The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.67 e-ISSN: 2251-7545

A Review on Software Engineering Methods for Distributed Systems

Hamid Reza Ranjbar
Department of Computer Engineering, Islamic Azad University, Science and Research Branch,
Booshehr, Iran
Email: H.R_Ranjbar@yahoo.com

Mehdi Alimi Motlaghfard
Department of Computer Engineering, Islamic Azad University, Science and Research Branch,
Booshehr, Iran
Email: Alimi@LianPro.com

***Abstract*— Today we collect data from different size of data, different locations and different type with a large scale in each site. Current computer server systems cannot process and collect these big data. For this issue, distributed computing system proposed in the literature. Supercomputers are changed to distribute computing, such as cloud computing systems. Software engineering is a part of each software system. The software engineering describes the architecture, connection. The software architecture should responsible for whole the life cycle of a system include analyzing, designing, implementing and maintaining a software system. In this paper we will review different methods of software engineering for distributed systems.**

Keywords—Distributed System; Software Engineering; Software Methods;

I. INTRODUCTION

The study of software engineering has always been complex and difficult. The complexity make technical problems [1, 2]. Some aspects of these problems come from complex systems.

Several distributed system [10] studies such as [8 ,9] have been conducted in software engineering for several years, but have only relatively recently achieved significant recognition in the broader software engineering research community. However this subarea has also reached a discernibly new level of maturity that is evidenced by the new types of questions and methods seen in more recent studies. In particular, software engineering are beginning to address the big data in software development. One indication of this broadening of focus is the nature of recent work in traditionally software engineering research [1]. Some systems and methods are developed to concur to this complexity such as Cloud Template [6] that shows how an integrated system could improve flexibility in the cloud computing systems.

Some important reasons for using software engineering are listed below [4]:

- This method is useful for large, high quality software systems such as distributed systems.
- Software engineering techniques are needed because large systems cannot be completely understood by one person.
- Teamwork and co-ordination are required.
- Key challenge: Dividing up the work and ensuring that the parts of the system work properly together.
- The end-product that is produced must be of sufficient quality

II. WATERFALL DEVELOPMENT

The waterfall model is essentially a slight variation of the model. The waterfall model which shows in Figure 1is generally attributed to Royce in 1970 [3]. However, a clearly phased approach to the development of software including iteration and feedback, could already be found in publications from the early 1960s and particular for distributed systems could be found in the early 2000.

The waterfall model [5] development particularly emphasizes the interaction between each subcategory phases. Testing software is not an activity which strictly follows the implementation phase. In each phase of the software development process, we have to compare the results obtained against those that are required. In all phases, quality has to be assessed and controlled.

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.67 e-ISSN: 2251-7545

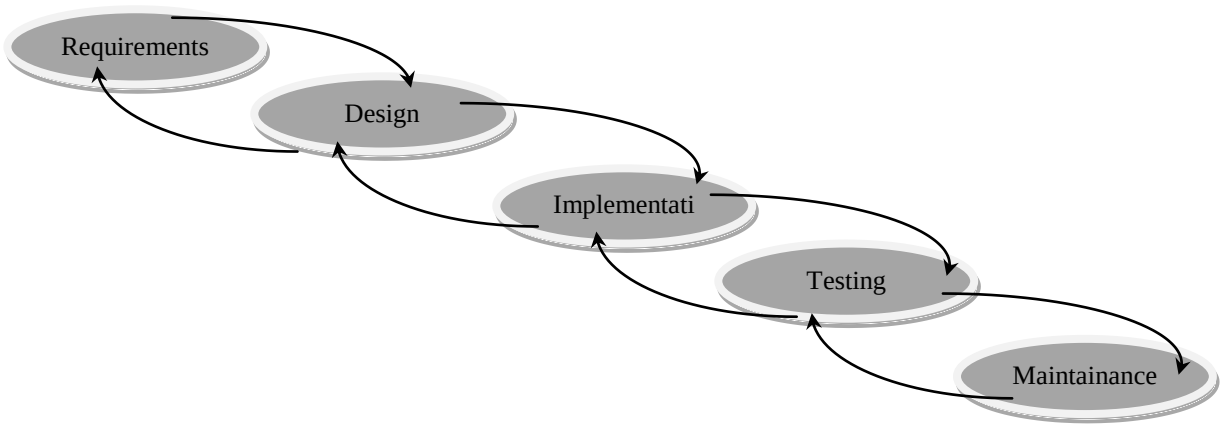

Figure 1. Waterfall Software Development

In figure 1, in each steps we have a verification and a validation that presents for asks if the distributed system meets its requirements such as concurrency and thus tries to assess the correctness of the transition to the next phase. Validation asks if the system meets the user's requirements such as ACID transactions in distributed systems [12].

III. Rapid Application Development

Rapid Application Development (RAD) [4] is another method of development process models for a software application. This development emphasizes user involvement, reuse, prototyping, the use of automated tools, and small development teams. In most development models such as RAD, a set of requirements is fixed and then the project attempts to fulfill these requirements within some estimated period of time. Using this method is difficult for distributed systems. By using this method, the time frame is decided upon first and then the project tries to realize the requested functionality within that time frame. If it turns out that not all of the functionality can be realized within the time frame, some of the functionality is sacrificed. The agreed deadline however is immovable.

An increasing number of computer systems are being viewed in terms of autonomous agents. Agents are being espoused as a new theoretical model of computation that more closely reflects current computing reality than Turing Machines. Agents are being advocated as a next generation model for engineering complex, distributed systems. Agents are also being used as an overarching framework for bringing together the component AI subdisciplines that are necessary to design and build intelligent entities. Yet despite this intense interest, a number of fundamental questions about the nature and the use of the agent-oriented approach remain unanswered.

IV. Agile Software Development

Agile software development is another important method in software engineering. The goals of this method are listed below [4, 5, 7, 12]:

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face–to–face conversation.

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.67 e-ISSN: 2251-7545

- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity – the art of maximizing the amount of work not done – is essential.
- The best architectures, requirements, and designs emerge from self–organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

V. SPIRAL" MODEL

The "Spiral" Model [11, 14] shows in figure 2 addresses many of problems by explicitly including multiple paradigms in the process model. The spiral model describes software development as an iteration over four phases of activity which combine several other approaches, including specification- driven and prototype-driven [13] development. The particular combination of these activities which is to be used at a given point is determined by the need to identify and resolve risks in the process. The spiral model, therefore, represents a significant advance over previous models due to its incorporation of risk analysis and multiple paradigms. It does, however, have some shortcomings. In particular, while the spiral model is well-suited to describing internal software development projects for an organization, it appears to be less suitable for describing software development under external Contract [14].

VI. CONCLUSION

In this paper we review some important development method of software engineering that could help us to improve develop software application for distributed systems. As we mentioned in this paper, the complex method of development of software engineering will be more useful than simple method such as Agile or RAD methods.

REFERENCES

[1] Booch, Grady, Douglas L. Bryan, and Charles G. Petersen, "Software engineering with Ada", Addison-Wesley Professional, 1994.

[2] Roger Pressman, "Software Engineering: A Practitioner's Approach", 7/e McGraw-Hill, Slides copyright 2009

[3] Van Vliet, Hans, Hans Van Vliet, and J. C. Van Vliet. Software engineering: principles and practice. Vol. 2. Wiley, 1993.

[4] Beck, Kent, et al. "Manifesto for agile software development." (2001).

[5] Van Vliet, Hans, Hans Van Vliet, and J. C. Van Vliet. Software engineering: principles and practice. Vol. 2. Wiley, 1993.

[6] Mehdi Bahrami, "Cloud Template, a Big Data Solution", International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 2, pp. 13-17, 2013, Doi: 10.7321/jscse.v3.n2.2

[7] Martin, Robert Cecil. Agile software development: principles, patterns, and practices. Prentice Hall PTR, 2003.

[8] Tanenbaum, Andrew S., and Maarten Van Steen. Distributed systems. Vol. 2. Prentice Hall, 2002.

[9] Chandy, K. Mani, and Leslie Lamport. "Distributed snapshots: determining global states of distributed systems." ACM Transactions on Computer Systems (TOCS) 3.1 (1985): 63-75.

[10] Magee, Jeff, Naranker Dulay, and Jeff Kramer. "Regis: A constructive development environment for distributed programs." Distributed Systems Engineering 1.5 (1994): 304.

[11] Sommerville, Ian. "Software process models." ACM Computing Surveys (CSUR) 28.1 (1996): 269-271.

[12] Tamer èOzsu, M., and Patrick Valduriez. Principles of distributed database systems. Springer, 2011.

[13] B.W. Boehm, "A spiral model of software development and enhancement", IEEE Computer 21:5, 1988

[14] B. W. Boehm, "A Spiral Model of Software Development and Enhancement," Proceedings of an International Workshop on the Software Process and Software Environments, Coto do Caza, California, March 1985, published as Software Engineering Notes, vol. 11, no. 4, 1986, pp. 22-42.

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.67 e-ISSN: 2251-7545

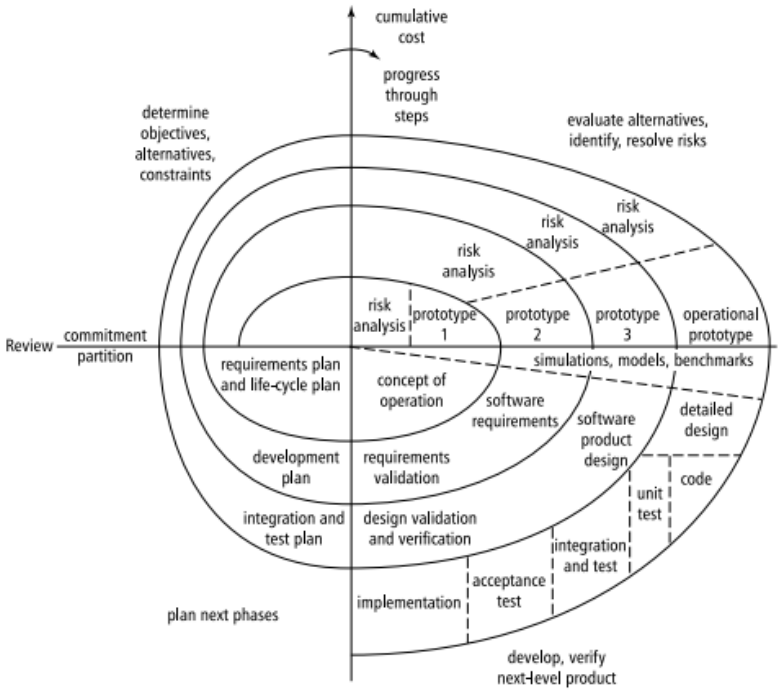

Figure 2. Spiral Method Development [14]