Enhancing Scrum with DevOps

Saliya Sajith Samarawickrama[#1], Indika Perera[#2]

[#]*Department of Computer Science, Faculty of Engineering*
University of Moratuwa, Moratuwa, Sri Lanka

[1]saliya.samarawickrama.15@cse.mrt.ac.lk

[2]indika@cse.mrt.ac.lk

Abstract— **With the presence of Agile practices, new forms of software development practices were evolved. Scrum, XP are some of them. But none of them can support the hourly deliver. The main factor for accepting the DevOps as the current trend is the fast nature of delivery. But development and delivery are not the only parts in software development life cycle. The research was to bring the rapid delivery to Scrum using DevOps and convert it to rapid software development method. A framework was developed to practice during the development life cycle and practiced with a development team to ensure the stability.**

Keywords— Rapid software development; Agile; Scrum; DevOps;

I. INTRODUCTION

Current context of software development is complex and all the manual operations are moving towards automated solutions [1]. The accelerated software delivery is a key to success in business. New technology trends i.e. mobile, cloud, mobile and analytics are demanding for better life cycle management of software. Maintaining the balance between stakeholders' requirements and developers' requirements is also a challenge. One of the most important facts is that in traditional software development approaches the yearly delivery plan has come to two-week build cycles and then to daily builds [2]. But the process has not stopped from that point, now build cycle time has been further reduced to hours [3]. The systems are cautiously evolving with the rapid business requirement changes and stakeholders are demanding for more transparency and measurements in development.

The current software development is driven by business and the main consideration factor in the business is cost [4]. Business stakeholders are more focused on reducing cost and in traditional software processes, such as waterfall the return of investment period is long. However in the current mainstream business competition the return of investment period needs to be short. To reduce the cost it is necessary to improve the software productivity and writing less code, getting the best from people, avoiding rework, developing and using integrated project support environments are the important aspect to improve the process productivity [4]. In order to improve the productivity the understanding on types of costs is a must. Development and rework costs, code and documentation costs, labor and capital costs, software costs by phase and activity are the types of costs involve in software developments [4].

To eliminate above costs the spiral model was proposed in [4]. However the software industry continued to demand better productivity and flexibility; as a result Agile was introduced to the software development [5]. From late 1990 Agile is in the mainstream software development process types and there are many forms of the Agile process such as Scrum; the first postulated form of scrum [6]; Extreme Programming, Crystal etc.

The present status is much more competitive and business landscape is changed to demand for "continuous delivery" and "development and operations" [7]. As a result DevOps comes to the software development industry. Devops has been adopted to leading organizations i.e. IBM [8] in order to support enhanced collaboration across the organization and value chains. Other than continues delivery and coexistence between development and operation the following are also considered [9].

- Improve the quality and performance of applications
- Enhance the end customer experience
- Simultaneously deploy software across different platforms

II. RESEARCH PROBLEM

Software process models cover following items [10].

- Requirements Analysis and Specification
- Software architecture and Implementation
- Testing and Documentation
- Training, Support and Maintenance

In all the process models software maintenance was considered as a separate sub process and developers focus on rest of the process activities. This has created a crucial gap between the developers and the production and operations staff, leading to a number of challenges and process overheads. DevOps, an improved version of merged development and operations has been explored recently in order to cover the development and operation gaps, which are present in the regular software engineering process models.

As discussed above DevOps focuses on continues integration and continues delivery. It does not cover any managerial aspect of software development lifecycle. Organizations practice DevOps at present without a proper framework for their software process management.

III. LITERATURE REVIEW

With the increasing demand for alteration, new software development methodologies were presented. Rapid deliveries and time to market has become the most dominant factors in software industry. Traditional approaches such as Waterfall was moved out from the mainstream and new methodologies based on Agile have gained the popularity. As per Andrew and Nachiappan [5], Agile is the most used concept in the software industry. Agile introduced four principles such as:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

Above concepts do not explicitly relate with the major activities of software development lifecycle which include deriving the specification, design and development evaluation and evolution [11]. As a result several methodologies have been evolved from the basic principles of Agile concept. Scrum, Extreme Programming, Lean, Kanban are some examples.

Scrum is the widely used methodology as an Agile software development practice [5], [12] and Scrum covers the following by its rich set of guidelines. It comprises project management as part of its practices [13]. Scrum creates a product backlog about the pending development. The Scrum method includes product owners, developers which will allow the get a broad picture of the development. As per Hneif and Ow [13] Scrum is not suitable for products which focus of usability because product owner is focused on business aspect.

The other most popular Agile practice is Extreme Programming (XP) [4]. XP consist of four phases: Planning, coding, designing and testing. XP can adapt changes at any time. XP defines the concepts such as following.

- Small releases with high-value elements
- Refactoring and Pair programming
- Continuous builds
- Sustainable development and reduce maintenance costs

Above points out that XP focuses towards the technical side of Agile developments. The concept of small releases and continuous builds lead to the new concept of DevOps.

DevOps is a new approach/new trend in software development and it reduce the gap between developer, operation and the end user [3] which leads to detect problems early. As in Scrum, though the system developed according to specification it may not get validated by end users. DevOps support continuous development and integration to avoid those pitfalls. But in DevOps there is no known framework or process [3]. To become stable DevOps should have a complete software process model, which supports specification, design and development, evaluation and evolution phases.

IV. METHODOLOGY

As discussed in the previous section, demand for rapid delivery is booming. This is purely because of the competition in the business domain and having the accelerated deliver the key to success. Companies believe that the antidote to this issue is DevOps. Most of the companies adapt DevOps without knowing that the DevOps is a cultural movement. Companies who practice Agile still struggling to adapt to rapid delivery cycles.

From the literature, it's proven that the Scrum is the best methodology to be used as an Agile practice. The success behind this is based on the proper declaration of the flow throughout the software development process. Scrum never discuss about the maintaining the quality of the product. As a result one drawback of Scrum is that the developers may conform that a development has been done without the most suitable implementation. Rather than the most suitable implementation, developers may focus on the easiest implementation.

In the other hand DevOps focuses on the long run of the software with shorter feedback, quick integrations and rapid deliveries. DevOps falls behind due to the unavailability of proper defined guideline to follow during the software development lifecycle.

The study is focused on building a methodology where Scrum is used to cover the managerial aspect of the software development lifecycle and DevOps is used to support the rapid delivery requirement. Scrum is chosen because it has proven that it's the best Agile practice. Most of the software developments fails due to less control over the total flow. In Scrum the loop holes have been closed and the adaptation proved that the Scrum is the best framework in Agile.The solution will bridge the Scrum which covers the development aspects and DevOps which covers the rapid delivery aspect.

A. Gaps and overlaps

The development and deployment flow between Scrum and DevOps cannot be done without filling the missing components. To understand this need to check the role and the requirements of Scrum and DevOps in different stages of software lifecycle. Given below the stages of software development lifecycle and current scrum methodology covers the first three aspects. They are covered up to the development and release stages only. Final stage is not covered at all by the scrum process.

- Planning
- Execute
- Inspect and adapt
- Operation and support

But DevOps cannot combine directly with the current Scrum flow. To understand the gaps and missing components, first it's required to understand the industry requirement. The main requirement is to minimize the time to market factor and stability of the product. The competition between vendors are much high and one failure may causes the business to end and increase the customer churn rate. Other than that industry need to attend the production issues as fast as possible to keep the customer satisfaction and retain the customer with the product.

According to Michaels' [14] Components of DevOps framework (namely, Metrics and measurement view, Process view, Technical view) is crossed checked with the industry requirement given above, the focus area is getting narrowed.

To cater the industry requirement, the lagging parts are fast feedback from production, push features faster to production. This is covered under the technical view. With that fact, we can conclude the Scrum doesn't need to adapt to all the views. It's enough to adapt the technical view.

To achieve the fast feedback and the frequent feature update to production requires continuous integration and continuous delivery. These are the main components of technical view. In order to achieve continuous integration and continuous delivery following factors are required [15].

1) Single Repository to Manage the Code Base

In the continuous integration and continuous delivery context a central repository is mandatory. So to merge the two solutions, it's a mandatory requirement to have a central repository.

2) Automate the Build

Once a feature is developed, developer has to test the build in the local machine. For this, the developer should have the capability to build the solution automatically. Other than that after committing the feature to main line, main line should be built automatically. This is a new concept for Scrum. So to build the solution, build automation should be introduced.

3) Keep the Build Fast

In Scrum user stories are developed from end to end. Because there's no requirement to build fast. In order to merge the continuous integration and continuous delivery to Scrum, user stories should be developed in an incremental manner and should commit to the central repository for the build.

4) Self-Testing Build.

In the Scrum context, the quality assurance team (QA team) develop the test cases for the software and after releasing a user story they test the component of the software. Other than that sometimes developers develop the unit test cases and check for issues.

5) Commits to the Mainline Every Day

In the current Scrum practice the estimations are done by whole team and estimation time are done in days. But to support "At least one commit per day" need to break the user stories feather down. This should be a responsibility of the developer who owns the user story.

6) Fix Broken Builds Immediately

This is not practices in Scrum currently and to merge continuous integration and continuous delivery to Scrum this condition should be introduced.

7) Test in a Clone of the Production Environment

In order to smooth line the continuous integration and continuous delivery, software should test in the clone of the production system.

8) Everyone Can See What's Happening

To fill the gap between the Scrum and the continuous integration and continuous delivery a tool should be developed. This will be discussed in detail under design and implementation chapter.

9) Automate Deployment

In Scrum concept of automated deployment is not present. So adhere continuous integration and continuous delivery to Scrum, automated deployments should be available. With this getting the latest executable is also possible.

B. New Framework

Supported by above facts we can conclude that the methodology should cover following areas. Each area will have a set of condition that need to satisfy. So to define the new framework the combination of Scrum, continuous integration and continuous delivery, need to modify Scrum rituals, rules. There modifications are done to support continuous nature.

- Development environment setup
- Planning
- Execute
- Inspect and adapt
- Operation and support

According to section A following are the identified Requirements and the policies for each section.

1) Development Environment Setup

In order to achieve the continuous integration and continuous delivery in a Scrum environment, the code management and the automation should be implemented. Following are the key factors that need to have in the environment.

- A Single Source Repository to manage code base
- Separate identical environments for testing, staging and production
- Build automation tool
- Test automation tool
- Configuration management tool
- Overall progress monitoring tool

2) Planning

Requirement gathering will not be much differ from the current approach. The conditions that need to cover are commits to the mainline every day and giving high priority for production issues.

Rule set for planning is given below.

- The user story estimation is done as per the current practice

- If the estimation of the user story is greater than a day, sub user story should be developed which should be shippable
- The sub user story definition is the responsible of the assigned developer. This is to minimize the time take to the meeting.
- High priority should be given to the production issue than the backlog

3) *Execution*

To Support continuous integration and continuous delivery in a Scrum environment can be achieved by competing the test automation and build automation.

Rule set related to developers is given below.

- Developer should write unit test cases
- Quality assurance should write the test cases
- Developer should test the code against unit test
- Developer should test the build against the test cases
- If both passes, commit to the main line
- Developer is responsible for the commit and should make sure the mainline is built successfully
- If not developer should fix it immediately
- Developer should run all the test cases in the mainline
- In this stage developer can use a code quality check tool.

4) *Inspect and adapt*

To support the continuous integration and continuous delivery, testing should be automated as far as possible. Some cases manual testing cannot be replaced. For example UI testing, the look and feed cannot be automated. Main focus is to automate the test and this will lead to have merged stage of implementation and testing in the software development. Each test should be executed in each environment before going to production.

Rule set for quality assurance / testing is given below.

- Should write test cases and finish before end of development
- Make the test cases available to developers to run on the new build
- Should not port the feature to staging till all the test cases passes
- Should run all the automated test cases in staging

5) *Operation and support*

This is the new area that appears, which was not in the Scrum earlier. This area covers the automated deployments, performance monitoring of the system and escalating the operational issues to the developer. DevOps discuss about this aspect as the fast feedback from production to development.

To support continuous integration and continuous delivery deployment should be automated. In a theoretical scenario, deployment of automation should be implemented to the production level. But in a practical scenario there can be restriction applying updates to production each day or there can be approval process on applying updates on production. But this restrictions are only valid for production and automated deployment can be done up to staging without any issue.

Rule set for Deployment is given below.

- Deployments should be automated up to the staging level
- If there's no restriction, automations should extend to production
- If there's restrictions, maximum time should be defined to release a feature to production.

Rule set for performance monitoring is given below.

- Performance issues should be escalated to developers from production
- Developers should address related performance issues in the next sprint

Operational issues can be identified either from support desk or log analyzers. Issues reported from both method should be treated as follows.

Rule set for operational issue escalation is given below.

- Critical bugs reported should be fixed in the current sprint or as hot fix
- Medium and low bugs should be added to future sprints based on the impact

C. *Evaluation methodology*

To verify a new method, an evaluation has to done. Without the quantitative factors we cannot say that the new method is more successful than the previous method. To compare current Scrum practice and the proposed methodology there should be an evaluation method.

In scrum evaluation, following matrices are been used [16] [17].

- Sprint Burn-down
- Epic and Release Burn-down
- Velocity
- Control Chart
- Cumulative Flow Diagram
- User stories planned versus user stories delivered
- User stories delivered versus user stories accepted
- Defect-removal efficiency (DRE)

In DevOps context following matrices are been used [18].

- Deployment (or Change) Frequency
- Change Lead Time
- Change Failure Rate
- Mean Time to Recover (MTTR)

According to paper Jeeva Padmini [19], following are the leading matrices used in Sri Lanka for agile software development.

- Work capacity
- Percentage of adopted work
- Sprint-level effort burndown
- Velocity
- Percentage of found work
- Focus factor

To evaluate the new methodology use of following matrices are proposed. This matrices coves the Scrum and the continuous integration and continuous delivery aspects. The matrices were chosen in such a way that enables to compare standard scrum and proposed methodology

- User stories planned versus user stories delivered
- User stories delivered versus user stories accepted

- Defect-removal efficiency (DRE)
- Deployment (or Change) Frequency
- Velocity

V. EVALUATION AND FEEDBACK

There were multiple aspects that need to evaluate. Those are given below and different approaches have been used to evaluate the components. Finally combined outcome had been evaluated.

- Proposed Framework
- Framework practice outcome

The proposed framework was evaluated by a set of Scrum Masters, Project Managers and Developers. To evaluate the proposed framework set of questions was used. Given below the set of question that used.

- Proposed framework will deliver better results relative to Scrum
- Proposed framework can be adapted with minimum changes
- Proposed framework contains all the necessary guidelines for development
- Proposed framework enforce the CI CD with proper guidelines
- Proposed framework improve the end to end viability (from Development to operation)
- Proposed framework improves the velocity
- Proposed framework improves the code quality and standard
- Proposed framework improves the rate of delivery
- Proposed framework improve the Mean time to recover
- Provided tool enhance the end to end visibility
- Provided tool shows the bottlenecks in the development cycle and get the early attention
- Provided tool's modular architecture enables easy integration with third-party tools

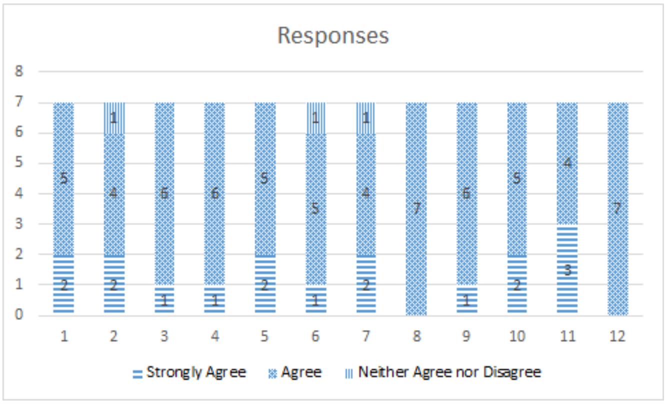

Fig. 2 Questioner responses

TABLE 1

TEAM FOLLOWING THE STANDARD SCRUM

Matrix	Value
User stories planned	5
User stories delivered	4
Percentage of user story delivered	80
User story accepted	4
Percentage of user stories accepted	100
No of Dev defects found	5
No of Dev defects fixed	3
Defect-removal efficiency	60
Deployment Frequency	1
Expected Velocity	25
Actual Velocity	20

Evaluation of results are shown in table 1 to table 3 and the comparison of results has been shown in table 4.

TABLE 2

TEAM FOLLOWING THE ENHANCED SCRUM - FIRST SPRINT

Matrix	Value
User stories planned	4
User stories delivered	3
Percentage of user story delivered	75
User story accepted	3
Percentage of user stories accepted	100
No of Dev defects found	8
No of Dev defects fixed	8
Defect-removal efficiency	100
Deployment Frequency	3
Expected Velocity	20
Actual Velocity	15

TABLE 3

TEAM FOLLOWING THE ENHANCED SCRUM - SECOND SPRINT

Matrix	Value
User stories planned	4
User stories delivered	4
Percentage of user story delivered	100
User story accepted	4
Percentage of user stories accepted	100
No of Dev defects found	5
No of Dev defects fixed	5
Defect-removal efficiency	100
Deployment Frequency	4
Expected Velocity	20
Actual Velocity	20

TABLE 4

COMPARISON ACROSS SCRUM APPROACHES

Matrix	**Standard Scrum**	**Enhanced Scrum**	
		First Sprint	**Second Sprint**
Percentage of user story delivered	80%	75%	100%
Percentage of user stories accepted	100%	100%	100%
Defect-removal efficiency	60%	100%	100%
Deployment Frequency	1	3	4
Expected Velocity	25	20	20
Actual Velocity	20	15	20

The first attempt of the Enhanced Scrum shown a decrease in user story delivery and actual velocity. This was due to the learning curve of the new framework. But there was an improvement in deployment frequency and the defect removal rate.

The second sprint that followed the Enhanced Scrum has shown an improvement over the first attempt and actual velocity was equal to the reference value. Also it had improved over the user story delivery and acceptance. When the second sprint was compared again the reference sprint the defect removal rate and deployment frequency had been improved.

In a nutshell, after the learning curve the Enhanced Scrum approach had improved the team performance.

VI. CONCLUSION

The rapid delivery nature that demands by the industry is major concern and development practices are searching for various alternative methods to address this issue. Though the industry follows different practices which are developed in house to cater the requirement, there's no standard framework. This framework was proposed to standardize the adaptation of continuous nature to the industry.

The most adapted methodology in the Agile context is Scrum that because of its rich management aspect. To benchmark the new framework a previous sprint data was used where the team was following the traditional Scrum practice. Key aspect that targeted at the beginning of the research was to reduce the time to go live, increate the feedback from operation and improve the end to end visibility.
The matrices were chosen to validate above key aspects. Results in the table 4 shows the outcome of the proposed framework. The comparison between traditional scrum sprint and the second sprint of the new framework shows the successfulness of the proposed framework. With the improvement of the defect removal efficiency, it proved the fast feedback from the operation. With the improvement of deployment frequency, it proves the reduction of time to go live. The expected velocity and the actual velocity was equal in the second sprint and that proves the improved end to end visibility. With that we can conclude that the proposed framework address all the key aspects.

REFERENCES

[1] N. M. J. Basha, S. A. Moiz, and M. Rizwanullah, "Model based Software Develeopment: Issues & Challenges," Int. J. Comput. Sci. Informatics, 2(1-2): 226–230, 2012.

[2] Walker Royce, "5 Biggest Challenges for Software and Systems Delivery Teams (The Invisible Thread)." .https://www.ibm.com/developerworks/community/blogs/invisiblethread/entry/5-biggest-challenges-for-software-and-systems-delivery-teams?lang=en May 24 2012

[3] F. Erich, C. Amrit, and M. Daneva, "Report: DevOps Literature Review," no. October, p. 27, 2014.

[4] Boehm, B. & Papaccio, P. Understanding and controlling software costs. IEEE Transactions on Software Engineering, 14(10), 1462-1477, 1988

[5] A. Begel and N. Nagappan, "Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study," First International Symposium on Empirical Software Engineering and Metrics, no. 9, p. 10, 2007.

[6] K. Schwaber and M. Beedle, Agile Software Development with SCRUM: Prentice-Hall, 2002

[7] Saugatuck Technology, “Why DevOps Matters : Practical Insights on Managing Complex & Continuous Change,” no. October, 2014.

[8] A. Reddy, “DevOps : The IBM approach,” pp. 1–12, 2013

[9] C. A. Technologies, “DevOps : The Worst-Kept Secret to Winning in the Application Economy DevOps : The Worst-Kept Secret to Winning in the Application Economy,” no. October, 2014

[10] M. S. Maheshwari and D. C. Jain, “A Comparative Analysis of Different types of Models in Software Development Life Cycle,” Int. J. Adv. Res. Comput. Sci. Softw. Eng., vol. 2, no. 5, pp. 285–290, 2012.

[11] O. Salo and P. Abrahamsson, “Empirical Evaluation of Agile Software Development: the Controlled Case Study Approach,” City, 2004.

[12] A. B. M. Moniruzzaman, “Comparative Study on Agile software development methodologies.”

[13] M. Hneif and S. H. Ow, “Review of Agile Methodologies in Software Development 1,” in International Journal of Research and Reviews in Applied Sciences, 2009, vol. 1, no. 1, pp. 2076–73

[14] Michael Hüttermann, “DevOps for Developers,” Apress , 2012.

[15] M. Fowler, "Continuous Integration", martinfowler.com, 2017. [Online]. Available: http://www.martinfowler.com/articles/continuousIntegration.html. [Accessed: 19- Nov- 2016].

[16] "Five Agile Metrics You Won't Hate The Agile Coach", Atlassian, 2017. [Online]. Available: https://www.atlassian.com/agile/metrics. [Accessed: 15- Oct- 2016].

[17] "A Simple Way to Measure the Performance of Scrum Teams - Scrum Alliance", Scrumalliance.org, 2017. [Online]. Available: https://www.scrumalliance.org/community/articles/2014/may/simple-way-to-measure-performance-of-scrum-teams. [Accessed: 15- Mar- 2017].

[18] C. Riley, "Metrics for DevOps - DevOps.com", DevOps.com, 2017. [Online]. Available: https://devops.com/metrics-devops/. [Accessed: 20- Mar- 2017].

[19] K. V. J. Padmini, H. M. N. Dilum Bandara, and I. Perera, “Use of software metrics in agile software development process,” 2015 Moratuwa Eng. Res. Conf., no. May, pp. 312–317, 2015.