ASIAN JOURNAL OF PHARMACEUTICAL AND CLINICAL RESEARCH

INNOVARE ACADEMIC SCIENCES
Knowledge to Innovation

Special Issue (April)

Online - 2455-3891
Print - 0974-2441

Full Proceeding Paper

IMPLEMENTATION OF REUSE IN THE AGILE SOFTWARE DEVELOPMENT PROCESS SCRUM

JAYASUDHA R[1], VISWANATHAN V[2]*, SHANTHI P[1]*

[1]Department of Computer Applications, Sri Krishna College of Engineering and Technology, Coimbatore, Tamil Nadu, India. [2]School of Computing Sciences and Engineering, VIT University, Chennai, Tamil Nadu, India. Email: viswanathan.v@vit.ac.in/shanthi@skcet.ac.in

Received: 23 January 2017, Revised and Accepted: 03 March 2017

ABSTRACT

The concept of reuse is applied in one of the agile development methodologies called the scrum. Sprint is a single functionality and the result at the end of the sprint functionality is derived as the shippable or bugs. This paper makes an attempt to use the concept of reuse in the agile software development to meet the dynamic change of customer requirements in banks. A banking project is created using both waterfall model and scrum model, and the knowledge gained is stored in the ontology-based repository for the first time. Again, the same project is created for different vendors using the ontology-based repository. The result shows that maximum sprint is reused and all the knowledge gained is stored in the form of ontology. This ontology helps identify the shippable component of each sprint which is a small executable functionality. This leads to less cost and time to deliver the product. The main aim is to increase the availability of the reusable artifacts, which lead to increase the reusability of the developer. The experimental results show improvements in the performance of retrieving the components for the software development.

Keywords: Scrum, Agile software development, Sprint ontology.

 DOI: http://dx.doi.org/10.22159/ajpcr.2017.v10s1.19597

1. INTRODUCTION

Software development process needs to be improved various studies and suggestions have been done on this field. There is a need to implement the fast changing organizational and business needs using traditional methods; therefore, agile methods were introduced. In agile methods [1], various artifacts are developed through various iterations, which will be a workable product and gives customer satisfaction. Agile software development emphasizes on plan-based control since this method needs to do frequent change during the project; this is the main difference between the current traditional method and the agile method.

Some of the difficulties in the existing traditional methods are as follows:

- New requirements: Requirements are changing due to evolving business needs. Customers do not have a clear vision about the specifications of their requirements at the early stages. Some customers realize what their true requirements are only when they use an application that does not really meet their needs. Another source of change comes from experiences gained during the development.
- Customer involvement: Lack of customer involvement leads to higher chances of project failure. Many companies usually do not allocate any effort for customer involvement.
- Deadlines and budgets: Often, customers do not accept failure. On the other hand, companies usually offer low budgets, tight deadlines, while at the same time, requiring high demands, due to competition in the markets.
- Miscommunications: One cause of the misunderstanding of requirements is the miscommunication between developers and customers. For example, each party uses its own jargon, and this leads to misunderstanding of customer's needs.

With the existence of such problems, the object-oriented software development methodologies cannot satisfy the objectives of software development companies. New development methodologies have to be applied to overcome these problems [2].

The agile software development is a recent software development methodology based on the concept of incremental and iterative development. In this method, various phases are visited over and over again. It takes customer feedback to improve the quality of software and to meet the end requirements. It gives more importance to customer participation. The agile method believes in testing of the project at the beginning of the project and continuing it throughout the project. The main improvement factors of agile program over the traditional method are as follows:

1. Customer involvement in the early stage
2. Iterative development
3. Self-organizing teams
4. Adaptation to change.

Agile authors built their methodologies on four principles. First, the main objective is to develop software that satisfies the customers, through continuous delivering of working software, and getting feedback from customers about it. The second principle is accepting changes in requirements at any development stage so that customers would feel more comfortable with the development process. The third principle is the cooperation between the developers and the customers (business people) on a daily basis throughout the project development. The last principle is developing on a test-driven basis, that is, to write test before writing code. A test suite is run on the application after any code change.

Agility in short means to strip away as much of the heaviness, commonly associated with traditional software development methodologies, as possible, to promote quick response to changing environments, changes in user requirements, accelerate project deadlines, and the like. Agile methodologies prefer software development over documentation. Their philosophy is to deliver many working versions of the software in short iterations and then update the software according to customer's feedback. Applying this philosophy will help overcome the problems mentioned earlier, by welcoming changes, satisfying user requirements, faster development, and at the end, users will have just the system they need. In any project, the user requirements keep changing dynamically. To create such dynamic

projects, agile software development methods have been followed by the traditional methods for the development.

Section 2 discusses the various agile methods and its process; scrum method and different terms used in this method are discussed in Section 3. Proposed architecture and the implementation of scrum ontology are discussed in Section 4. The result is analyzed to improve the reusability in the scrum method in the 5th section.

AGILE METHODS

The popular agile methods are as follows:

- Extreme programming
- Crystal methodologies
- Feature-driven development (FDD)
- Adaptive software development (ASD)
- Scrum.

Extreme programming XP is a package of several practices and ideas, most of which are not new. The process of extreme programming is shown in Fig. 1. The combination and packaging of all of these are, however, new. Extreme programming was in fact targeted especially at small co-located teams developing non-critical products. It has been suggested that the early adopters of agile methods have been small high-tech product companies. Currently, however, it has already been proven at many companies of all different sizes and industries worldwide [3].

XP provides a list of simple, specific, and seemingly naïve principles and values that guide the software development process throughout the main four phases of software development: Planning, coding, designing, and testing. The main purpose is to deliver what the customer needs, at the time it is needed. In addition to this, one of the main reasons of its success is its ability to accept changes at any time during the development. XP also emphasizes teamwork; experiences from all stakeholders are employed to meet the specific goals and within the given constraints [7].

Crystal methodologies were established by Cockburn in 2000. They concentrate on efficiency and habitability as components of project safety. Each of the crystal methodologies requires certain roles, policy standards, products, and tools to be adopted. Crystal clear, which is one of the crystal methodologies, can be applied to development teams of six to eight members, working on non-life critical systems. It focuses on people, not processes of artifacts [8].

FDD was founded by Jeff De Luca and Peter Coad. It combines some practices recognized in the industry into one methodology. These practices are all determined from a client-valued functionality (feature) viewpoint. As of other agile methodologies, its key goal is to deliver tangible, working software repeatedly in a timely manner [4].

ASD was created in 2000 by Jim Highsmith. It has grown out of the rapid application development. Like other agile methodologies, ASD aims to increase a software organization's responsiveness while decreasing development overhead (Maurer and Martel 2002). It embodies the belief that continuous adaptation of the process to the work at hand is the normal state of affairs. Scrum is an agile process most commonly used for product development, especially software development. Scrum is a project management framework that is applicable to any project with aggressive deadlines, complex requirements, and a degree of uniqueness. In scrum, projects move forward via a series of iterations called sprints. Each sprint is typically 2-4 weeks long (Sutherland and Schwaber, 2011).

SCRUM OVERVIEW

Scrum (n): A framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value [5]. Scrum is lightweight, simple to understand, and difficult to master.

Software development using agile methodologies is becoming a bigger reality in the daily life of software development companies. Agility brings quality to the software development and management process. To add value to the final software, one must have a well-structured team that follows the methodology and uses correct strategies [6]. An introduction to scrum would not be complete without knowing the scrum terms you will be using. This section in the scrum overview will discuss common concepts in scrum. The terms used in the scrum and the process of scrum are shown in Fig. 2.

Scrum team

A typical scrum team has between five and nine people, but scrum projects can easily scale into the hundreds. However, scrum can easily be used by one person team. This team does not include any of the traditional software engineering roles such as programmer, designer, tester, or architect. Everyone on the project works together to complete the set of work they have collectively committed to complete within a sprint. Scrum teams develop a deep form of camaraderie and a feeling that "we're all in this together."

Product owner

The product owner is the project's key stakeholder and represents users, customers, and others in the process. The product owner is often

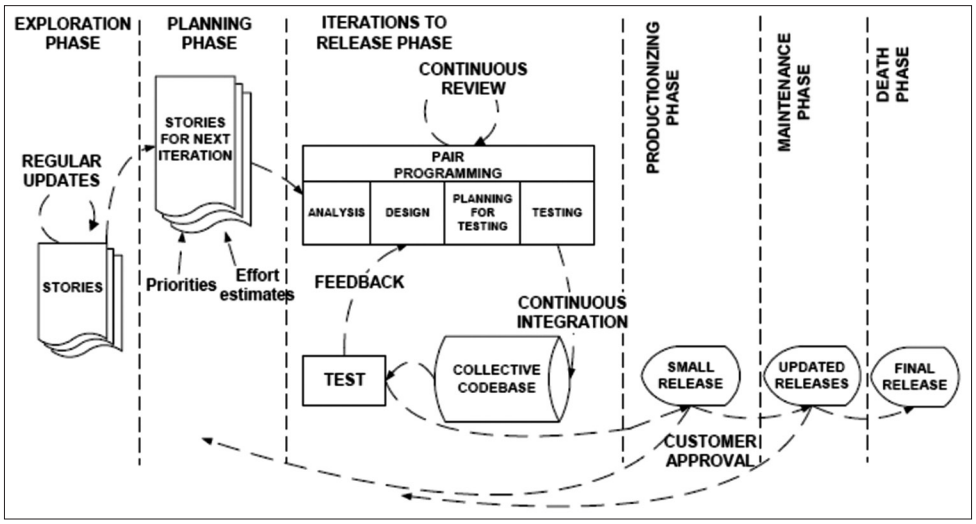

Fig. 1: Extreme programming [3]

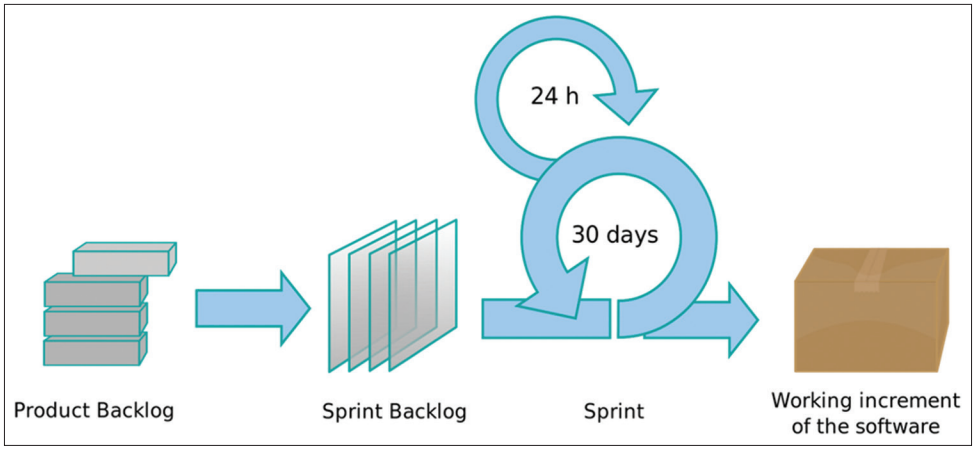

Fig. 2: Scrum process Sutherland and Schwaber (2011)

someone from product management or marketing, a key stakeholder or a key user.

Scrum master
The scrum master is responsible for making sure the team is as productive as possible. The scrum master does this by helping the team use the scrum process, by removing impediments to progress, by protecting the team from outside, and so on.

Product backlog
The product backlog is a prioritized feature list containing every desired feature or change to the product. Note: The term "backlog" can get confusing because it is used for two different things. To clarify, the product backlog is a list of desired features for the product. The sprint backlog is a list of tasks to be completed in a sprint.

Sprint planning meeting
At the start of each sprint, a sprint planning meeting is held, during which the product owner presents the top items on the product backlog to the team [9]. The scrum team selects the work they can complete during the coming sprint. That work is then moved from the product backlog to a sprint backlog and developed.

Daily scrum
Each day during the sprint, a brief meeting called the daily scrum is conducted. This meeting helps set the context for each day's work and helps the team stay on track. All team members are required to attend the daily scrum.

Sprint review meeting
At the end of each sprint, the team demonstrates the completed functionality at a sprint review meeting, during which the team shows what they accomplished during the sprint. Typically, this takes the form of a demonstration of the new features, but in an informal way, for example, PowerPoint slides are not allowed. The meeting must become neither a task in itself nor a distraction from the process.

Sprint retrospective
Furthermore, at the end of each sprint, the team conducts a sprint retrospective, which is a meeting during which the team (including its scrum master and product owner) reflects on how well scrum is working for them and what changes they may wish to make for it to work even better.

Fig. 2 shows the essential elements of using scrum for agile software development. On the left, we see the product backlog, which has been prioritized by the product owner and contains everything desired in the product that is known at the time. The 2-4 week sprints are shown by the larger green circle.

At the start of each sprint, the team selects some amount of work from the product backlog and commits to completing that work during the sprint. Part of figuring out how much they can commit to is creating the sprint backlog, which is the list of tasks (and an estimate of how long each will take) needed to deliver the selected set of product backlog items to be completed in the sprint.

At the end of each sprint, the team produces a potentially shippable product increment – i.e., working, high-quality software. Each day during the sprint, team members meet to discuss their progress and any impediments to completing the work for that sprint. This is known as the daily scrum and is shown as the smaller green circle above.

In early studies done on reusability in agile, it has been observed that most of the emphases were made on the classification of components in database so that extracting reusable components become easier. Identification of the reusable component plays a major part in this methodology [10]. It is important to find the components with higher reusability factor. There are number of factors which affect the reusability of components, and if value of these factors is high, reusability of component decreases. It is necessary to find those components which have low value of the factors having negative effect on reusability and then to add those components to database.

If this approach is adopted, time of finding reusable component from repository will be decreased and further efforts for modifying those components will be reduced because those components in repository already have high reusability value. In the existing agile process, reuse does not have much role. A lot of components are produced at the various processes [11]. For every change in the user requirement, new process has been created. This knowledge has to be stored for future processing. All the valuable knowledge acquired need to be recorded for better usage in the future.

The objective is to store all the dones into the repository, each done is a single unit of completed task or the user requirements. Thus, each done is a reusable component and stored in the repository [12]. Sprint may also be a reusable component with the collection of dones. Thus, in the new development environment, reuse can be easily applied to reduce the cost and time of software development.

The storage of dones helps retrieve more relevant reusable component than the normal traditional method of reusable component creation and storage [13].

PROPOSED ARCHITECTURE

In the proposed architecture shown in Fig. 3, all the user requirements are split into small executable requirements. These executable parts will do a functionality which needs some input and gives the output

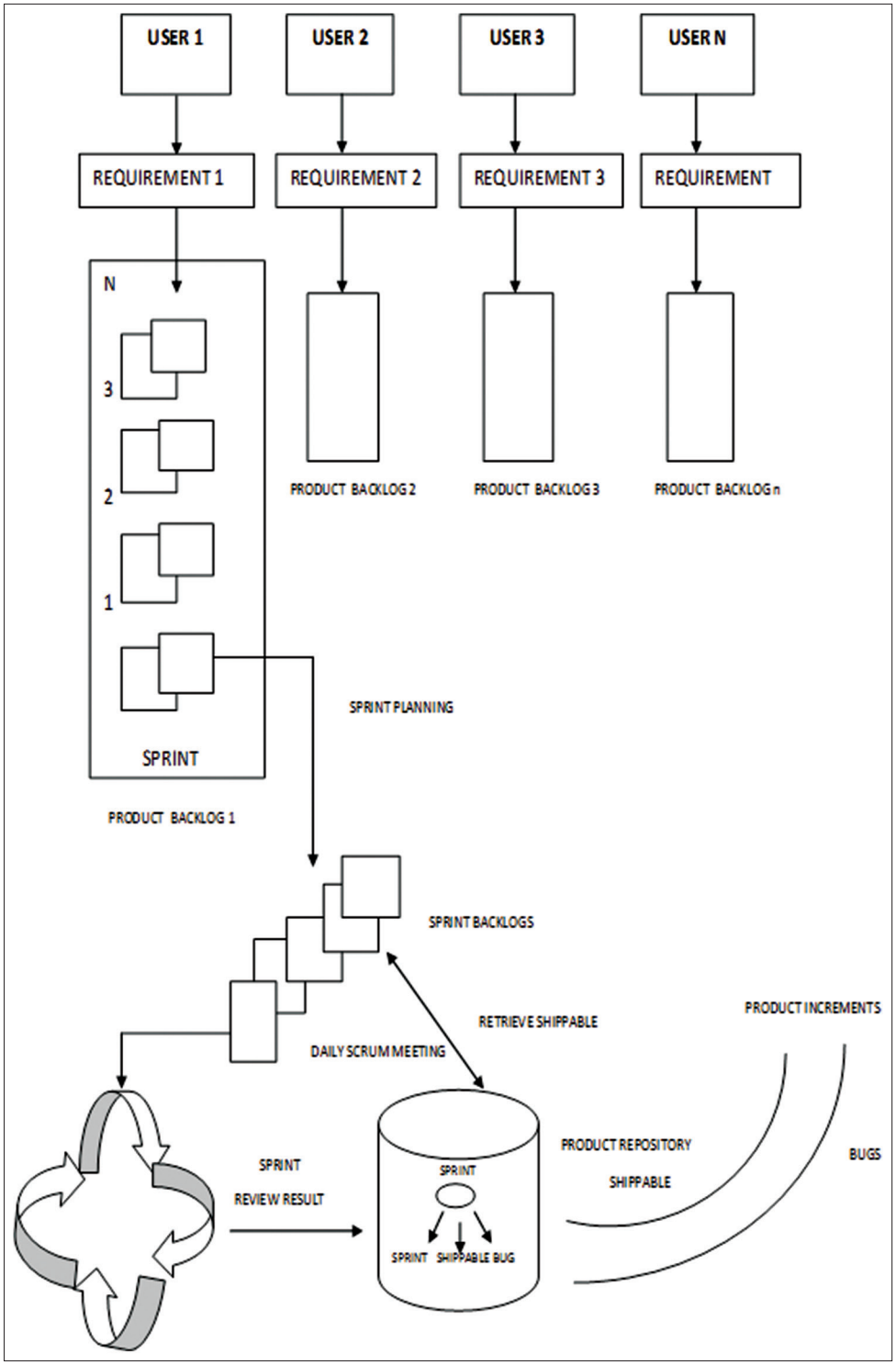

Fig. 3: Reuse-based scrum methodology

at the end. For every user, requirements are converted into a product backlog, which consist of very small units of component called the sprint. Each component in the product backlog is called sprint. Sprint consists of smaller units of work. Each sprint has to undergo sprint review process such as daily scrum meeting and sprint review. Sprint review is the knowledge gained and it should be recorded for the future usage. Hence, this is stored in the repository. The final stage of the sprint review is product increment which provides outputs such as sprint, shippable and bugs. Shippable are otherwise called dones. All the dones and bugs are stored in the repository for future usage. Sprint is also stored for future reference. Product repository is structured as sprint ontology. This classifies each sprint from the initial stage to the end of the final shippable product.

There is an improvement in the searching of ontology-based repository than the normal repository. This process will go till the final release of the product to the customer. The search of dones in correspondence to the sprint can also be done in the sprint ontology. If the matching dones are available, then that is reused. If the dones are not available, then it is created and stored in the repository for future reuse. Thus,

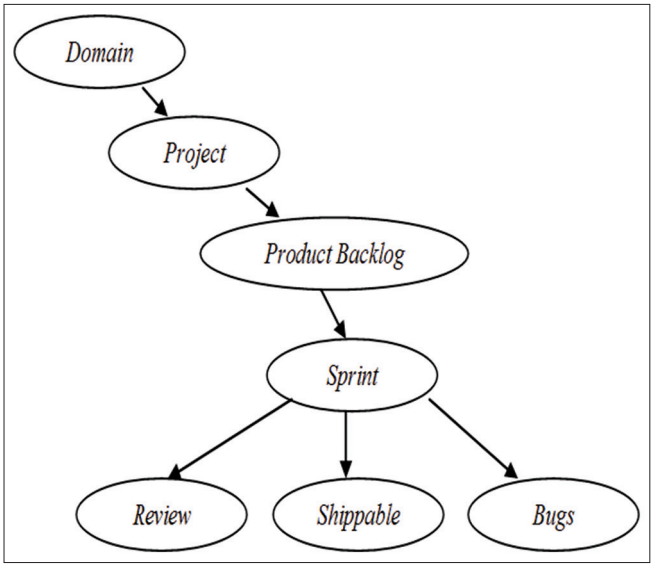

Fig. 4: Sprint ontology

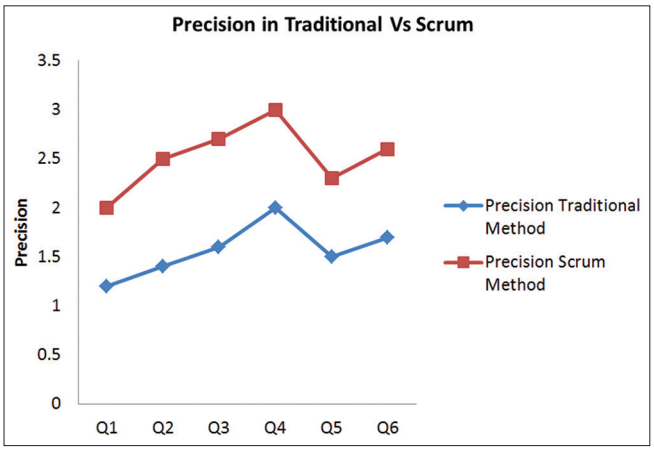

Fig. 5: Precision in traditional method versus scrum method

this whole process is used for both retrieval and storage of the reusable components known as the dones into the repository. The retrieval and storage of reusable component are indexed using the sprint ontology. The sprint details are stored in the form of sprint ontology where sprint of a particular domain is called class and properties of these classes are sprint review, date, shippable, bugs. This ontological storage methodology helps retrieve the needed dones of a particular sprint in an easier manner since it is semantically stored.

The sprint ontology showed in Fig. 4 acts as reference for the retrieval of dones. Each sprint class has the details of review, which will give us the discussions and knowledge gained during the scrum daily meetings. Shippable is the final customer accepted done and complete reusable component. Bugs contain the details of error occurred during the creation of products.

RESULTS AND DISCUSSION

This method is tested with the banking project which is developed both using traditional and scrum method, the creation of sprint ontology helps retrieve more relevant component than the ordinary traditional method. All the details such as scrum review, shippable, bugs are retrieved for a single requirement. This helps the developer to have more knowledge about the particular requirements. The developer can use this knowledge for further improvement.

From Fig. 5, it can be seen that precision of the scrum method is more than the traditional method for various queries. Thus, it is understood that the scrum methodology gains and stores more knowledge than the traditional method.

CONCLUSIONS

From the result, it is inferred that the usage of ontology-based repository improves the precision than the normal repository. All the knowledge gained was relatively stored for easy retrieval. It is also clearly observed that agile methodologies are inclined only toward the client's need. However, work is not done keeping reusability in mind. As in agile methodology, time spent on development is limited; hence, a compromise with quality is made as quality of software also depends on quality of code and documentation. Hence, there is a need of reusable artifacts (analysis, design document, patterns, etc.). Lack of documentation and design in development make it difficult to extract reusable functionalities. Because of this, difficulty level and cost of modification also increase. Hence, by adopting reusability in agile development, quality of system can be maintained and time can further be reduced to many folds. In this proposed work, we have added the concept of reusability at coding stage, which can help in reducing coding efforts and saving a lot of time.

REFERENCES

1. Elbanna A, Sarker S. Risks of Agile Software Development: Learning from Adopters, IEEE Software; 2016. p. 72-9.
2. Sharma A, Beniwal MK. Software development life cycle - Traditional and agile-comparative study. IJSRD Int J Sci Res Dev 2013;1:2321-0613.
3. Cao DB. An Empirical Investigation of Critical Success Factors in Agile Software Development Projects. Ph.D. Thesis, Capella University, USA; 2006.
4. Bari MA, Ahamad S. Managing knowledge in development of agile software. Int J Adv Comput Sci Appl (IJACSA) 2011;2(4):72-6.
5. Rola P, Kuchta D. Implementing scrum method in international teams-a case study. Open J Soc Sci 2015;3(7):300-5.
6. Lima IR, Freire TD, Costa HA. Adapting and using scrum in a software research and development laboratory. Rev Sist In FSMA 2012;9:16-23.
7. Ahuja MS, Sadana N. Agile methodology and software reuse a common approch to software development. Haryana, India: Shivalik Institute of Engineering and Technology; 2012.
8. Hneif M, Ow SH. Review of agile methodologies in software development. Int J Res Rev Appl Sci 2009;1(1):1-8.
9. Pathak K, Saha A. Review of agile software development methodologies. Int J Adv Res Comput Sci Softw Eng 2013;3(2):270-6.
10. Rao KN, Naidu GK, Chakka P. A study of the agile software development methods, applicability and implications in industry. Int J Softw Eng Appl 2011;5(2):35-45.
11. Singh J, Singh A. Agile software development and reusability. IJREAS 2012;2(2):1181-7.
12. Szalvay V. An Introduction to Agile Software Development. Bellevue, WA: Danube Technologies; 2004. p. 1-9.
13. Spoelstra W. Reusing software assets in agile development organizations - A management tool: A case study at a medium sized software development organization. Hengelo: Business and Information Technology, School of Management and Governance; 2010.
14. Sutherland J, Schwaber K. The scrum papers: Nut, bolts, and origins of an Agile framework. One Broadway: Scrum Inc; 2011.