$\lceil_{N-1}\rceil$ Effort Distribution in Scrum Framework for Complexity Drop

Saurabh Ranjan Srivastava[1], Girdhari Singh[2]
Department of Computer Science and Engineering
[1]Swami Keshvanand Institute of Technology Management & Gramothan, Jaipur
[2]Malviya National Institute of Technology, Jaipur
Email- [1]saurabh.ranjan.srivastava@gmail.com, [2]girdharisingh@rediffmail.com

Received 17 February 2016, received in revised form 6 March 2016, accepted 10 March 2016

Abstract: **Distribution and estimation of effort remains a decisive parameter in the success of every system engineering project. In case, when the system being developed is a software, this parameter becomes more critical as the number of factors to be included becomes outsized. Due to this, the precision and scheduling of effort estimation, heavily determines the failure or success of the software development projects [1], [2]. To tackle this challenge, various software development methodologies are adopted time to time, specific to the needs and priorities of software to be developed.**

Scrum is the most successfully practiced flavor of Agile development methodology in commercial software development, academics and other industries of multiple spheres.

In spite of all the flexibility and uniqueness available in scrum, effort distribution still remains a constant area of research. This paper proposes an empirical approach of effort distribution in scrum framework, based upon the number and expertise of team members involved in the project. Our effort distribution criterion distinctively prioritizes the number of tasks to be executed by every member of a scrum team.

Keywords: **Scrum, agile, framework, methodology, sprint, backlog, scrum-master.**

1. INTRODUCTION

Software effort estimation is a complex process that involves familiarity of multiple parameters responsible for the output of the whole software development project. Here effort estimation can be considered as the calculation of approximate results which are practically useful even if input data is inadequate or uncertain. As the development cost involved for a software project is directly proportional to the effort being involved, scheduling and distribution of effort for a software product is considered to be one of the most challenging and error–prone assignments in software industry. Various algorithmic approaches have been proposed from time to time as an optimal solution for estimation of effort. These approaches range from traditional waterfall [1] and spiral methodology [2] to computation intensive COCOMO framework [3].

But the ever increasing development parameters in a software project always demand a more flexible and dynamically growing development methodology.

The latest and most popular answer to this demand has been the Agile development methodology. Agile offers different technical flavors of development methodologies such as Extreme Programming, KANBAN, Crystal and Scrum [4], [5] suited to different scenarios as per user requirements. Among all these flavors, scrum has been found to be most capable of dynamically including the changing user requirements, system parameters as well as the project team dynamics [6], [7].

In scrum, details of every module are collected and developed in the form of user stories. These user stories are summarized to reports known as product backlogs according to their predefined priorities. During all this development, the distribution of effort according to the responsibilities among the team members can pay a crucial role in estimation of effort [7].

In this paper, we introduce a unique $\lceil_{N-1}\rceil$ approach for effort distribution, applicable in the initial stage of the scrum process. This technique assumes some activities to be fixed in every sprint of software development and distributes the remaining ones as per the dynamic requirements of the project.

2. OVERVIEW OF SCRUM METHODOLOGY FRAMEWORK

Scrum is a software project management framework from the Agile family of system development methodologies. Scrum is a simple but extremely efficient set of principles and practices that guides teams in delivering marketable products in short cycles with fast response. During this process Agile maintains large space for continual enhancement and speedy adaptation to changes in user requirements [6], [7].

Scrum can be used by anyone who has a complex project to manage, whether developing a system software, a desktop application, mobile app, e-commerce website or going to maintain old projects. The main objective of Scrum is to deliver the highest priority product backlogs in every sprint release [8].

The Scrum Process:

The Scrum process is composed of the following phases.

A **Sprint Planning meeting** is held with the development team, management, and the Product Owner. The Product Owner creates and prioritizes the Product Backlog and selects the features to be included in the next 30-day increment (called a Sprint) usually prioritized by highest business value and risk involved. Finally a Sprint Goal is established which presents a minimum, high-level success criteria for the Sprint and keeps the Scrum Team focused on the ultimate targets of the projects.

The development team figures out the tasks and resources required to deliver those features. They also define a reasonable number of features to be included in the next Sprint. Once this set of features has been identified, no re-prioritization takes place during the subsequent 30-day Sprint in which features are designed, implemented and tested on daily basis [8].

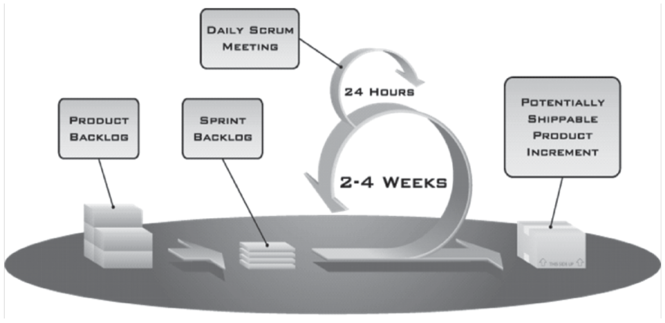

Figure – 1: The Scrum Framework

15-minute short *Scrum Meetings* are held daily by the scrum development team where tasks and blocks are written down. Other people like product manager and managers may attend the Scrum Meeting, but only the team members and the Scrum Master can speak. Each team member answers the following questions:

- What have you done since the last Scrum?
- What will you do between now and the next Scrum?
- What roadblock is preventing you from achieving your target?

At the end of one sprint increment, a *Sprint Review* meeting conducted by the Scrum-Master, takes place to review progress, demonstrate features to the customer, management, users and the Product Owner and review the project from a technical perspective. The Product Owner and other interested stakeholders attend the meeting. The latest version of the product is demonstrated in which the functions, design, strength, weaknesses, and trouble spots are shared with the Product Owner [9], [10].

Instead of presentations, the major focus remains on showcasing the finished product itself till date. The cycle is repeated till the finish of the software development with a Sprint Planning meeting taking place to decide the features for the next Sprint.

Scrum Roles:

In Scrum framework, there are three possible roles for team members namely Product Owner, Scrum Master and Scrum Members [7], [8].

Product Owner is responsible for creating and refreshing a prioritized list of project features known as the Product Backlog, selecting what will be included in the next iteration/Sprint, and reviewing the system with other stakeholders at the end of the Sprint.

Scrum allows the possibility of one and only one Product Owner who is responsible for the quality and value of the project.

Scrum Master implements and reinforces the product iteration targets, scrum values and practices [6]. He conducts the daily scrum meeting and the sprint review demonstration, marks the progress achieved, removes roadblocks in development, and provides resources. The Scrum Master himself is a developer and also participates in product development.

Scrum Development Team is the group of developers**,** designers and other members of the project team who are committed to achieving the sprint goal. Scrum development team members have full authority to do all necessary actions to achieve the sprint goals. The general size of a scrum development team is seven, plus or minus two [6], [7], [8].

Scrum Artifacts:

Scrum teams are responsible for production of following 3 main artifacts, the Product Backlog, the Sprint Backlog, and the Sprint Burndown chart. All of these are flexibly accessible and purposely visible to the scrum team.

Product Backlog is an evolving, prioritized (ordered 1, 2, 3, ...) file of customer-centric of business and technical functionalities. The target of the scrum team remains to develop the product backlog into a system while fixing the defects and incorporating the upcoming changes. The Product Backlog stores a unique identifier for each feature requirement, with parameters such as category (feature, enhancement, defect), the status, the priority, and the estimate for the feature. Generally the product backlog is maintained in a spreadsheet like format and is evolved over the lifetime of the product.

Sprint Backlog is the list of all business and technology features, enhancements, and flaws scheduled to be covered for the current iteration known as a Sprint.

The requirements taken from the product backlog are broken down into tasks and maintained in the form of a short task description, who originated the task, who owns the task, the status and the number of hours remaining to complete the task. The Sprint Backlog is updated each day by the scrum-master.

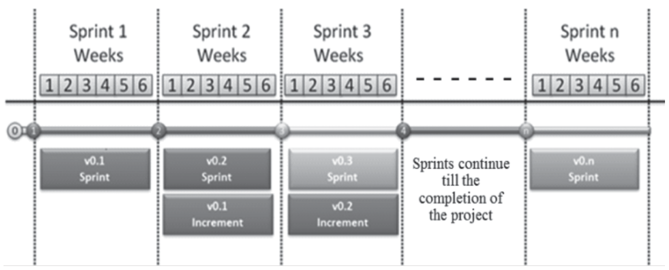

Figure – 2: The 6-week sprints of scrum framework [9]

Sprint Burndown chart is the document utilized to estimate the eventual conclusion of an estimated backlog of work. The main use of burndown charts is made to estimate the quantity and quality of the work left.

It is also useful to estimate the number of hours remaining to complete by mapping the sprint backlog features and displaying them for the team.

3. SCRUM EFFORT DISTRIBUTION

The scrum Development Team is "cross-functional" in nature which implies that it includes all the expertise necessary to deliver the potentially deliverable product after each Sprint. The scrum team is also a "self-organizing" or self-managing body with a very high degree of autonomy and accountability. The Team decides how many items (from the set offered by the Product Owner) to build in a Sprint [9], [10], and how best to accomplish that goal.

The special feature of a scrum team is that there are no fixed specialist titles in a group that adopts Scrum; there is no business analyst, no DBA, no architect, no team lead, no interaction / UX designer, no programmer. They work together during each Sprint in whatever way is appropriate to achieve the goal they have set for themselves.

Since there are only team members, the Team is not only cross-functional but also demonstrates multi-learning ability. Each person certainly has special strengths, but also continues to learn other specialties. Each person will have primary, secondary and even tertiary skills, and is meant to "go to where the work is"; individuals take on tasks in less familiar areas to help complete an item. For example, a person whose primary skill is interaction design could have a secondary skill in automated testing; someone with primary skill in technical writing might also help with analysis and programming.

4. PROPOSED SCHEME FOR EFFORT DISTRIBUTION

Viewing this nature of scrum teams, here we propose an empirical approach of effort distribution. This approach divides the required effort of team members in a prioritized manner at the outset of project initialization by pre-assuming some factors.

In this approach we consider the whole software development lifecycle to be composed of following major stages: Analysis, Design (Architectural and Detailed), implementation (code generation), testing (unit and integrated), deployment and maintenance.

Now here we assume that analysis and testing are inherently group practices that demand complete team effort. This implies that whole team will go on for analysis before starting the project. Similarly generation of test cases, unit and integrated testing require efforts of complete team.

After setting apart these 2 activities that consume major time and effort, we are left with designing, coding and deployment-service activities to be scheduled.

For distinct prioritized effort distribution among these activities, we implement the $\lceil N - 1 \rceil$ criterion where **N** is the number of available team members.

By using this criterion, every team member can have at most $\lceil N - 1 \rceil$ number of tasks without any clash of responsibilities. Stating this mathematically we have the equation

$$\mathbf{T} = \lceil \mathbf{N - 1} \rceil \qquad \textbf{- Equation-1}$$

This equation represents the ceiling (maximum limit) of the number of tasks to be allotted to an individual team member. A sample distribution of efforts for a scrum team of 4 team members is given in the table ahead for demonstration.

Team Member-1	Team Member-2	Team Member-3	Team Member-4
ARD		ARD	ARD
	DTD	DTD	DTD
CDG	CDG	CDG	
DS	DS		DS

Table – 1: Tentative Effort Distribution for a Scrum Team

Here following conventions have been used:

ARD	=	Architectural Design
DTD	=	Detailed Design
CDG	=	Code Generation
DS	=	Deployment Services

This table presents a tentative distribution of effort among srcum team members without any clash of responsibilities. Further, responsibilities can be specifically divided among team members by dedicatedly tagging tasks relevant to their expertise and / or interest given as follows.

ARD	Architectural Designing (Modular Decomposition / UML Diagrams / ERD Designing)
DTD	Detailed Designing (Algorithm Design / Logic Development / UI Designing)
CDG	Source code generation (Login / Display Panel, Dashboard / Database Connectivity / Reports)
DS	Deployment-Services (Database Backup / Database Migration / Report Generation / Reboot Initialization)

Table – 2: Possible Classifications for Tasks Assigned in Table 1

The usual size of a Team in Scrum is 7 ± 2 people [6], [8]. The Team including people with skills in analysis, development, testing, interface design, database design, architecture, documentation, etc can appropriately accommodate $\lceil N - 1 \rceil$ this criterion for effort distribution among team members.

	14-Jul	15-Jul	16-Jul	17-Jul	18-Jul	21-Jul	22-Jul	23-Jul	24-Jul	25-Jul
Not Started	229	222	197	93	93	93	93	93	93	93
In Progress	55	76	106	139	139	139	139	139	139	139
Completed	0	10	41	122	122	122	122	122	122	122
Total Backlog	284	308	344	354	354	354	354	354	354	354

Table-3: Backlog Coverage for a 2 week sprint by standard scrum

	14-Jul	15-Jul	16-Jul	17-Jul	18-Jul	21-Jul	22-Jul	23-Jul	24-Jul	25-Jul
Not Started	229	222	197	93	93	93	93	93	93	93
In Progress	55	76	106	139	115	113	108	107	97	90
Completed	0	10	41	122	146	148	153	154	164	171
Total Backlog	284	308	344	354	354	354	354	354	354	354

Table-4: Backlog Coverage for a 2 week sprint by $\Gamma_{N-1}\Gamma$ scrum

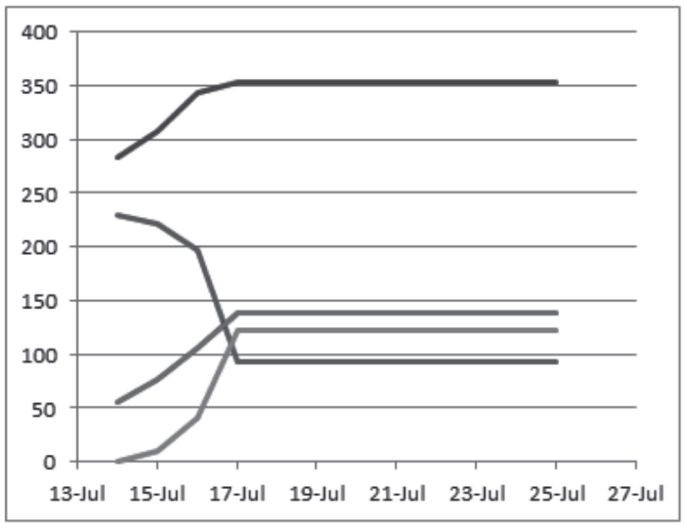

Figure – 3a: Standard Effort Distribution

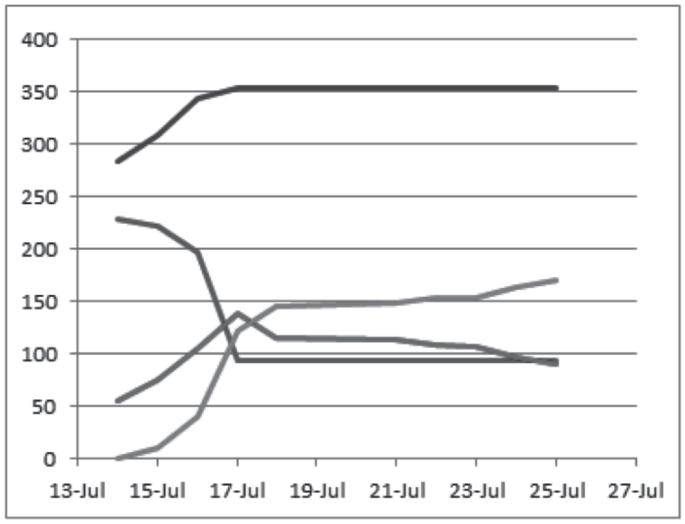

Figure – 3b: [N-1] Effort Distribution

Tasks Not Started **Tasks in Progress** **Tasks Completed** **Tasks Total Backlog**

Figure – 3: Effort Distribution and Backlog Coverage in Standard vs [N-1] Schemes for Scrum

5. COMPARISONS OF THE 2 SCHEMES

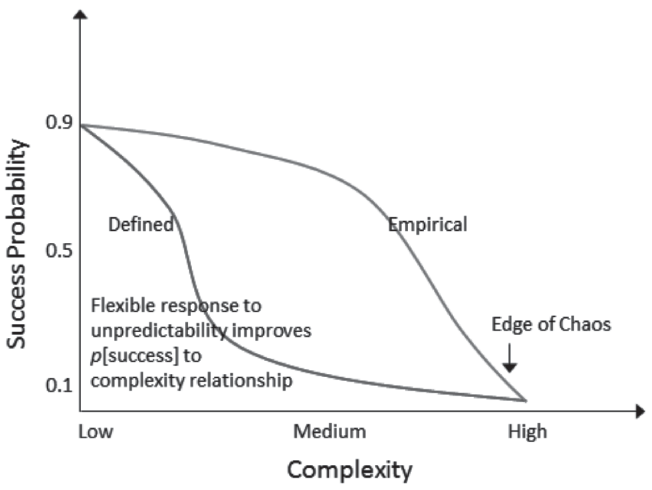

Figure 4: Complexity / Success Graph for Usual Effort Distribution in Scrum Framework

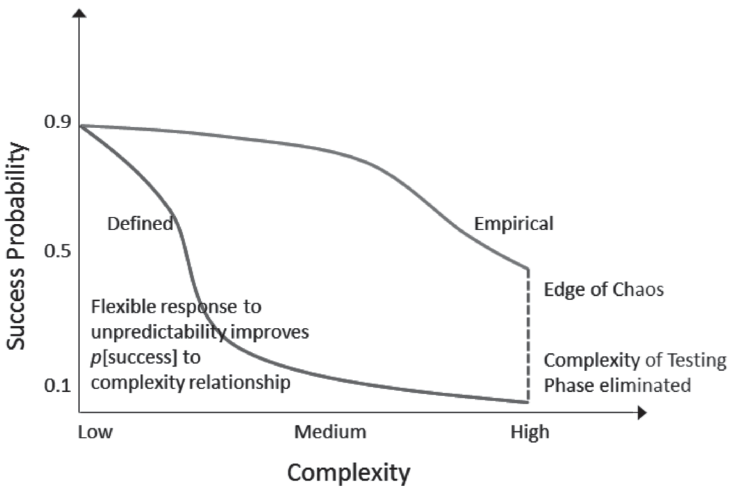

Figure 5: Complexity / Success Graph for (N-1) Effort Distribution in Scrum Framework

As already discussed, scrum framework consumes 7 ± 2 persons as team members. To demonstrate the contrast of our scheme, here we discuss an example of backlog coverage for a 2 week sprint. Table – 3 contains the task coverage data of the standard scrum practice, while table – 4 has the similar data for the proposed $\ulcorner N-1 \urcorner$ scheme in scrum framework.

Now as shown in Table – 3 and Table – 4, the tasks not started were 229 and tasks in progress were 55. After 3 more days of progress, these numbers settled to 93 and 139 in both schemes. Where the standard scrum carried on with these numbers till the end of the sprint, the $\ulcorner N-1 \urcorner$ scheme showed a constant coverage of tasks and their transfer to the burndown list. This continuous coverage of tasks, also known as the velocity, is crucial for the success of the project in long term.

As the sprints proceed with time, this scheme can save considerable amount of effort and time of the scrum development team.

In turn, a wide gap between the empirical success rate and the defined success rate can be expected at the end of the sprint. As the velocity of the sprint is saved from slowing down, the coverage of tasks will not be at a threshold edge.

At the end of the sprint, in the sprint reviews meeting the Scrum Team and the Scrum Master together can work on to maintain and update the velocity of the scrum process. For this they can work upon following questions, what went well during the sprint, what didn't, and what improvements could be made in the next sprint, and so on.

The comparison of estimated effort distribution in a scrum team is given in Figure-3 and Figure-4. The immediate cutoff of complexity in Figure-4 represents the implication of the **N – 1** criterion during analysis and testing phases. Hence the overall complexity graph can be expected to remain same for each upcoming sprint as whole team will resolve analysis and testing complexities at the outset.

6. COMPARISON & FUTURE WORK

As we have already discussed, a scrum development team is a group of project team members working in multiple sprints. Now if the jobs to be fetched up by the members of the scrum development team are according to their caliber and also provide them a chance to expand their skillset, then a significant drop in the struggle of task allocation can be observed in the productivity of the existing Scrum framework.

Figure-3 represents the usual allocation of jobs among the scrum development team such as analysis, design, coding, testing, deployment and maintenance, etc. The success probability of the sprint decreases by the rise in complexity. But we can expect an empirical reduction in the success probability drop for the $\ulcorner N-1 \urcorner$ effort distribution scheme for a general scrum development team of 7 ± 2 members.

The major cause for this drop is the pre-allocation of duties regarding effort distribution. This pre-allocation relative to Table-1 and Table-2 will not reduce the clash of responsibilities among the scrum team, but also facilitate collaboration at various levels. This will be eventually improve utilization of manpower as well as give chances to team members to expand their skillset.

In future, we expect to improve the precision of this $\ulcorner N-1 \urcorner$ criterion for a scrum team larger than the size of 7 ± 2 members and more deeper level of software project tasks. The prime focus of the future research will be to empirically quantify the allocation of effort distribution and minimize the team size for commercial software environments.

REFERENCES

[1] Bassil Youssef; **A Simulation Model for the Waterfall Software Development Life Cycle**; 2012; International Journal of Engineering & Technology (iJET) , ISSN: 2049-3444, Volume 2 , Issue 5,2012; May 2012

[2] Boehm Barry; **A Spiral Model for Software Development and Enhancement**; 1988; *Computer – ACM Digital Library;* Volume 21, Issue 5; Page 61-72, May 1988

[3] Boehm Barry, Clark Bradford, Horowitz Ellis, Westland Chris, Madachy Ray, Selby Richard; **Cost models for future software life cycle processes: COCOMO 2.0**; 1995; Annals of Software Engineering; December 1995; Volume 1, Issue 1, Page 57-94

[4] Boehm Barry; **Get Ready for Agile Methods, with Care**; 2002; IEEE Computer, Volume 35, Issue 1, Jan 2002, Page 64-69

[5] Fowler Martin, Highsmith Jim; **The Agile Manifesto**; August 2001; http://andrey.hristov.com/fht-stuttgart/The_Agile_Manifesto_SDMagazine.pdf; 2001.

[6] Greening Daniel R.; **Enterprise Scrum: Scaling Scrum to the Executive Level; 2010**; 43rd Hawaii International Conference on System Sciences (HICSS); 5-8 January 2010; Honolulu; ISSN 1530-1605; Page 1 - 10

[7] Schwaber Ken, Beedle Mike; **Agile Software Development with Scrum**; Prentice Hall, 2001; Page 89-94

[8] Cockburn Alistair; **What the Agile Toolbox Contains**; 2004; CrossTalk Magazine - The Journal of Defence Software Engineering; November 2004; Pages 4 -7

[9] Heys Bill; **Branching for Scrum**; 2011; MSDN Blogs; http://blogs.msdn.com/b/billheys/archive/2011/01/18/branching-for-scrum.aspx; 18 Jan 2011

[10] Asproni Giovanni; **An Introduction to Scrum**; 2006; Software Developer's Journal; giovanniasproni.com; June 2006

◆ ◆ ◆