LGURJCSIT
Volume No. 1, Issue No. 4 (October- December), pp. 75-81

A Non-Parametric Comparison between Advances Software Engineering Process Model

Sadia Kousar, Sundus Munir, Afrozah Nadeem, Shafia Kousar

.

Abstract: Software development process provides detailed guideline for development testing and maintenance of software products. It deals with the risks associated with software development and a road map to manage its complexities. In other words, software development processes are considered as optimized solution specific to any particular software product development. There are many software process models available in literature. This research performs a non-parametric comparison between formal process model, agile process model and agent based process model to aid software community in developing quality software product.

Keywords: *- Software engineering aspect oriented software engineering, X-Machines, Agile Methodology, formal methods,*

1. INTRODUCTION

A software system is an interrelated and interacting units designed to work as single entity [1]. To model software system, software engineers uses a common framework known as System Development Life Cycle. A typical SDLC follows Planning, Requirement Analysis, Design, Code, Testing, Implementation, Maintenance phases. In 2012,the National Institute of Standards and Technology (NIST) reported that software defects cost the U.S. economy 59.9 billion Dollars annually. NIST estimated that 22 billion dollars could be saved by using efficient process modeling and testing techniques. Facts and figures of proposed NIST Report tell that software error identification and correction cost is 80% of the overall cost of software development [2]. So, there is a compelling need of efficient software development to help software developers in producing quality software products and to save revenue that is spent on error identification and correction at later stages. In order to entertain this issue a non-parametric comparison between formal process model, agile process model and agent based process model is performed.

Proposed paper is organized as follows. Section 2 expresses brief detail of formal software process model and its comparison with mentioned models. Section 3 discusses detail of agile software process model and its comparison with earlier mentioned models. Detail of agent based process model and their comparison is given in section 4. Conclusion and references are given in section 5.

2. FORMAL PROCESS MODEL

A formal process model is a model that describes the structure and methodology of a software process in mathematics based on set theory and Boolean algebra [4]. These models are considered as a foundation for describing complex or safety critical systems [5]. X-machine is formal process model that is used to write software specification as well as testing and also for verifying their implementation is behaviorally equivalent to its specification. Another application of formal process model is cleans room software engineering. It is an approach that emphasizes the need to build correctness into software as it is under development [16, 17, 18]. Variety of languages and tools (Z, VDM, CSP, Petri Nets, Abstract State Machine, etc) is also available for writing specification of software [19, 20, 21, 22].

2.1. Critical distinctions between formal and other models:

- The software requirements and specification phases are refined into a detailed formal specification, which is expressed mathematically.
- The design, implementation and testing are replaced by a formal transformation phase

3. AGILE PROCESS MODEL

These are advance methodologies based on multiple process model e.g. iterative and incremental models with customer collaboration and involvement [5]. On each iteration (complete in two or three weeks) software engineer listen customer story and goes through each step of SDLC, analysis, design, code, test etc. on each complete iteration, software engineer wins the customer satisfaction then they starts next iteration. Most commonly used methodologies for agile process model [6] are Scrum, Extreme programming, Agile Modeling, Agile Unified Process (AUP), Agile Data Method, Responsive Development, Test Driven Development (TDD), Feature Driven Development (FDD), Behavior Driven Development (BDD), Essential Unified Process (EssUP).

3.1. Critical distinctions between agile and other models:

- Accommodate changing requirements of customer on later stages.
- Win the customer satisfaction at each stage.

4. AGENT-BASED PROCESS MODEL

Constructing high quality software for complex system is still a debate. In order to achieve this goal many software development paradigm are offered. A recent contribution in field of software engineering is agent oriented software engineering [7, 8]. An agent is software that is capable for interacting with other agents in order to meet requirement of that particular system. Widely used agent-based model are GAIA methodology and x-machine frame work. According to [9, 10] an agent is an encapsulated computer system that is situated in some environment, and that is capable of flexible, autonomous action in that environment in order to meet its design objectives.

4.1. Critical distinctions between agent and other models:

- System based on agents situated in particular environment.

- Implement in environment where high integration between subsystems is required.

5. ASPECT ORIENTED PROCESS MODEL

Aspect oriented software development is advanced technique for software development. It emphasize on a new type of abstraction called aspect [11]. Aspect oriented Development (AOD) explicitly deals with separation of concern [12]. Primary objective of AOSD is to isolate cross cutting concern from its primary functionally called core concern. Where, Core concerns represent primary functionality of a system and cross cutting concerns represent non-functional requirements of software [13]. These cross cutting concern are modularized into separate units to increase reusability. This approach usually used in combination with object oriented software engineering. Aspect oriented (AO) development offers new benefits as well as new challenges to software community. Due to these new benefits and challenges it gains popularity in research community.

6. CONCLUSION

All three models are developed to manage complexities of software development. Regardless of similarities these software have many differences in term of specification, verification and testing.

In one perspective where Formal development process is very efficient, reliable and offer quality product, in other perspective formal people who have good knowledge about languages and tools for formal specification, designing, testing are required. Sometime special training sessions are required to train people involved in software development process [4][5].

Secondly, agile methodologies minimize gaps between customer and software engineer; on other hand it requires very experienced developers who understand business and administration as well as software development. And it is also difficult to estimate cost of project [6].

Thirdly, agents-based software development is a big revolution in software industry and agent-oriented decompositions are an effective way of partitioning problem of complex system, at the same time there are many pitfalls of agent models[8], one of them is understanding of the situations in which agent solutions are appropriate [9].

Beside these issues these development processes is widely accepted by software industry.

7. REFERENCES

[1] Pressman roger, prentice hall.3rd edition

[2]Research Triangle Institute, NIST Planning Report 02-3: The Economic Impacts of Inadequate Infrastructure for Software Testing", March 5, 2003.

[3] S.Patwa, A.K.Malviya. "Testability of software systems". International Journal of Research and Reviews in Applied Sciences, Vol: 5, 2010.

[4] What are formal methods? PF Gibbins, Information and Software Technology, Volume 30, Issue 3, April 1988, Pages 131-137

[5] J. A. Hall, "Seven myths of formal methods," IEEE Software, Sept. 1990, pp 11-19.

[6] Agile software development method: A review and analysis, VVT Publication.s

[7] Wooldridge M. J., Jennings N. R. and Kinny D. "The Gaia methodology for agent-oriented analysis and design". *Autonomous Agents and Multi-Agent Systems*, 3(3):285–312, September 2000.

[8] M. Wooldridge and N. R. Jennings (1998) "Pitfalls of agent-oriented development" Proceedings of the 2nd Int. Conf. on Autonomous Agents, 385-391, Minneapolis/St. Paul, MN.

[9] M. Wooldridge (1997) "Agent-based software engineering" IEE Proc. on Software Engineering, 144 (1) 26-37.

[10] M. Wooldridge and N. R. Jennings: "Agent Oriented software engineering"

[11] Ian Sommerville, Software Engineering (8th Edition), Pearson Addison Wesley, chap. 32 ,2004.

[12] R. T. Alexander, and J. M. Bieman, "Towards the systematic Testing of Aspect-Oriented Programs", IEEE, 2004.

[13] C. Chavez, and C. Lucena, "A Theory of Aspects for Aspect-Oriented Software Development", In Proc. of the7th Brazilian Symposium on Software Engineering (SBES'2003), 2003.

[14] Y. Han, G. Kniesel, and A. B. Cremers, "Towards Visual AspectJ by a MetaModel and Modeling Notation", 6th International

Workshop on Aspect-Oriented Modeling, 2005.

[15] Przybylek, “Separation of Crosscutting Concerns at the Design Level: an Extension to the UML Metamodel”, Proceedings of the International Multiconference on Computer Science and Information Technology, IEEE, 2008.

[16] M. Basch, and A. Sanchez, “Incorporating Aspects into the UML”, Proceedings of the third International Workshop on Aspect-Oriented Modeling, 2003.

[17] S. Salahuddin. "Analysing the Impact of Change on Test Sets Using X-Machines", 1-4244-1256-0, IEEE 2007.

[18] M. Holcombe, "X-Machines as a basis for dynamic system specification". Software Engineering Journal, vol. 3, no. 2, pp. 69-76.

[19] K. Bogdanov, "Testing from X-Machines Specifcations", Formal Methods and Testing, Springer-Verlag Berlin Heidelberg, pp.184-208, 2008.

[20] D. Dranidis, D. Kourtesis, and E. Ramollari, “Formal verification of Web service behavioral conformance through testing”. Annals of Mathematics, Computing & Teleinformatics, vol. 1, pp. 36-43, 2007.

[21] M. Holcombe, F.IPate, IFSoft, and Romania. “Using State Diagrams to Generate Unit Tests for Object-Oriented Systems”. Extreme programming and agile processes in software engineering, 2005.

[22]. E. Kehris, G. Eleftherakis and P. Kefalas, “Using X-machines to model and test discrete event simulation programs”, Proc. 4th World Multiconference on Circuits, Systems, Communications and Computers, Athens, 2000.