Volume 2, Issue 10, October 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Novel Approach to Implement Burndown Chart in SCRUM Methodology

Tanzeem Bin Noor[1], Hasib Bin Shakur[2], Md Khairul Bashar Chowdhury[3], Imrul Mukit Siddique[4]
[1,2]*Lecturer, Dept. of Computer Science & Engineering, Stamford University Bangladesh*
[3,4]*Software Engineering, NilavoTechnologies Ltd.*

Abstract— Software development methodology is a pool of rules, theories, and methods that provide a paradigm for planning, developing and maintaining software engineering projects. The Agile methodology is a software development methodology that provides iterative and incremental solutions towards building software's of varying scales. Scrum is a popularly practiced agile software development method in the software industry. In order to track down the progress of a project a Burn Down Chart is used in Scrum where remaining tasks are plotted against working days and the expected and actual progress lines are compared to determine the overall progress of a project. But this comparison often fails to indicate the reasons for not achieving the expected goal even with taxing efforts of the software development team, as there is no reflection of any unplanned task in the Burn Down Chart. In this paper, we have proposed a modified version of the original Burn Down Chart which along with displaying the existing expected and actual progress line also indicates to what extent unplanned tasks have affected the progress of the project upon their existence.

Keywords— Agile, Scrum, Sprint, Sprint Backlog, Burn Down Chart

I. INTRODUCTION

The Software Development process advances through a series of steps. For successful completion of software projects there arise several challenges throughout the software development life cycle. User's requirement, designer's planning, adoption of new features and developer's knowledge are needed to be integrated properly in the development of a software project. A number of tools or methodologies have been evolved for the successful delivery of software. Different software development approaches share common principles such as improvement of customer satisfaction, management of change requirements, delivery of working software or software modules, and creation of collaboration among stakeholders and developers [1]. Methodologies enforce a well-organized process of software development with an aim of making the software project more predictable and more proficient by maintaining high quality.

Agile software development methodology is a collection of software development methods based on iterative and incremental development [2] that promotes adaptability and open collaboration between self-organizing, cross-functional teams throughout the life cycle of the project. According to the Agile Manifesto principles [3], more emphasis is placed on customer satisfaction, interaction with the customer, working software, customer collaboration, and change requirements, rather than on processes, tools, contracts and plans. Scrum is one of the agile software development methods which are broadly used in software development firms.

After initial planning, a series of short development phases, or *Sprints*, deliver the product incrementally. A sprint typically lasts one to four weeks. A closure phase usually completes product development. The team tracks all currently identified tasks, capturing them in a list called the *Backlog*. The backlog drives team activity. Before each sprint, the team updates the backlog and reprioritizes the tasks—each team signs up for a number of tasks and then executes a sprint [4]. Generally a spreadsheet named *Burn Down Chart* is used to monitor project progress with the current work remaining. During the sprint this chart is daily updated. In a burn-down chart, time proceeds along the X-axis whereas the remaining work is plotted on Y-axis. In ideal scenario line slopes down with time and reaches to "zero" at the end of the sprint. Some agile methodologists prefer to use *Burn Up Chart*, a modification of Burn Down Chart. In a Burn Up Chart the X-axis represents iterations or time and Y-axis displays accumulated story points. Total planned work and total produced work are represented by two different lines [5].The remainder of this paper is structured as follows: The history of the Scrum method along with its pros and cons is overviewed in Section II. Section III describes the detailed methodology of Scrum. Case Studies have been discussed in Section IV. Section V analyzes the Burn Down Charts of the case studies. Alternative Burn Down Chart has been proposed in Section VI. Finally, Section VII summarizes and concludes this paper.

II. SCRUM IN SOFTWARE DEVELOPMENT

Scrum is a process used for agile software development. Rather than a full process or methodology, it is an agile framework for effective team collaboration on complex projects. In Scrum, a structure is created by a small set of regulations for teams so that they can focus their development efforts and deliver products of highest possible values.

When Jeff Sutherland applied the first Scrum process in 1993, he borrowed the term "Scrum" from a study of comparable high-performing, cross-functional teams to the Scrum formation used by Rugby teams described by Takeuchi and Nonaka in 1986 [6]. After a joint presentation on *Scrum methodology* by Sutherland and Schwaber in 1995[7], Schwaber worked with Mike Beedle to describe the method in the book *Agile Software Development with Scrum in* 2002[8]. Today, Scrum is used by different companies including: Yahoo!, Microsoft, Google, Johns Hopkins APL, Siemens, Cisco, Motorola- SAP, Nokia, GE, US Federal Reserve, Capital One [9].

The inherent properties of Scrum have made this methodology widely adopted. The object oriented approach of the Scrum methodology welcomes changing requirements and focus on prioritizing different tasks to ensure highest quality. Scrum ensures team spirit through better communication and mutual understanding among team members. Teams are self-organized and free to develop ideas and solutions to get the best potential formula in order to complete the project properly, and as efficiently as possible. Overall, the lack of external policy and procedure makes Scrum a useful and unparalleled approach to project management and effective workflow [10]. However, Scrum is a small team methodology with co-located and fully committed team. Iteration planning needs a lot of expertise and daily Scrum meeting sometimes offers obstacles or overhead to team members as well as for organizations. Additionally finding a meta-Scrum master and capable product owner are other challenges for Scrum being implemented accurately and appropriately.

III. DETAILED METHODOLOGY

Scrum is an agile framework that consists of Scrum Teams and their associated roles, events, artifacts, and rules [11].

A. The Scrum Team

A Product Owner, the Development Team, and a Scrum Master together form a self-organizing and cross-functional Scrum Team. This team figures out the best approach to carry out their tasks directed by the members of the team without depending on others.

1) The Product Owner: Product owner is the project's key stakeholder who conveys the vision of the product to the team by defining a prioritized feature list for the product. The product owner is a person rather than a committee but however he can represent a committee.

2) The Development Team: An ideal development team consists of 3-8 professionals empowered by organization who do the work for delivery of products with significant increment. Generally the team comprises with programmers, QA (Quality Assurance) professional and analysts.

3) The Scrum Master: Scrum Master is a servant-leader for the Scrum Team and works as an interface between the product owner, development team and organization. It is the responsibilities of Scrum Master to ensure that Scrum is well understood and the practice, rules of Scrum theory are adopted properly.

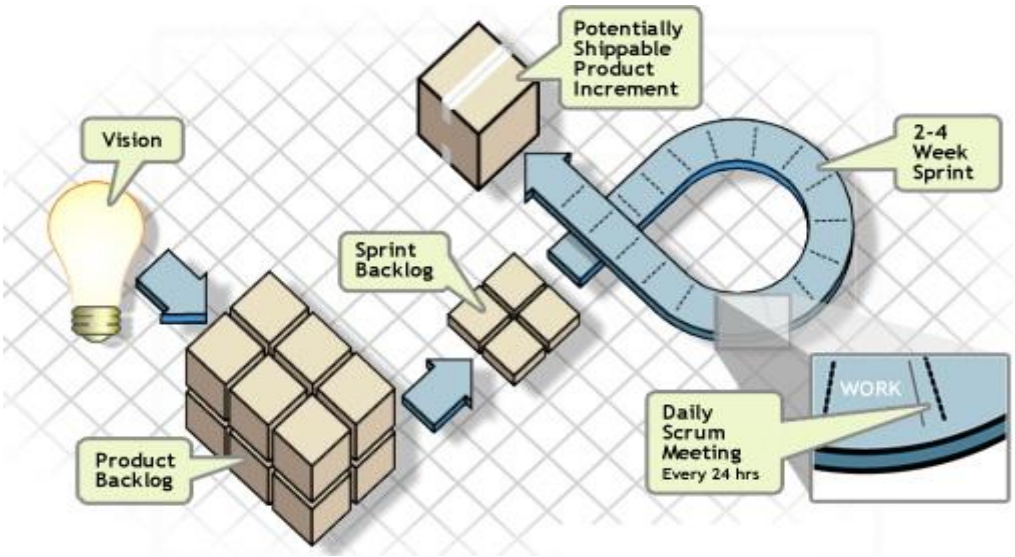


Fig.1 Scrum Methodology [12]

B. Scrum Events

Scrum framework is maintained by Sprint which consists of Sprint Planning Meeting, Daily Scrums, the development work, the Sprint Review, and the Sprint Retrospective.

4) The Sprint: Sprint is a confined work cycle or set period of time during which potentially releasable product increment is created. The sprint lasts for 30 days or less. Before this time box is over, if sprint goals become obsolete then the product owner can cancel the sprint.

5) Sprint Planning Meeting: The collaborative work of the entire Scrum Team determines the sprint goal of the Sprint planning meeting. They agree on the functionalities that will be developed during the sprint and also design the system and workflow to achieve the functionalities.

6) *Daily Scrum:* The daily Scrum meeting is a 15 minute event that held on every day at the same time and same place. The meeting reviews the last day's task status and explains what will be done for that day with task assignment to individual members of the development team.

7) *Sprint Review:* At the end of the sprint a sprint review meeting is held to look over the overall progress of the sprint evaluated by product owner. This is an informal demo that presents the features accomplished during the sprint.

8) Sprint Retrospective: In a short meeting of sprint retrospective, scope of improvement and opportunities are planned by the development team after sprint review.

C. Scrum Artifacts

Three types of scrum artifacts represent the value or work to ensure success of scrum team with maximum clarity.

9) Product Backlog: The product backlog is a prioritized list that consists of all requirements and features managed by the product owner. It is often ordered by importance, risk, values or priority. Sometimes smallest unit of product backlog is referred to as a story.

10) Sprint Backlog: From product backlog, a set of items is selected to perform for the sprint, is referred to as sprint backlog. Product requirements are broken into manageable tasks and development team works on the tasks to make it "Done" for the sprint increment.

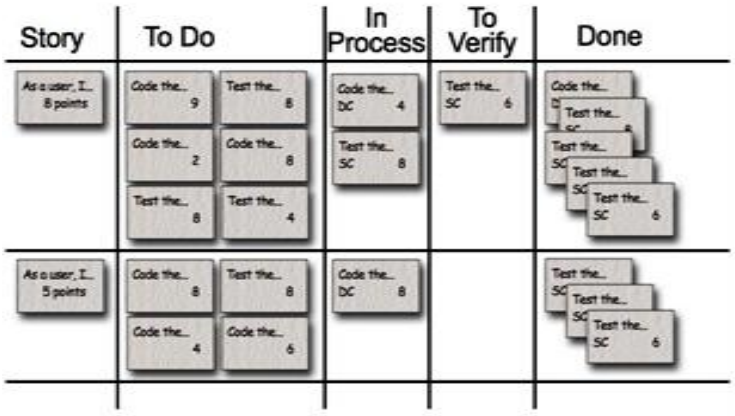
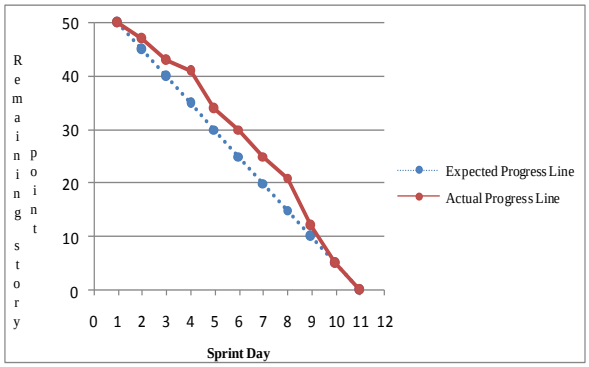


Fig.2 Scrum Task board [13]

11) *Increment:* Product backlog items completed during the sprint indicate the increment. In this stage, the features aimed for the sprint must be at usable condition that refers to "Done" state defined by the Scrum team.

12) Burn Down Chart: To observe the progress of project development Burn Down Chart is used. It illustrates the remaining work in respect of time. The horizontal axis of Burn Down Chart displays time; the vertical axis shows the amount of work (story point, work hour, team days) remaining.

A sprint Burn Down Chart exhibits daily progress. The remaining work should reduce at every day of a sprint, so the ideal scenario for sprint Burn Down Chart would look like downward straight line as Fig. 3. However, in real time some deviations from actual straight line exist due to wrong estimation, programmer inefficiency or addition of new requirements by the product owner.



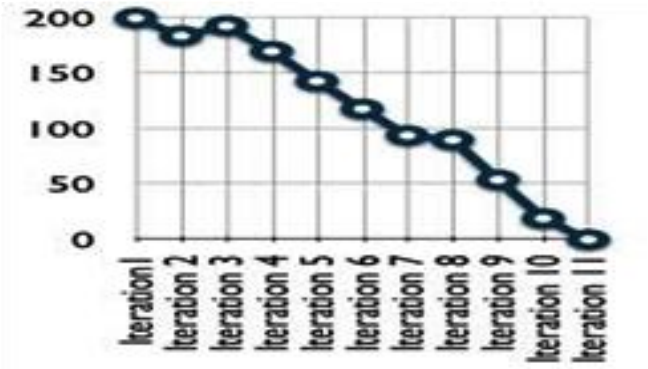


Fig.3 Sample Burn Down Chart

Fig.4 Iterations in Scrum [14]

On the other hand, product's release Burn Down Chart demonstrates monthly (per iteration or sprint) progress. According to the release Burn Down Chart of Fig 4 it is observed that within planned 11 iterations, estimation was changed during the 2nd iteration because of new requirements and standard progression was slowed down during the 7th iteration.

IV. CASE STUDIES

A. *Case Study 1*

***Project*:** Stock market portfolio management system of Non Resident Investor Taka Amount (NITA) account holders of XYZ Bank

Project Description: The aim of this project is to provide a stock market portfolio management system for the NITA account holders of the XYZ bank. Users can see their daily portfolio updated at the end of each stock market day; No. of available stocks, value of the stocks according to the latest market price etc. The system will generate different reports regarding user's transaction daily/biweekly/ monthly/yearly. A mechanism is also available in the system in case of any anomaly in user's transactions where the system will generate a report and mail it to the respected user about the anomaly. A reconciliation mechanism is also available in the system to resolve this anomaly.

Project Duration: 8 weeks (2 months)

Technology: Java 6.0 , Eclipse Helios , Apache Axis 1.0 , SOAP, JSP , Servlet , XML , Struts 2.0 , Spring 2.0 , Hibernate , Apache Tomcat 5.5 Web Container , Microsoft SQL Server 2005 etc.

Team Size: 5 persons

Team Member's Experience: 1 team member is 5 years experienced and the rest are 1 year experienced

Man Days: (Team size)*(No of weeks)*(No of working days) = 5*8*5 = 200

Overview: Agile Scrum methodology was implemented throughout the project. The total project was broken into 4 two (2) week sprints (10 working days per sprint). The total design, development and testing phase of the software were divided into 4 major modules as given below

i. Requirement gathering, analysis, use case preparation and database schema design
ii. File uploading, processing module and related User Interface (UI) module development and testing
iii. Portfolio, mailing , reconciliation and related UI module development and testing
iv. Report generation module and related UI module development and testing

Each sprint was assumed to cover (200/4) = 50 story points that is, 25 story points per week by the whole team and 1 story point per person in each day. First two weeks of the project (1st sprint) were spent in requirement gathering and requirement understanding, preparation of use cases and database schema for the system. As a result 200-50 = 150 man days were available for developing and testing the whole system before the final delivery. After the completion of sprint 2, 3 and 4 intermediary releases were made.

B. *Case Study 2*

Project: ABC Forum

Project Description: The aim of this project is to create an online forum. Users will register online and an admin will approve their registration. Users will be ranked into different classes based on their contribution to the forum. Users will be able to manage their own profile, customize it, create a new discussion topic on various topics (Interactive topic creation), comment on posts they are allowed to comment and send private mail to other users etc.

Project Duration: 12 weeks (3 months)

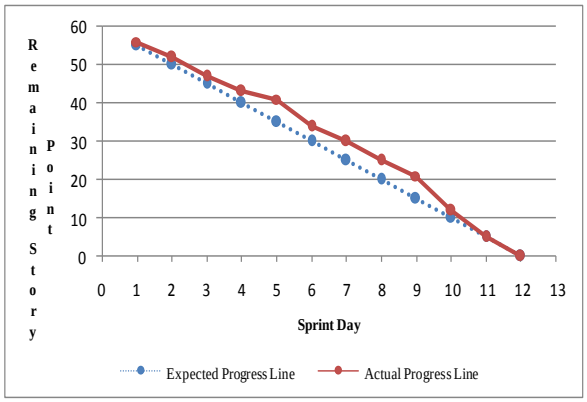


Fig.5 Burn Down Chart of a Sprint of Case Study 1

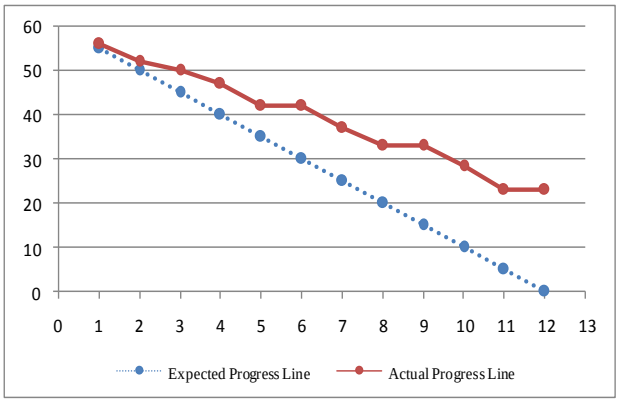


Fig.6 Burn Down Chart of a Sprint of Case Study 2

Technology: Java 6.0 , Eclipse Helios , Apache Axis 1.0 , SOAP, JSP , Servlet , XML , JForum , Apache Tomcat 5.5 Web Container , AJAX, Freemarker, MySQL etc.

Team Size: 5 persons

Team Member's Experience: 1 team member is 5 years experienced and the rest are beginners with their experience between 3 to 6 months.

Man Days: (Team size)*(No of weeks)*(No of working days) = 5*12*5 = 300

Overview: This project also used agile Scrum methodology but this time the software development teams experience is much less than the software development team of case study 1. The total project was divided into 6 two (2) weeks long sprints. The software development and testing phase was divided into 3 major modules:

i. Requirement collection, analysis , use case preparation, database schema design
ii. User Interface design, development and testing
iii. Operational functionality development and testing

Each sprint was considered to cover (300/6) = 50 story points, which means 1 story point per person in each day. After each sprint an intermediate release was made and a demo presentation was given based on the progress of the project.

V. Analysis of Burn Down Chart on Case Studies

Burn Down Chart is a popularly used tool in scrum to track down the progress of the project. It clearly indicates the current status and deviations from ideal planning. In addition, it can also anticipate the last standing of the project from any point of the chart. According to the Burn Down Chart of case study 1, though the target of the sprint was achieved at the end but the team was unable to keep pace with actual planning in the middle of the sprint. The team was well enough to overcome the deviations faced during the sprint.

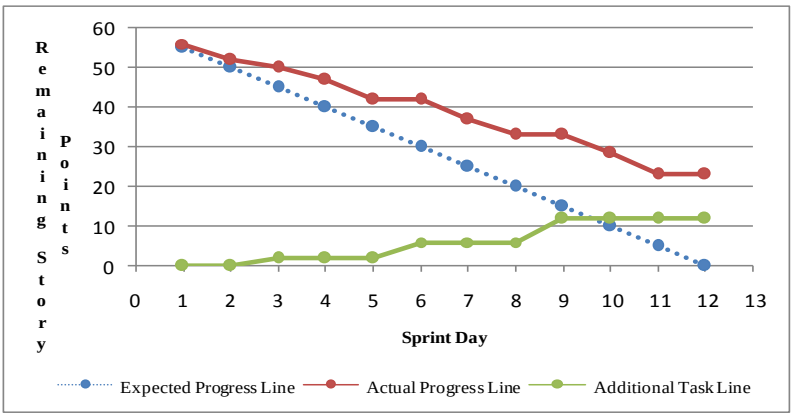
However, in case study 2, development team faced some problems and was unable to achieve the target of the sprint. Total story point estimated for the sprint was 50 and each member of the team (5 persons) was assigned to complete 1 story point a day. The progress digressed from the 3rd day of the sprint and the development team was unable to overcome it during the sprint. So at the end of the sprint they were unable to complete 23 story points.

The reasons for deviation from ideal planning vary with different teams in different circumstances. We identify three major causes for which the actual line deviates from ideal scenario. Firstly, inaccurate planning can initiate severe problem to sprint Burn Down Chart. This occurs when the product owner somehow fails to clearly convey the product needs to the Scrum team or the Scrum master wrongly estimate the story points for the development team. Secondly, members of the development team sometimes fail to complete the assigned task within time period due to lack of experience and inadequate knowledge on specific tasks. Finally, the changing requirements of the customer hamper the regular progress of the project which results in deviation in the Burn Down Chart. Whatever the reasons behind the problem are, software organizations always need to identify those specifically so that appropriate steps can be taken immediately.

VI. Alternative Burn Down Chart (Proposed)

A standard Burn Down Chart illustrates the current progress of the project along with differences (if any) from actual estimation. However, it does not represent the actual reasons for deviation. For example, in the Burn Down Chart of case study 2, it is undefined whether the development team faced any unavoidable circumstances or was unable to accomplish assigned tasks on day 6. One of the solutions proposed to overcome this problem is to use Burn Up Charts [5]. Burn Up Chart displays the amount of completed story points in each iteration along with the total estimated points for the project. Moreover, it is also necessary to know the answer whether the team was slow to complete a specific task or some new tasks were added to the project. One solution of this problem can be Phil Goodwin and Russ Rufer's Modified Burn Down Chart [5]. This is actually a bar graph where the height of each bar corresponds to the remaining story points. Additionally, new tasks are shown at the bottom of specific bars.

Here we are going to propose another form of Burn Down Chart that can easily display the addition of new tasks along with the status of the project. In this chart, a new line has been added that represents the addition of new tasks. The specific sprint day is represented in X-axis whereas new tasks are shown in Y-axis in term of story points.



The initial point of this line stands at (1, 0) which indicates no unplanned task appeared on the very first day. New tasks with story point 2 first came out on the 3rd day of the project. According to the sprint task board of Table 1, the team completed 4 story points on that day but only 2 points decreased from remaining story point as the team solved

unplanned task with 2 story points. Similarly, at the end of day 6, actual remaining story point stayed unchanged at 42 due to completion of new task with value 4. Since sprint day 3, our proposed new line for unplanned task remained stable at story point 2 until the sprint day 5 which implies no addition of unplanned task during these days. Finally, the largest amount of new task (story point of 6) added to the project on the 9th day of the sprint which is represented by a steep rise in unplanned task line from 6 story points to 12 on sprint day 9. Even though the team completed the highest amount of tasks (6 points) on that day, the actual progress line remained stable due to the accumulation of the new task. On the other hand, the stability of both actual progress line and unplanned task line on the last day of sprint clearly indicates that the team completed no tasks on that day.

This proposed graph mentioned in Fig. 7 can provide a clear idea regarding the progress of the project along with team performance. In addition, the reason for deviation in actual progress line can be easily identified by comparing the actual progress line with the unplanned task line. At the end of the sprint it is clearly visible from the graph that the sprint resulted in the unfinished task of 23 story points of which 12 were added as unplanned task. Moreover, the graph also implies that out of total 60 planned story points, the team was able complete (60-23) +12=49 story points which justifies the data of table-1.

TABLE I

Day	Member 1	Member 2	Member 3	Member 4	Member 5	Total Done/ Planned	Unplanned Task
1	1	1	.25	1	.75	4/5	0
2	.75	.25	2	.5	.5	4/5	0
3	1	.75	1	.5	.75	4/5	2
4	1	0	1	1	0	3/5	0
5	1	1.25	.75	1	1	5/5	0
6	.25	1	1	1	.75	4/5	4
7	1	1.25	1	1	.75	5/5	0
8	1	.25	.75	1	1	4/5	0
9	1	1.75	1	1.5	1.75	6/5	6
10	1	1	1	.5	1	4.5/5	0
11	1	1	1	1.5	1	5.5/5	0
12	0	0	0	0	0	0/5	0
Total	10	9.5	10.75	10.5	8.25	49/60	12

VII. CONCLUSIONS

Scrum is an immensely popular agile methodology among software organizations. A typical Burn Down Chart is unable to clearly indicate the reasons for deviation from ideal planning. In our proposed Burn Down Chart an additional unplanned task line along with the expected and actual progress line indicates the reason for digressing from the expected progress line. From this proposed graph, it is possible to find whether team's inexperience or addition of unplanned tasks results in deviation in the actual progress line. However, this Burn Down Chart still does not provide other reasons for deviation apart from the addition of new task. We will try to upgrade this chart in the future so that more reasons for deviations can be identified from a Burn Down Chart.

REFERENCES

[1] Paetsch, F., A. Eberlein, and F. Maurer, "Requirements Engineering and Agile Software Development", *in Proceedings of the Twelfth International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, IEEE Computer Society*, pp. 308, 2003.

[2] C. Larman and V. Basili, "Iterative and Incremental Development: A Brief History", • *IEEE Computer*, vol. 36(6): 47-56, June 2003.

[3] (September, 2012) Principles behind the Agile Manifesto 2001 [online]. Available : http://agilemanifesto.org/principles.html

[4] Rising, L.,Janoff, N.S.,"The Scrum software development process for small teams", *Software, IEEE*, Volume: 17 , Issue: 4, pp. 26 - 32, Jul/Aug 2000.

[5] Md. Junaid Arafeen1, Saugata Bose, "Improving Software Development Using Scrum Model by Analyzing Up and Down Movements on The Sprint Burn Down Chart: Proposition for Better Alternatives", *International Journal of Digital Content Technology and its Applications Volume 3, Number 3*, pp. 109-115, September 2009.

[6] (September, 2012) Why Is It Called Scrum? [online]. Available: http://www.scrumalliance.org/pages/what_is_scrum

[7] Sutherland, Jeffrey Victor; Schwaber, Ken, "Scrum Methodology, Business object design and implementation", *OOPSLA '95 workshop proceedings,* The University of Michigan. pp. 118, 1995.

[8] Schwaber Ken, Beedle Mike. *Agile software development with Scrum,* 1st Ed, Prentice Hall, 2002.
[9] (September, 2012) Jeff Sutherland's Scrum Handbook [online]. Available: jeffsutherland.com/scrumhandbook.pdf
[10] (August, 2012) Scrum advantages [online]. Available: http://www.scrummethodology.org/scrum-advantages.html
[11] (September, 2012) The Scrum Guide-Developed and sustained by Ken Schwaber and Jeff Sutherland October 2011 [online]. Available: http://www.scrum.org/Scrum-Guides
[12] (October, 2012) Scrum Is an Innovative Approach to Getting Work Done [online]. Available: http://www.scrumalliance.org/learn_about_scrum
[13] (October, 2012) Training for Scrum Task Board Use [online]. Available: http://www.mountaingoatsoftware.com/scrum/task-boards
[14] (October, 2012) Release Burn Down chart [online]. Available: http://www.mountaingoatsoftware.com/scrum/release-burndown

