AGILE SOFTWARE DEVELOPMENT AND REUSABILITY

Jaspreet Singh*

Ashima Singh**

ABSTRACT

The increased speed and change in business world increased the need to develop software faster and cheaper as well as higher quality and more adaptable to constant change. New software development methods were developed and named as being agile. Despite the variety of literature about agile software development, we could not find any that would discuss possible bottlenecks of agile software development. However, according to Goldratt [10] every process has a bottleneck – a weakest link in the chain that limits throughput.

Identifying and eliminating bottlenecks, it will increase throughput what leads to more profit. Reusability is one of the bottlenecks of Agile Software Development process. Reusability finds limited scope in Agile Software Development and therefore adding reusability to Agile Software Development is a challenge. Essence of Agile Software Development is rapid software development and less cost. Thus, somewhere it compromises with quality and also unable to provide reusability of its developed components. Agile Software Development provides specific solutions whereas Reuse and Component based Development believe in generalized solutions.

*Department of Computer Science and Engineering, Thapar University, Patiala.

**Assistant Professor, Department of Computer Science and Engineering, Thapar University.

INTRODUCTION

The Agile Manifesto [11] stresses the importance of a) people and interactions over processes and tools, b) working software instead of detailed documentation, c) active customer participation and involvement rather than time and effort expended on negotiating contracts, and d) willingness and ability to take on changes over steadfast commitment to a static plan. Agile software development methods including eXtreme Programming (XP), Scrum, Adaptive Software Development and Feature-Driven Development are based on the principles of the Agile Manifesto and geared towards realizing its goals and objectives. In general, the feedback from organizations that have implemented agile development is positive. Some of the benefits attributed to agile development are increased productivity, expanded test coverage, improved quality/ fewer defects, reduced time and costs, understandable, maintainable and extensible code, improved morale, better collaboration, and higher customer satisfaction.

Despite the fact that all of the agile software development methods have the same goal, each of them has a different approach. This was a base for researchers to look into differences and similarities of agile software development methods from various angles [1][2][3]. Many case studies were performed investigating if and when agile implementations work [4][5][6]. Researchers investigated how to fit agile methods for large organizations [7] and traditional development organizations [8]. Besides, they investigated even more specifically: e.g. how to manage requirements in agile processes [9] and what are the various bottlenecks in it. The adoption of agile development has also revealed some challenges such as slow participant buy-in, opposition to pair-programming, lack of detailed cost evaluation, scope creep, reduced focus on code base's technical infrastructure and maintainability, difficulty evaluating and rewarding individual performance, and the need for significant on-site customer involvement, management support, competent managers and developers, extensive training and lack of reusability in Agile Software Development.

The following quotes offer descriptions of agile development with some arguing that its practice is nothing new:

"Agile does a set of values and principles and many powerful practices, but all of them have to be adapted to fit to each development situation. There is no recipe, rather a set of cooking techniques that must be adapted based on many variables -- all of which must be understood by the chefs in order to produce a pleasant tasting stew."

"Agile Processes are not "new" they are trying to strike a balance between common sense, the "urge" to code, and "analysis paralysis"."

"Agile development is a development strategy where the most important issue is to produce a software solution that fits exactly what the client needs with the least possible resources and in a very short period of time."

Lack of reusability in Agile Software Development process

Agile Software Development methods and techniques are being followed in the industry from the last decade to get quality product and to reduce development time. Rapid development and accommodate changes at any level of development gives the competitive advantage to the agile processes over traditional processes. But to get best and being new in software engineering, research on agile processes is going on as to combine light-weight processes and other processes. Recent research shows that only limitation of Agile software Development is its inability to reuse components those are developed through agile processes. On the whole rapid software development ignores reusability.

Agile processes such as Extreme Programming focus on building software products that solve a specific problem. Development in "Internet time" often precludes developing generalized solutions even when it is clear that this could yield long-term benefits. In such an environment, the development of generalized solutions and other forms of reusable software (e.g., design frameworks) is best tackled in projects that are primarily concerned with the development of reusable artifacts. This separation of the product-specific development environment from the reusable artifact development environment is a primary feature of the reuse-oriented framework called the Experience Factory developed by researchers at the University of Maryland at College Park [12]. The wide applicability of a reusable artifact requires that the process used to build the artifact emphasize quality control because the impact of low quality (in particular, severe errors) is as wide as the number of applications that reuse the artifact. On the other hand, timely development of reusable artifacts is desirable. While there seems to be a case for applying agile processes to the development of reusable artifacts, it is not clear how agile processes can be suitably adapted.

Approaches to achieve Reusability

To share code between different applications is considered to be the reusability but a variety of assets can be reused across software development processes as shown in Table given below

Intermediate Artifact	Implemented Artifact	Project Management and Quality Assurance Artifact
Requirements	(Sub) Systems	Process Models
Architectures	Frameworks, Components, Modules, Packages	Planning Models
Designs	UML Models, Interfaces, Patterns	Cost Models
Algorithms	Libraries	Review and inspection Forms e.g. checklists
Documentation (including templates)	Test Cases	Analysis Models e.g. Performance ,reliability
	Classes, Procedures, Routines, Functions, Methods, Source Code , Data	Design and Coding Conventions

Table 1: Possible Assets for Reuse [13]

Reusability increases not only the productivity of the developers but also the reliability and maintainability of the software products. Many software companies have repository to support the reusability. Object-Orientation also offers reusability.

No of techniques are available to support reusability. Considerable research and development is going on in reuse; industry standards like CORBA have been created for component interaction; and much domain specific architecture , toolkits, application generators and other related products that support reuse and open systems have been developed [14]. Reuse Landscape as shown in figure 1 depicts the different reuse approaches.

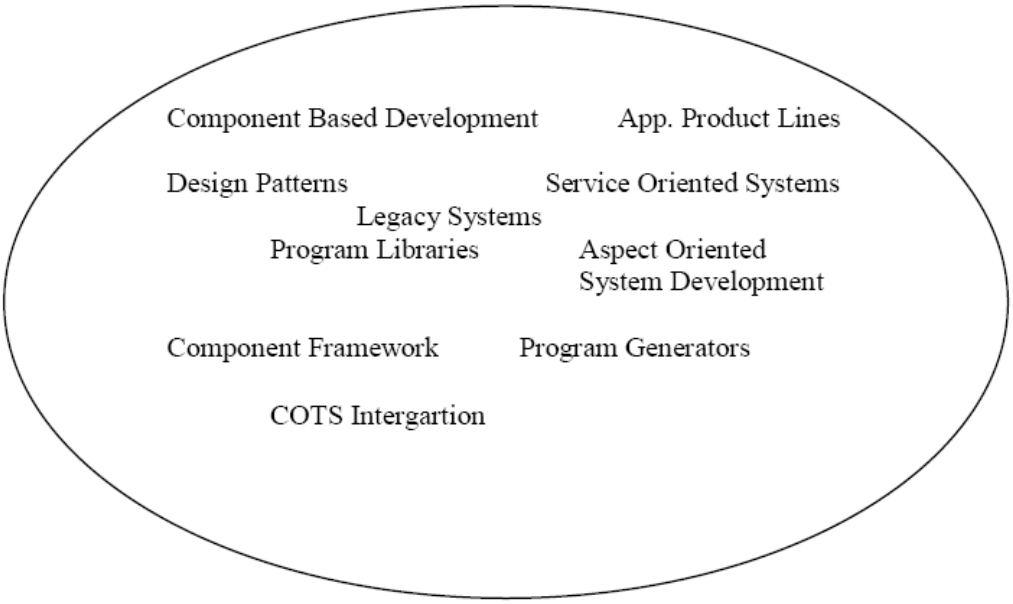

Figure 1: Reuse Landscape [15]

Reusability in Agile Software Development

There are some ways or technologies discussed one by one below, by which reusability can be incorporated in agile software development.

1. Component Based Development (CBD)

CBD is a reusability approach that can be found in Microsoft.NET Framework and J2EE (Java2 Enterprise Edition). Component Based Software Engineering(CBSE) process identifies not only candidate components but also qualifies each components interface, adapts components to remove architectural mismatches, assembles components into selected architectural style, and updates components as requirements for the system change .

2. Refactoring to Design Patterns

To provide a software system quality in terms of reusability, flexibility and extendibility, refactoring is a significant solution. Use of design patterns in an application increases reusability and maintainability. One new emerging approach that refactoring to design patterns seems to have promising future in reusability discipline. Research is going on in the field of pattern mining as to find new approaches and new tools.

3. Reusable Architectures

Reusable architectures can be developed from reusable architectural patterns as in Forefront Identity Manager Architecture. which operates at three different levels of reuse: Federation, domain and application. Focuses on how non-functional property reusability relates to the software architecture of a system. Presented a suggested software process model for reuse based software development approach.

CONCLUSION

Agile software development emerged as a need to respond to the rapidly changing market. Creating software using document driven, rigorous software development processes became too slow. Many agile software development methods were developed and successfully implemented in different organizations. Despite the fact that all agile software development methods follow the same values, they address them in different ways. They all have bottlenecks (week parts) that should be carefully monitored while implementing the method.

From the overall literature survey, we conclude that agile development which has promising future in the software industry and is capable of fulfilling the demands of the industry. It will be accepted widely if pattern based architecture designing, design patterns, UML based analysis and designing is incorporated. Pattern based architecture oriented agile development and use of OO patterns as refactoring to design patterns will make a space for reusability and reusable artifacts. Reuse based software engineering and agile development is an open research area in rapid development. A tool that would be capable of learning from components and design patterns stored in agile repository can be designed using pattern matching techniques, neural networks or machine learning algorithms. And it can be further developed to make the whole or some part of the process automated. Also, incorporating little bit of object oriented designing can create much scope of reusability in Agile Software Development.

REFERENCES

1. David J. Anderson: “Agile Management for Software Engineering. Applying the Theory of Constraints for Business Results”, Prentice Hall, 2006
2. Jim Highsmith: “Agile Software Development Ecosystems”, Addison-Wesley, 2002
3. Craig Larman: “Agile and Iterative development: a Managers Guide”, Addison-Wesley, 2004
4. Jonathan Rasmusson: “Agile Project Initiation Techniques – The Inception Deck & Boot Camp”, Proceedings of AGILE 2006 Conference (AGILE’06), 2006
5. Sanjiv Augustine, Fred Sencindiver, Susan Woodcock: “Agile Project Management: Steering From The Edges”, Communications of the ACM, Vol. 48, No. 12, 2005
6. Damon Poole: “Breaking the Major Release Habit”, ACM Queue, October, 2006
7. Mikael Lindvall, Dirk Muthig, Aldo Dagnino, Christina Wallin, Michael, Stupperich, David Kiefer, John May, Tuomo Kähkönen: “Agile Software Development in Large Organizations”, IEEE Computer Society, December, 2004

8. Barry Boehm, Richard Turner: "Management Challenges to Implementing Agile Processes in Traditional Development Organizations", IEEE Software, 2005
9. Lan Cao, Balasubramaniam Ramesh: "Agile Requirements Engineering Practices: An Empirical Study", IEEE Software, 2008
10. Eliyahu M. Goldratt: "Critical Chain", The North River Press, 1997
11. Agile Manifesto, "Manifesto for Agile Software Development," http://agilemanifesto.org/, December 2007.
12. Basili, V., Rombach, D. Support for comprehensive reuse. Department of Computer Science, University of Maryland at College Park, UMIACS-TR-91-23, CSTR- 2606, 1991.
13. www.win.tue.nl/~mchaudro/cbse2007/managing%20CBSE%20and %20reuse.pdf
14. Garlen D., Allen R., and Ockerbloom J., "Architectural Mismatch : Why Reuse is So Hard", IEEE Software, November1995, vol. 12, no 6, pp 17-26.
15. Pressman R. S., "Software Engineering" , 7th edition, McGraw Hill Education,2009.