I.J. Information Technology and Computer Science, **2018, 12, 41-48**
Published Online December 2018 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2018.12.06

ScrumFall: A Hybrid Software Process Model

Md Shamsur Rahim, AZM Ehtesham Chowdhury, Dip Nandi, Mashiour Rahman
American International University-Bangladesh/Computer Science, Dhaka, 1229, Bangladesh
E-mail: {shamsur, ehtesham, dip.nandi, mashiour}@aiub.edu

Shahadatul Hakim
Enosis Solutions, Dhaka, 1212, Bangladesh
E-mail: shahadat@gmail.com

Received: 04 August 2018; Accepted: 22 September 2018; Published: 08 December 2018

***Abstract*—**Every software project is unique in its own way. As a consequence, a single software process model cannot be suitable for all types of projects. In the real world, practitioners face different difficulties with the existing process models during development. Still, they cope up with the challenges by tailoring the software development lifecycle according to their needs. Most of these custom-tailored practices are kept inside the walls of the organizations. However, sharing these proven and tested practices as well as acquired knowledge and experience would be highly beneficial for other practitioners as well as researchers. So in this paper, we have presented a software process model which contains the characteristics of both Scrum and Waterfall model and named it "ScrumFall". This model has been practicing in an Anonymous Software Development Company, Bangladesh to solve the shortcomings of Scrum and Waterfall models. Moreover, we have analyzed the performance and suitability for applying this process model. The result shows that this process model is highly effective for the certain projects.

***Index Terms*—**Agile, Hybrid Process Model, Software Development Life Cycle, Software Process Model.

I. Introduction

Software Process Model, also referred as Software Development Life Cycle (SDLC) describes the sequence of required phases for the entire lifetime of a software product from a perspective of management. This model covers everything from the inception of a project by communicating with clients until the phase-out of the software product. The goal of following a process model is to split the software development activities into distinguishable, unambiguous phases with the intent of better planning and management to achieve economic success. Although there were no emerging process model or framework until early 1960's, then from the late 1960's to present, many models have been proposed and used in the industry to develop software in an effective manner [1]. Some of the commonly used models are: Waterfall Model, Incremental Development, Iterative and incremental development, Spiral Development, Rapid Application Development, Prototyping, Scrum, Kanban, Dynamic System Development Method (DSDM) and so on. All these models have their own strength and weakness and based on their nature, these models can be divided broadly into two categories: Plan driven and Agile processes.

In the case of a plan-driven process, all the activities for the entire lifetime of the product are planned and progress is calculated based on the plan [2]. As a result, it is difficult to adopt any changes in the middle of development. A plan driven model is suitable when the product and the team are large, the product is highly critical and hard to scale down and the development environment is stable. In a plan driven model, experienced personnel is required only at the beginning of the project and success is achieved through structure and order. Despite that, if the development environment is dynamic that means changes in requirements are occurring frequently, then it is expensive [3]. Waterfall Model, Incremental Development, Iterative and incremental development, Spiral Development, Rapid Application Development and Prototyping are the examples of the popular plan-driven process models.

On the other hand, it is easier to change in plans as planning is incremental in Agile methodologies. As a consequence, changes in user's requirements can be easily adopted and reflected in the software. An agile process is suitable when the product and team are small in size, the development environment is dynamic in nature. However, experienced personnel is required throughout the project and success is achieved through freedom and chaos. Scrum, Kanban and Dynamic System Development Method (DSDM) are the examples of the Agile process. Furthermore, issue starvation [4] is also a great concern in agile.

In practice, most of the software has its own unique characteristics and shares common characteristics such as feature, functionality, complexity compared with the previously developed software. These unique characteristics of a software make it difficult for the practitioners to plan and manage effectively using their experience. However, based on these common characteristics, the practitioner can choose a suitable process model to develop the product.

Due to the nature of the software to be built, sometimes

it is difficult to choose one between agile and plan-driven process. For instance, the environment for developing a software can be dynamic at the beginning but after passing few phases it can be stable. In this scenario, it is difficult to choose the right process model for developing the software. Although Pressman [3] stated that, in practice, the process models include both agile and plan-driven elements however very few process models have described how to do it.

In this paper, we have presented a hybrid software process model which we have named as "ScrumFall". This process model includes the elements of plan-driven and agile processes which facilitate to cope up with the dynamic as well as stable nature at the different phases of SDLC. The main contributions of this paper are:

- Presentation of a practical hybrid software process model combining elements of plan-driven and agile processes.
- Possible characteristics of a software project that is suitable to apply this model.
- Verification of the effectiveness of this model using real-world data.

The rest of this paper is organized into following four sections: the strength and weakness of the previously proposed models in section II, description of the proposed model and characteristics of the software suitable for the model in section III, discussion on the effectiveness of the model from analyzing real-world data in section IV and conclusion is discussed in section V.

II. Related Works

The software process models facilitate an abstract representation that carries out proper description of the process from a particular perspective. Several software process models are proposed in the past decade. In this section, we provide the insights and advantages and disadvantages regarding efficient process management. Here we have mainly included the process models that are commonly used in software industries [5], according to a recent survey [6].

A. Plan-Driven Development

Plan-driven development [2] is a more formal specific approach for developing systems. This model tries to plan and anticipate user's requirements that might be wanted in the end product. In plan-driven development, specific phases are followed in a sequential manner. Some of the most widely used plan-driven development models are briefly discussed below:

a) Waterfall

The waterfall model is considered as the classic Software Development Life Cycle (SDLC) model. It is the first process model that follows sequential stages of software development [7]. It is also the first plan driven model that emphasize on early planning, requirement analysis, the design of the process. It also offers the quality control and stability of software as there is less scope of adaption to new requirements after the deployment. So, ensuring flaws free design and requirement analysis are the major concern in this model. It reduces the overhead of planning and process output is consistent with the requirements. This model works fine on big and weak teams. However, the model lacks flexibility as the adaption to requirement change may cost time and more money. For small teams and projects, the overhead of administration increases using this process model. The elapsed time increases in case of testing failures. Deployment time is stretched long as no scope for deployment of feature versions.

b) Incremental Development

As the waterfall model lacks flexibility, so several new process development models have been established. One of these models is incremental development model [8]. Unlike waterfall, it allows to break down the software delivery into increments. The user gives the priority of the requirements. Then highest prioritized requirements are included in earlier increments. Each increment provides the functionality through which customer value can be delivered. The risk of project failure gets reduced as starting increments are a prototype for later increments. Yet this model lacks flexibility regarding activity switching.

c) Iterative development

Iterative model [9] is another flexible software development model. Like the incremental model, iterative model divides a product into smaller products. This model aids earlier demonstration of the project and obtains responses from system users. At each step, when the mini products are released, another mini product can be going into production. The iterative model facilitates effective risk analysis and management as high prioritized products are done earlier. However, this model does not suitable for smaller projects. Continuing work by following this model, the management of the project can be going to be complex as more resources and attention are required. Furthermore, as all requirements are not analyzed at the beginning of the entire life cycle, so architecture or design issues may arise.

d) Spiral Development Model

The spiral model is more likely to incremental model [10]. Spiral model has four phases: Planning, Risk Analysis, Engineering & Evaluation. This model is more concerned about risk analysis. Previously, spiral model vastly adopted by software industries. This model accommodates the changes in requirements as early involvement of the user in the system development is ensured. This model aids viewing of the system very early. The spiral model is suitable when risk and cost evaluation is important, and requirements are complex. Like iterative, the management is complex as well as the process itself to maintain and the spiral can go indefinitely period of time. This model is not significant to use for small projects and project success highly

dependent on risk analysis. So, it is costly to some extent.

e) Prototype Model

The prototype model [11] is one of the traditional software process models. This model is followed when requirements are not clear entirely which serves as a mechanism for identifying software requirements. It increases user involvement at the earlier phases of a project even before implementation. Through prototyping, the functionalities of a software are modeled that may not contain the exact logic of the of the original software which results in better understanding and quick feedback from the customers. Despite the significant advantages, the prototype model suffers from disadvantages like insufficient requirement analysis, confusion among the customers between prototypes and the actual system.

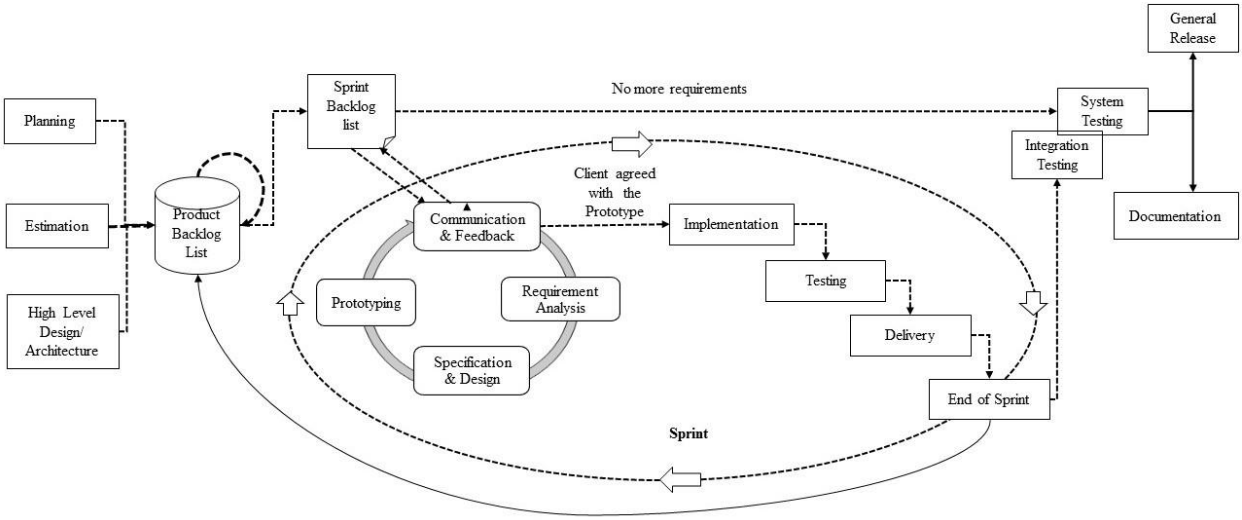

Fig.1. ScrumFall Model.

Table 1. Comparison of the advantages and disadvantages of the plan-driven and agile processes

	Software Process Model	Advantages	Disadvantages
Plan Driven	• Waterfall • Incremental Development • Iterative development • Spiral Development • Prototype Model • Rapid Application Development	• Suitable for large systems and teams. • Handles highly critical systems effectively. • Appropriate for stable development environment. • Require experienced personnel at the beginning. • Success achieved through structure and order.	• Longer length in each iteration or increment. • Cannot accommodate changes any time. • Lack of user involvement throughout the life cycle of the product. • Costly for the dynamic development environment. • Assume that, future changes will not occur.
Agile	• Scrum model • Extreme Programming (XP) • Dynamic System Development Method • Kanban • Feature Driven Development	• Suitable for small to medium systems and teams. • Can accommodate changes at any time. • Effective for the dynamic development environment. • Required expert agile personnel throughout the life cycle. • Success achieved through freedom and chaos.	• Not suitable for large systems (except FDD). • Shorter length in each iteration. • Can accommodate changes at any time. • Costly for the stable development environment. • Assume that, frequent future changes will occur.

f) Rapid Application Development

Rapid Application Development (RAD) model follows the iterative development and prototyping model [12]. Functional modules are developed concurrently as prototypes and later on, these modules are integrated to complete the software. This model accommodates the change in requirements, progress can be tracked and reduces development time with fewer people as well as increases reusability of prototypes. Still, this model is dependent on highly skilled personnel to identify and analyze the business requirements and development of software. It also demands client engagement throughout the different phases of the model.

B. Agile Models

Agile model is a subset of iterative and evolutionary methods where the key difference lies in the length of

each iteration [13]. In agile methods, the length of each iteration varies from one to four weeks. This results in small incremental releases based on previously built functionalities. Agile methods are considered as lightweight and people-oriented process model rather than plan based. Agile proponents believe that traditional software development models are so heavyweight and rigid that more customer involvement is required. The agile model has several variants that are briefly discussed below:

a) Scrum model

It is a popular variation of the agile methodology where the software development processes are partitioned into three distinct phases: Pre-game phase, Game/ Development phase and Post-game phase [14]. The pre-game phase is comprised of two sub-phases: Planning and High-Level Design/Architecture. The Game phase also referred as development phase is considered as a "Black Box" where uncertainty is expected, and the software is developed in several sprints. Sprints cycle iteratively to develop or enhance functionalities to produce new increments. Generally, each sprint is lengthened from one week to four weeks which includes the traditional software development phases: Requirements, Analysis, Design, Evolution, and Delivery.

b) Extreme Programming (XP)

The extreme programming model is originated from the disadvantages of the traditional process model [15]. This process model addresses the fundamental risks in software development and it is formed based on common sense principles and understandable practices. XP has five phases: Exploration, Planning, Iterations to Release, Productionizing, Maintenance, and Death [16]. XP model requires the centralized development practice and are only suitable for the XP-able projects.

c) Dynamic Systems Development Method

DSDM has been developed from the experience gained by a large consortium of vendor and user organizations [17]. The main focus of DSDM is to deliver immediate business needs on time. It focuses on frequent product delivery. It also facilitates the reversibility of changes at any time. DSDM has the potential risks of lack of user involvement or too much user involvement, excessive time on decision making, development of irreversible implements. The system may fail as testing is not integrated throughout the life cycle.

d) Feature Driven Development

FDD is an agile software development process which uses short iteration model [17]. In FDD, the feature is an important aspect. This model was created to scale down the larger projects and teams easily by combining key advantages of other well-known agile approaches and industry recognized best practices. FDD is a feature-centric model where a feature is a small function expressed in client-valued terms. Each feature needs to be small enough so that it will take no more than two weeks to complete. The development process of FDD is divided into five processes: Develop an overall model, build a features list, plan by feature, design by feature and build by feature. This model is mainly suitable for large systems and pays less attention to the initial design.

e) Water-Scrum-Fall Model

Water-Scrum-Fall model is a hybrid Agile method [19] where the software development activities are divided into three distinct phases: Water, Scrum, and Fall. Water represents the upfront activities such as defining requirements and planning. Scrum is the middle phase of the process where the software is actually built. On the other hand, Fall controls the release of software release frequency by forming gates. Although the Water-Scrum-Fall model suggests limiting the time spent on upfront activities, however, there is no direction regarding how to deal with the mixed nature of environment during development.

The comparative study of the advantages and disadvantages of both plan-driven and agile processes are summarized in Table 1 based on our findings and findings of [1, 20].

From the discussion in Table 1, we identify that among the existing models some are suitable for the stable environment and some are for dynamic. The length of each iteration is lengthier in some models and shorter in other models. However, in practical, for a specific product, the development environment can be dynamic in the initial phases of development life cycle of the product however the development environment can be stable in the later phases. To the best of our knowledge, no process model has been observed that can accommodate both the stable and dynamic environment. We have observed the research gap for a software process model that can incorporate both the stable and dynamic environment as well as the different size of systems and teams.

III. Proposed Model

The proposed ScrumFall model can be considered as a subset of Scrum and Waterfall model and combination of other industry best practices. However, this model emphasizes the agility in a large project and how to cope up with dynamic as well as a stable development environment. The presented model in this paper has the following group of phases: Pre-game Phase, Game/ development Phase and Post-game phase which are similar to the Scrum model. In addition, each of these phases can be divided into sub-phases and their steps. The description for each phase, sub-phases and their steps are discussed in this section. Furthermore, Fig. 1 demonstrates the different steps and activities of the proposed model.

A. Pre-game Phase

a) Planning & Exploration

After the initial communication and agreement with the client, the actual work of developing a software starts

with Planning. In this sub-phase, the definition of the new product is defined from currently known requirements along with the estimation and priority for each task. High level design or the architecture of the system is also developed in this phase. Moreover, the software quality attributes [21] and their scale or unit of measurement are identified and quantified. According to Pressman [7], quality attributes play a vital role in decision making of the architecture and design of the system. If the system being developed is new, then this phase consists broad analysis and sometimes training may be required for the personnel. On the other hand, if the system being developed is the existing one and it is being enhanced then this phase consists of limited analysis. The general steps in the Planning phase are as follows:

- Communication & Requirement gathering.
- Requirement Analysis & Specification.
- Identify tools, technologies and personnel required for the system.
- Device test strategies and test plan.
- Identify and quantify software quality attributes and their measuring scale or unit.
- Training on tools, technology or practices if required.
- Product backlog creation with requirements at hand.
- Cost and effort estimation for each task in the product backlog list.

b) High Level Design/ Architecture

At this sub-phase, the initial architecture of the system and high-level design for implementing the requirements from the product backlog list are constructed. Besides, identification of changes and refinement of architecture to implement new requirements, domain analysis, risk analysis are also included in this sub-phase.

B. Development Phase

The main difference between the Scrum model and proposed model is the development phase. In this proposed model, development work is done in a mix of iterative and sequential manner. The development team along with the other stakeholders determine whether the development time, quality or functionalities of the product are met as the requirements and at the end of iterations the product is delivered. As like the Scrum model, this proposed model also consists following macro processes [14]:

- Review release plans in the team meeting.
- Conformant of the distribution, review and adjustment of the standards.
- Several Sprints until product delivery.

Sprint consists of development activities which are performed both in the iterative and sequential manner in the ScrumFall model. The earlier activities like communication and design, requirement analysis, specification & design and prototyping are performed in an iterative manner in the Sprint and this iteration continues until the customer/client comes to an agreement with the developer. When the customer/client agrees with the developer, the rest of development activities like implementation, testing, delivery of the new increment is performed in a sequential manner like Waterfall model. The general steps in Sprint are as follows:

- Sprint Planning Meeting to set Sprint backlog list.
- Daily scrum meeting to track progress and resolve problems on daily basis.
- Peer-review to reduce risk and uncertainty.
- Update effort and cost based on new changes.
- Add/remove enhancement/fix to increase productivity and decrease the possibility of delay in delivery.
- Integration of new increment at the end of Sprint.
- Sprint review meeting at the end of each Sprint to discuss the results and find out the scope of improvement.

In the Scrum model, the length of each Sprint is one to four weeks. On the other hand, in the ScrumFall model, the length of each Sprint is flexible depending on the nature of the project, ideally it can be between 3 to 6 months.

C. Post-game Phase

The post-game phase at the end of each Sprint or when there are no more requirements to be implemented for the next release. In this phase, after each Sprint, integration testing is performed, and the product is prepared for general distribution. After performing system testing and updating/ modifying documentation the product is ready for general release.

IV. Result and Discussion

Although the presented model in this paper has similarities with Scrum model however it can be distinguishable easily for the clear and comprehensive combination of the elements from plan-driven and agile processes. The major difference between the proposed model and existing Scrum model is the nature of the Sprint. In the Scrum model, development activities are performed iteratively, and the length of each iteration is one to four weeks [14] where changes are always welcome. However, in the practical world, it is not possible to accommodate continuous changes in requirements in the later activities of a Sprint. Besides, the iteration length for each Sprint is too small to support large, complicated requirements. Keeping in mind these limitations of the Scrum model, the ScrumFall model has adapted hybrid approach in Sprint and provides flexibility in the length of each Sprint to support large and complex requirements.

The proposed process model is currently using the Anonymous Company, an offshore based software firm in Bangladesh. The team develops a framework which is

used by another team to develop the complete software. Each feature is large and complicated enough to develop. Due to the dependency and nature of the features, short development time is not suitable for this team. Previously, the team at Anonymous Company was using Waterfall model as their SDLC. However, they face problems such as for accommodate changes in requirements, delay in features delivery and less number of features to be delivered within the given time. Due to the large size and complexity of the project, the distributed team around the world and requirement of long iteration length, no agile methods were suitable for them. So, they start following a hybrid process model and self-organizing team model [22]. As a result, facts that are observed by the development team in Anonymous Company after choosing ScrumFall model as their SDLC are captured through an interview (Appendix A):

- Improvement in delivery time by lessening delay in feature delivery.
- Increase in the number of features to be delivered.
- The increase in the number of fixing bugs.
- Accommodate changes in requirements without risking a delay in delivery.
- Moderate user involvement when it should be needed.

Fig. 2 represents that, during the delivery of version 5.0 and 5.5, the development team followed Waterfall model. As a consequence, they suffered from 4 weeks' delay in each delivery. On the other hand, when the development team started following ScrumFall model for the delivery of version 6.0, they were late for only two weeks. Moreover, in the subsequent releases (6.5, 7.0, 7.5), there was no delay. This indicates that, this model helps the development team to stay always on schedule.

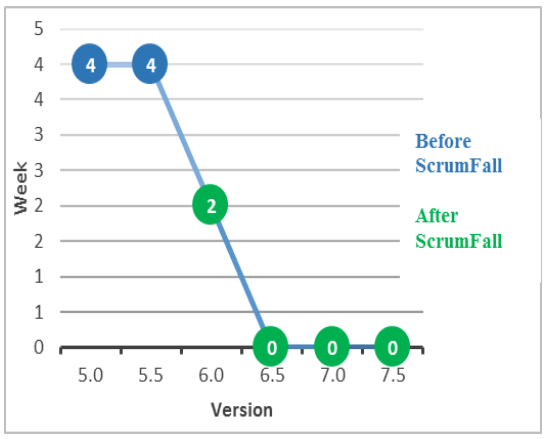

Fig.2. Comparison of Delay before and after following ScrumFall model.

The chart from Figure 3 indicates the number of enhancements delivered in each release before and after following ScrumFall model. During the release of the version 5.0 and 5.5 when the Waterfall model was followed, the number of enhancements delivered were 7 and 11 respectively. Although at the beginning of practicing ScrumFall model, the number of delivered enhancements was lowest (6), but it rocketed up to 23 for the later releases of the version 6.5 and 7.0. In the case of version 7.5, we can observe that the number of enhancements delivered is reduced to 17 whereas it was 23 for version 7.0. This actually happens due to the reduction of personnel in the project for version 7.5.

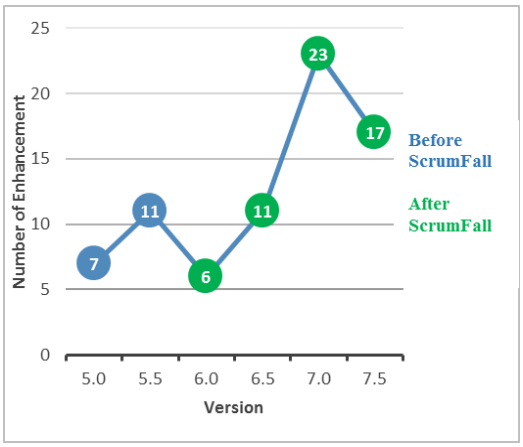

Fig.3. Comparison of Number of Enhancements delivered before and after following ScrumFall model.

From the practical experience of using ScrumFall model, we conclude that this model will be effective where the software project holds the following characteristics:

- The increment of each sprint such as Application Programming Interface (API), framework, library, Software Development Kit (SDK) is used by another team for development.
- Each feature is large and highly complicated.
- Geographically distributed large teams.
- The development environment is dynamic at earlier stages then gets stable in later stages.
- The team combines both experienced and inexperienced personnel.
- User involvement throughout the SDLC is not possible or necessary.

V .Conclusion

As the software technology is enriching every moment, the software models need to be improved to achieve a successful project with efficiency. Previous plan-driven models and agile models have their success on the specific types and scales of projects. For instance, Plan Driven models are suitable for large and stable systems, requires only experienced personnel at the beginning of the project and Agile is more suitable for small systems, requires expert agile personnel all over the project. From the practitioners' experience, ScrumFall holds the success over large, critical systems, geographically distributed large teams where the team is combined by both experienced and inexperienced personnel. In addition, ScrumFall has shown effectiveness in time, cost and

economic factors. In future, we plan to deploy this model on a large scale at different organizations to verify the further effectiveness of the proposed model.

APPENDIX A QUESTIONNAIRES TO EVALUATE THE PERFORMANCE OF SDLC

Characterization	
1.1	Let us know about your company (size of the organization, age, type of the organization, product or service type)
1.2	Let us know about your team (size of the team, distribution of the team, combination of personnel)
1.3	Let us know about the product (type, complexity, development environment, involvement of the user)
Process issue	
1.4	What were the difficulties with the previous process model?
1.5	Why was the ScrumFall model chosen?
1.6	How is the performance of the SDLC measured?
1.7	How much is user involvement required?
Impact Analysis	
1.8	How has the process model affected product delivery time?
1.9	Does the process model have any effect on features delivery?
1.10	Does the process model have any effect on bug fixing?

REFERENCES

[1] G. Elliott, *Global Business Information Technology*. Pearson Education UK, 2004, p. 87.

[2] K. Petersen and C. Wohlin, "The effect of moving from a plan-driven to an incremental software development approach with agile practices", *Empirical Software Engineering*, vol. 15, no. 6, pp. 654-693, 2010.

[3] R. Pressman, *Software engineering: a practitioner's approach*. Palgrave Macmillan, 2005, pp. 39-40.

[4] M.S. Rahim, A E. Chowdhury, D. Nandi, and M. Rahman, "Issue Starvation in Software Development: A Case Study on the Redmine Issue Tracking System Dataset*," Journal of Telecommunication, Electronic and Computer Engineering*, vol. 9, no. 3, pp. 185–189, Oct. 2017.

[5] A. E. Chowdhury, A. Bhowmik, H. Hasan, and M. S. Rahim, "Analysis of the veracities of industry used software development life cycle methodologies," *arXiv preprint arXiv:1805.08631 [cs]*, May. 2018.

[6] M.S. Rahim, M. Hasan, A.E. Chowdhury and S. Das, "Software engineering practices and challenges in Bangladesh: A preliminary survey*," Journal of Telecommunication, Electronic and Computer Engineering*, vol. 9, no. 3, pp. 163–169, Oct. 2017.

[7] S. Conte, H. Dunsmore and V. Shen, *Software engineering metrics and models*. Menlo Park, Calif.: Benjamin/Cummings, 1991.

[8] R. Pressman, *Software engineering: a practitioner's approach*. Palgrave Macmillan, 2005, pp. 41-42.

[9] C. Larman and V. Basili, "Iterative and incremental developments. a brief history", Computer, vol. 36, no. 6, pp. 47-56, 2003.

[10] B. Boehm and W. Hansen, "Spiral Development: Experience, Principles, and Refinements", Carnegie Mellon University, Pittsburgh, 2000.

[11] M. Smith, *Software prototyping*. London: McGraw-Hill, 1991.

[12]]N. Ruparelia, "Software development lifecycle models", *ACM SIGSOFT* Software Engineering Notes, vol. 35, no. 3, p. 8, 2010.

[13] R. Martin, *Agile software development: principles, patterns, and practices*, 1st ed. United States of America: Prentice Hall, 2002.

[14] K. Schwaber and J. Sutherland, *The scrum guide*. Scrum Alliance. Scrum Alliance, 2011.

[15] K. Beck and C. Andres, *Extreme programming explained: embrace change*. Addison-Wesley. Reading, 1999.

[16] N. K. Alexandros, D. P. Sakas, D. S. Vlachos, and N. K. Dimitrios, "Comparing Scrum and XP Agile Methodologies Using Dynamic Simulation Modeling," *Strategic Innovative Marketing Springer Proceedings in Business and Economics*, pp. 391–397, 2017.

[17] L. Plonka, H. Sharp, P. Gregory and K. Taylor, "UX design in agile: a DSDM case study", in *International Conference on Agile Software Development*, 2014.

[18] S. Palmer and J. Felsing, *A practical guide to feature-driven development*. Upper Saddle River, New Jersey: Prentice Hall PTR, 2002.

[19] D. West, M. Gilpin, T. Grant, and A. Anderson, "Forrester," *Forrester*, 26-Jul-2011. [Online]. Available: https://www.forrester.com/report/WaterScrumFall Is The Reality Of Agile For Most Organizations Today/-/E-RES60109#. [Accessed: 22-Jul-2018].

[20] S. Nerur, R. Mahapatra and G. Mangalaraj, "Challenges of migrating to agile methodologies", *Communications of the ACM*, vol. 48, no. 5, pp. 72-78, 2005.

[21] R. Chopra, *Software quality assurance: a self-teaching introduction*. Dulles, VA: Mercury Learning and Information, 2018.

[22] R. Hoda, J. Noble and S. Marshall, "Self-Organizing Roles on Agile Software Development Teams", *IEEE Transactions on Software Engineering*, vol. 39, no. 3, pp. 422-444, 2013.

Authors' Profiles

Md Shamsur Rahim obtained his B.Sc. in Computer Science & Software Engineering and M.Sc. in Computer Science from American International University-Bangladesh, Dhaka, Bangladesh. He is continuing his Doctoral study in University of Technology Sydney, Australia. His current research interest includes Data Science, Data Mining and Software Engineering. Currently, he is an Assistant Professor at the department of computer science at American International University-Bangladesh.

AZM Ehtesham Chowdhury completed B.Sc. in Computer Science & Engineering and M.Sc. in Computer Science from American International University-Bangladesh, Dhaka, Bangladesh. His current research interest includes Data Science, Data Mining, Software Engineering, Intelligent Systems. Computer Vision, Pattern Recognition and Human-Computer Interaction. He is an Assistant Professor at the department of computer science in American International University-Bangladesh.

Shahadatul Hakim is working as a Software Development Manager at Sensibill. He has completed his BSc in Computer Science from North South University, Bangladesh. He has more than 12 years of experience in the field of Software Development. His research interest includes Software Engineering, Software Project Management. Before that, he was the lead software engineer at Enosis Solutions, Bangladesh.

Dr. Dip Nandi pursued his B.Sc. in Computer Science from American International University-Bangladesh, Dhaka, Bangladesh. He obtained his MSc from the University of Melbourne and Ph.D. from RMIT University, Australia. Currently, Dr. Dip Nandi is the Associate Professor and Head of the Department of Computer Science, American International University-Bangladesh. His research interest includes Software Engineering, Technology E-learning, Theory of Learning and Human-Computer Interaction.

Mashiour Rahman is the Associate Dean of the Faculty of Science and Information Technology and Associate Professor at the Department of Computer Science, American International University-Bangladesh. He obtained his B.Sc. in Computer Science from American International University-Bangladesh and M.Sc. from National University of Singapore. His research expertise includes Artificial Intelligence, Machine Learning, Supply Chain Management, Theory of Computation and E-learning.

How to cite this paper: Shamsur Rahim, AZM Ehtesham Chowdhury, Dip Nandi, Mashiour Rahman, Shahadatul Hakim, "ScrumFall: A Hybrid Software Process Model", International Journal of Information Technology and Computer Science(IJITCS), Vol.10, No.12, pp.41-48, 2018. DOI: 10.5815/ijitcs.2018.12.06

