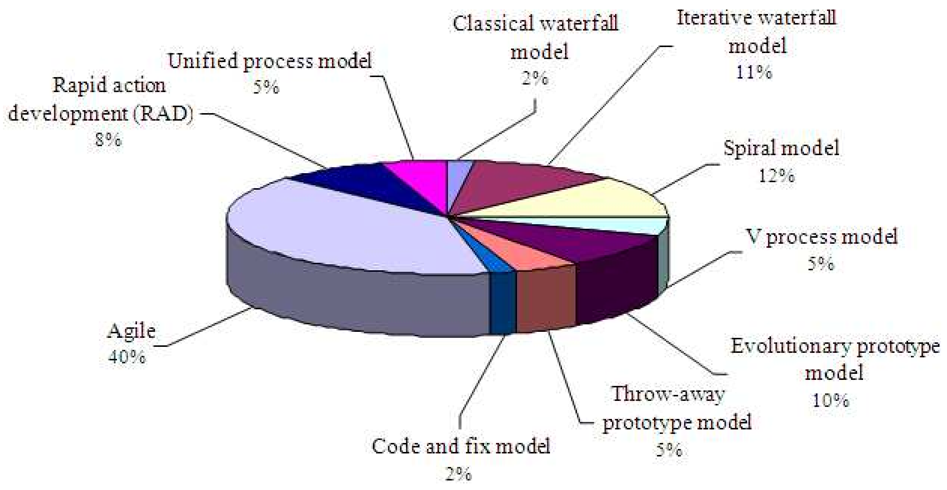

Fig. 4: Usage of software life cycle models

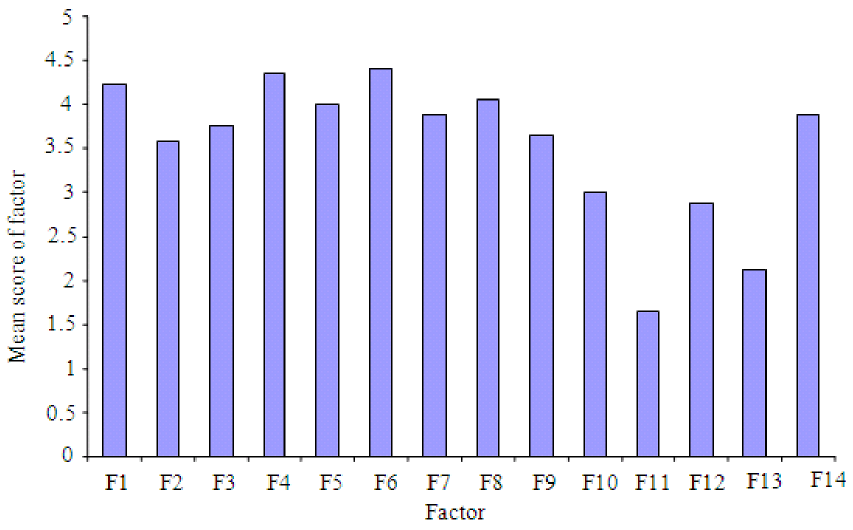

Fig. 5: Scores of each factor from survey results

Table 3: Factor affecting the choice of software life cycle models

Factor notation	Factor name	Mean	Standard deviation
F1	Nature of project	4.23529	0.70189
F2	Project size	3.58824	0.87026
F3	Project duration	3.76471	0.83137
F4	Project complexity	4.35294	0.60634
F5	Level and type of expected risk	4.00000	0.93541
F6	Level of understanding of user requirements	4.41177	0.71229
F7	Level of understanding of the application area	3.88235	0.99262
F8	Customer Involvement	4.05882	0.74755
F9	Experience of developers	3.64706	0.99632
F10	Team size	3.00000	0.79057
F11	Man-machine interaction	1.64706	0.78591
F12	Availability of tools and technology	2.88235	0.99262
F13	Versions of the product	2.11765	0.85749
F14	Level of reliability required	3.88235	0.70189

CONCLUSION

This article presents the results of a survey and telephonic interviews carried out in the Indian software industry regarding software life cycle models. The study presents the factors which are critical to choice of software life cycle models within software industry and the relation between the factors and the software life cycle models. A total of 14 factors were considered in the study. The survey revealed that Agile methodologies are the most popular models in the Indian software industry. This study was carried out with some boundaries such as the number of companies, available resources, time constraints and so on.

Different positions of the respondents may have different opinions. The limited sample size of the current study, calls for a survey on a larger scale for greater validity of the findings from this research. Because of limited budget and time constraints, an email survey was carried out for the companies. According to Gillham (2000), the scaled questions have disadvantages because respondents often do not use the whole scale, whatever response they tick, we do not know why a particular response was chosen. Semi-structured interviews with people at different levels of software development expertise are currently being conducted to obtain a deeper understanding of the obstacle and challenges in software development life cycles in the software industry. This research is a part of ongoing doctoral research project on software development life cycle models (Neogi *et al*., 2007). A part of this research involves the evolution of a new software development life cycle model to ensure better quality software (Bhattacherjee *et al*., 2008, Bhattacherjee *et al*., 2009a; 2009b).

REFERENCES

Archibald, D.R. and I.V. Voropaev, 2003. Commonalities and differences in project management around the world: A survey of project categories and life cycle models.

Archibald, D.R. and I.V. Voropaev, 2004. Project categories and life cycle models: Report on the 2003 IPMA Global Survey. Proceedings of the 18th IPMA World Congress on Project Management, Jun. 18-21, Budapest, Hungary, pp: 1-6.

Archibald, D.R., 2003. Life cycle models for high-technology projects-applying systems thinking to project.

Archibald, D.R., 2004a. Life cycle models for high-technology projects. Proceeding of the 4th International Project Management Seminar-PMI-SP Chapter, Dec. 9-10, Brazil, pp: 1-10.

Archibald, D.R., 2004b. State of the art of project management: 2004. Proceedings of the International Seminar on Project Management, Dec. 9-10, Sao Paulo, Brazil.

Beck, K., 2000. Extreme Programming Explained: Embrace Change. 2nd Edn., Addison-Wesley Professional, Reading, ISBN-10: 0201616416, pp: 190.

Bhattacherjee, V., M.S. Neogi and R. Mahanti, 2009a. Are our students prepared for testing based software development. Proceedings of the 22nd Conference on Software Engineering Education and Training, Feb. 17-20, IEEE Xplore Press, Hyderabad andhra Pradesh, pp: 210-211. DOI: 10.1109/CSEET.2009.57

Bhattacherjee, V., M.S. Neogi and R. Mahanti, 2009b. A Process model for software development amongst students. Int. J. Rece. Trend. Eng., 1: 69-74.

Bhattacherjee, V., N.S. Madhumita and M. Rupa, 2008. Software development approach of students: An evaluation. Proceedings of National Conference on Methods and Models in Computing, (MMC' 08), New Delhi, pp: 21-29.

Boehm, B. and W.J. Hansen, 2001. The spiral model as a tool for evolutionary acquisition. Software Engineering Institute.

Boehm, B., 2000. Spiral development: Experience, principles and refinements. Carnegie Mellon University.

Boehm, B., 2002. Get Ready for Agile Methods, with care. IEEE Comput., 35: 64-69. DOI: 10.1109/2.976920

Boehm, B., A. Egyed, J. Kwan, D. Port and A. Shah *et al*., 1998. Using the WinWin Spiral Model: A Case Study. IEEE Comput., 31: 33-44. DOI: 10.1109/2.689675

Boehm, B.W., 1988. A spiral model of software development and enhancement. IEEE Comput., 21: 61-72. DOI: 10.1109/2.59

Brooks, Jr., F.P., 1995. The Mythical Man-Month. 1st Edn., Addison-Wesley Professional, ISBN-10: 0201835959, pp: 336.

Butler, J., 1994. Rapid application development in action, managing system development. Applied Comput. Res., 14: 6-8.

Cockburn, A. and J. Highsmith, 2001. Agile software development, the people factor. IEEE Comput., 34: 131-133. DOI: 10.1109/2.963450

Cockburn, A., 2001. Agile Software Development. 1st Edn., Addison-Wesley Professional, ISBN-10: 0201699699, pp: 304.

Connell, M.C., J.J. Carta and D.M. Baer, 1993. Programming generalization of in-class transition skills: Teaching preschoolers with developmental delays to self-assess and recruit contingent teacher praise. J. Applied Behav. Anal., 26: 345-352. DOI: 10.1901/jaba.1993.26-345

Coram, M. and S. Bohner, 2005. The impact of agile methods on software project management. Proceedings of the 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems, Apr. 4-7, IEEE Xplore Press, pp: 363-370. DOI: 10.1109/ECBS.2005.68

Davis, A.M., E.H. Bersoff and E.R. Comer, 1988. A strategy for comparing alternative software development life cycle models. IEEE Trans. Softw. Eng., 14: 1453-1461. DOI: 10.1109/32.6190

Desaulniers, H., A. Douglas and J. Robert, 2001. Matching software development life cycles to the project environment. Proceedings of the Project Management Institute Annual Seminars and Symposium, Nashville, TN. Newtown Square, PA: Project Management Institute.

Ghezzi, C., M. Jazayeri and D. Mandrioli, 2002. Fundamentals of Software Engineering. 2nd Edn., Prentice-Hall, India, ISBN-10: 0133056996, pp: 604.

Gillham, B., 2000. Developing a Questionnaire. 1st Edn., Continuum International Publishing Group, London, ISBN-10: 0826447953, pp: 93.

Gomma, H. and D.B.H. Scott, 1981. Prototyping as a tool in the specification of user requirements. Proceedings of the 5th International Conference on Software Engineering, (SE' 81), IEEE Press Piscataway, NJ, USA., pp: 333-341.

Highsmith, J., 2002. Agile Software Development Ecosystems. 1st Edn., Addison-Wesley, Boston, ISBN-10: 0201760436, pp: 448.

Huo, M., J. Verner, L. Zhu and M.A. Babar, 2004. Software quality and agile methods, Proceedings of the 28th Annual International Computer Software and Applications Conference, Sept. 28-30, IEEE Xplore Press, pp: 520-525. DOI: 10.1109/CMPSAC.2004.1342889

Jacobson, I., G. Booch and J. Rumbaugh, 1999. The Unified Software Development Process. 1st Edn., Addison-Wesley Professional, ISBN-10: 0201571692, pp: 512.

Jalote, P., 2005. An Integrated Approach to Software Engineering. 3rd Edn., Springer, ISBN-10: 038720881X, pp: 580.

Jeffries, R., A. Anderson and C. Hendrickson, 2000. Extreme Programming Installed. 1st Edn., Addison-Wesley Professional, ISBN-10: 0201708426, pp: 288.

Kadary, V., D. Even-Tsur, N. Halperin and S. Koenig, 1989. Software life cycle models-industrial implications. Proceedings of the 4rth Israel Conference on Computer Systems and Software Engineering, Jun. 5-6, IEEE Xplore Press, Herzlia, pp: 98-103. DOI: 10.1109/ICCSSE.1989.72722

Konito, J., M. Hoglund, J. Ryden and P. Abrahamson, 2004. Managing commitments and risks: challenges in distributed agile development. Proceeding of the 26th International Conference on Software Engineering, May, 23-28, IEEE Xplore Press, pp: 732-733. DOI: 10.1109/ICSE.2004.1317510

Manhart, P. and K. Schneider, 2004. Breaking the Ice for agile development of embedded software: An industry experience report. Proceedings of the 26th International Conference on Software Engineering, May 23-28, IEEE Xplore Press, pp: 378-386. DOI: 10.1109/ICSE.2004.1317460

Martin, J., 1991. Rapid Application Development. 3rd Edn., Macmillan Coll Div, ISBN-10: 0023767758, pp: 736.

McConnell, S., 1996. Rapid Development: Taming Wild Software Schedules. 1st Edn., Microsoft Press, Redmond, Washington, ISBN-10: 1556159005, pp: 680.

McDermid, J., 1993. Software Engineer's Reference Book. 1st Edn., CRC Press, London, ISBN-10: 0849377668, pp: 1015.

Neogi, M.S., R. Mahanti and V. Bhattacherjee, 2007. Evolution of software process models. Proceedings of the National Conference on Technological Advances and Emerging Societal Implications, (TAESI' 07), NIT Rourkela, pp: 402-415.

Neuman, W.L., 2002. Social Research Methods: Qualitative and Quantitative Approaches. 5th Edn. Allyn and Bacon, ISBN-10: 0205353118, pp: 592.

Pressman, R., 2004. Software Engineering: A Practitioner's Approach. 6th Edn., McGraw Hill, ISBN-10: 007301933X, pp: 880.

Raymond, L., 2005. Software development life cycle models.

Royce, W.W., 1987. Managing the development of large software systems: Concepts and techniques. Proceedings of the 9th International Conference on Software Engineering, (SE' 87), IEEE Computer Society Press Los Alamitos, CA, USA., pp: 328-338.

Russell, K., 2002. QuickStudy: System development life cycle. Computer World.

Salo, O., 2004. Improving software process in agile software development projects: Results from two XP case studies. Proceedings of the 30th EUROMICRO Conference, Aug. 31-Sep. 03, pp: 310-317.

Sarantakos, S., 1998. Social Research. 2nd Edn., Macmillan, London.

Scacchi, W., 2002. Process models in software engineering. Encyclopedia Software Eng. DOI: 10.1002/0471028959.sof250

Siponen, M., R. Baskerville and T. Kuivalainen, 2005. Integrating security into agile development methods. Proceedings of the 38th Annual Hawaii International Conference on System Sciences, Jan. 03-06, IEEE Xplore Press, pp: 185a-185a. DOI: 10.1109/HICSS.2005.329

Szyperski, C., 1998. Component Software: Beyond Object-Oriented Programming. 1st Edn., Addison-Wesley, ISBN-10: 0201178885, pp: 411.