Applied Computer Systems

ISSN 2255-8691 (online)
ISSN 2255-8683 (print)
May 2017, vol. 21, pp. 71–79
doi: 10.1515/acss-2017-0010
https://www.degruyter.com/view/j/acss

Several Ideas on Integration of SCRUM Practices within Microsoft Solutions Framework

Jai Vigneshwar Alavandhar[1], Oksana Ņikiforova[2]
[1]*KPMG Crimsonwing Ltd, Malta,* [2]*Riga Technical University, Latvia*

Abstract – **In order to develop and deliver a software project successfully, any software development organisation has to follow a well-known and recognised software engineering process for successful delivery and maintenance of the software. However, when the organisation is willing to follow a new software development process, the success rate of adopting a new software engineering process is a question mark. In the paper, we aim at studying and comparing two software engineering processes, which are based on different paradigms or models, and proposing a hybrid methodology, which integrates advantages of both compared methods. They are Microsoft Solutions Framework as a representative for an iterative methodology and SCRUM for agile software development. The comparative analysis will help a software development company to make the transition easier from Microsoft Solutions Framework to SCRUM or vice versa.**

Keywords – **Agile software development, iterative software development, Microsoft Solutions Framework, SCRUM, software engineering.**

I. Introduction

In order to develop software, we need to follow some methodology for successful delivery of the software system. There are various software development models available, but choosing the “right” model is a very complicated task. Under each model or methodology, there are various frameworks available. Choosing the model of software development process and the underlying framework depends on customer requirements, budget, time and other factors.

Moreover, the issue of choosing the model of software development process becomes even more complicated, when software developers have a long-term experience in using such software project organisation and need to move to some other paradigm of software development based on customer needs or other conditions.

For example, one of the most popular software development methods, which define a strong set of activities to perform, artefacts to develop and role to share responsibilities, is Microsoft Solutions Framework (MSF) [1]. It is a strongly disciplined and well-defined model for software development, with planning and requirements definition. Another style of development is agile software development, where one of the most known and used methodologies is SCRUM [2]. When a software development company or a customer of the software company is willing to adopt SCRUM over MSF or vice versa as their software development model, then transformation to their software practices and principles requires an in-depth understanding and comparison between various processes and sub-processes that will allow for fast and steady transition from one software development model to another or a hybrid of SCRUM and MSF, which will probably bring the best of both models.

To make comparison between MSF and SCRUM, it is necessary to understand a software engineering process and its key points. Within the framework of the research, both software engineering processes will be compared by their principles, procedures, processes and best practices, as well as phases, segmentations and work responsibility classifications. Finally, the similarities and differences between MSF and SCRUM will also be examined. Based on the information collected, this comparative analysis can provide general recommendations and serve as a guide for recognising project situations for which these software engineering processes are appropriate and can be integrated within another one.

II. Background and Related Work

Software system development process has core activities, such as Requirements Gathering, Design, Development, Testing, Deployment and Maintenance [3]. There are various software development models, for example, waterfall [4], prototyping [5], incremental [6], V-model [7], spiral [8] and agile [9], which are used to develop software systems. Each of these models includes the core activities stated above, but may or may not have additional activities and the order of the activities may sometimes vary depending on the model. Choosing the corresponding model for developing a software system is a very complicated task [10]. Software companies, based on their customer requirements and the project itself, will select the type of software development model that suits their application. In each period, different software development models were popular, for example, the traditional waterfall model was used as a model for most software and system development projects. However, it failed in some cases where requirements were changing or the requirements were misunderstood and then there were other software development models introduced to overcome such situations. With the waterfall model, testing starts after development is completed, so the cost of fixing defects is too high and risky. The incremental, iterative, spiral models are based on the requirement of customer and need for the product [10]. V-model is also popular in the industry because the development and testing will look like running in parallel so that it will minimise some risks and defects known before delivery.

At present, the agile methodology is popular and commonly used due to the reason that software systems are delivered as parts and the priority is given to the most wanted functionality or future so the customer can go live within a short period of time. Thus, in the current fast growing software industry the customers cannot wait for long time as in traditional development and the Agile way of developing software helps the customer to go live soon and helps the software development company to bring more value to the customer. There are many agile software development frameworks such as SCRUM, Extreme Programming [11], Rapid Application Development (RAD) [12] etc., which are meant for their own features and benefits. In sequential process (Fig. 1), the core requirement to start developing the software system is the list of system requirements. The waterfall model is a kind of sequential process. Based on the system requirements, the software system will be built step by step. Each step is a phase and there is no option to look back to the previous phase after entering the other phase. This process will be carried out sequentially to deliver the whole software product.

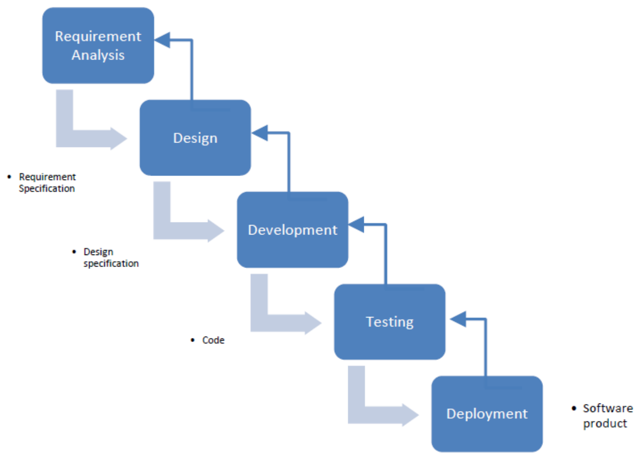


Fig. 1. Sequential development process.

In the sequential development process, there is no option to go back to the previous phase. Due to this reason, the rest of the phases in this process will be executed until the end without looking back and progress with assumptions that everything is going in right track. However, in the end the software solution group did not meet customer requirements and the solution failed. This is because one cannot identify all the requirements at the beginning stage and freeze it to move to next phase. For a complex system, it is not possible to understand the requirements fully so there will be some assumptions that the requirements are not in detail. Therefore, it is important to analyse the requirements in detail. Some requirements will be clear only after some progress is done and maybe in future requirements will also change because the needs will change. The problem with the sequential development process is that the customer will see the software system only at the end of the process or the implementation and so due to this reason the customer cannot correct anything in-between and the customer may go unsatisfied if the system does not satisfy real requirements. It is like launching a rocket, you have only one opportunity to perform the process of launching and once it is launched then you cannot correct your mistakes made and once the path is deviated it is completely deviated and you cannot improve anything other than starting the process from the beginning.

The iterative software development process (Fig. 2) will start with the basic set of requirements and complete the process, which is called on as one iteration finishes results in producing version 1 of the software product. During the first iteration, the requirements are understood well and with the next set of requirements the 2nd version of software development process will happen. The software system is developed part by part; thus, each part is developed in one iteration. In other words, this iterative process will bring enhancements on each build released, so the software system will go more matured after every iteration. At every iteration any modifications required can be included along with a new set of requirements. The iterative model can only be used under certain scenarios. The requirements of the system need to be complete and understood, in case there are chances that new technologies are introduced and resources with definite skills are available for only a short period of time and mainly there are high risks and the goal changes in future. However, there are also some disadvantages with the iterative process model. It is not suitable for projects of small size and the management involvement is required most of the time, which also creates more complexity. Moreover, highly skilled resources are required and also many resources are needed. There are also more project risks and the project is dependent of the analysis of risks.

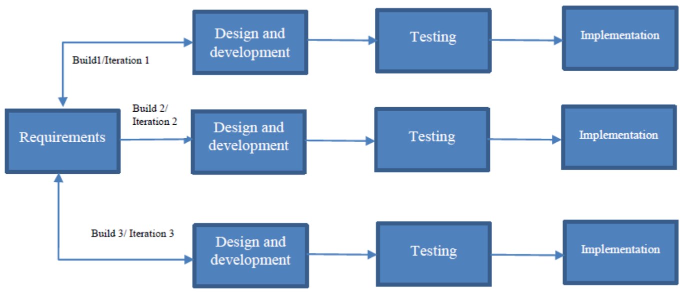


Fig. 2. Iterative development process.

Agile Development (Fig. 3) uses an iterative and flexible approach to software development. The software is developed in an incremental cycle, which results in small incremental releases. Usually the agile process is preferred in time critical projects. In agile model, each iteration is considered sprint, which has a planning stage, development, testing, and deployment and finally sprint demo. In the planning stage, the backlog item or user stories or bugs are gathered and planned according to the time period of sprint. After planning, the development stage starts and then testing will follow. After tests are passed, the items are deployed and presented to the customer as a working feature. Agile addresses the emergent nature of developing software, accounting for the fact that requirements change and become known during development.

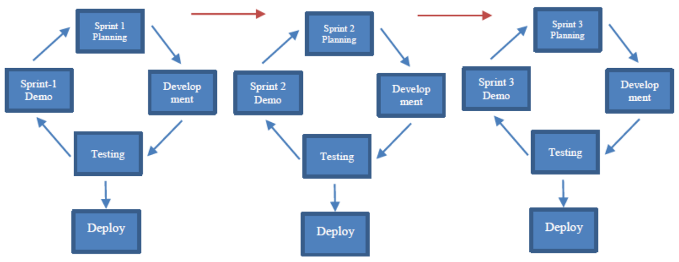


Fig. 3. Agile development process.

Software applications developed through the agile process have three times the success rate of the traditional waterfall method and a much lower percentage of time and cost overruns (Fig. 4):

- Project initiatives that are rigid run the greatest risk of dissolution. Yet, having a flexible, formal process has been proven to greatly improve the success rate.
- Steppingstones are a key driver for the success of the agile and iterative software development process.
- Quickness and velocity are vital to an agile process, and that encompasses feedback. However, a major advantage of the agile process is the closeness of the executive sponsor or, in agile terms, the owner.

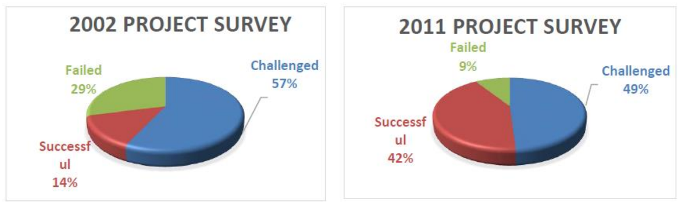


Fig. 4. Agile today – in the mainstream [13].

Both the Forrester (Fig. 5) and VersionOne (Fig. 6) surveys strongly indicate that SCRUM was the most common agile development method employed in 2010.

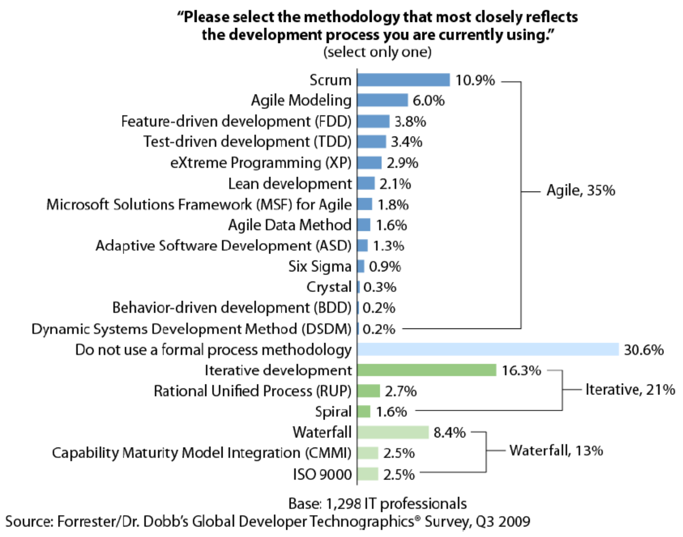


Fig. 5. Forrester Q2 2010 [14].

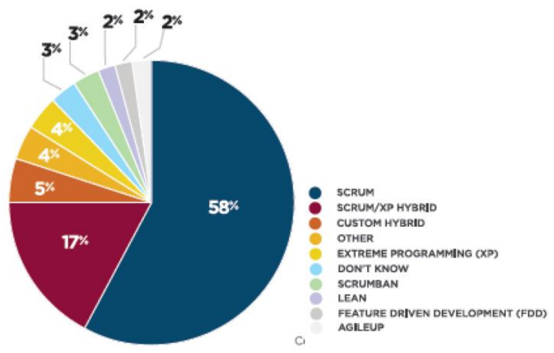


Fig. 6. VersionOne 2010 Survey [15].

The paper compares the software engineering processes called Microsoft Solutions Framework of iterative methodology and SCRUM of agile methodology based on their characteristics, foundation principles, processes, disciplines and the study of their merits and demerits, proposing a hybrid software development model for Agile software development.

III. Comparison of MSF and SCRUM

Managing the development of software products in real environment is much more complex than the theoretical line as in theory we cannot predict all problems that may arise in practice; however, often the methodology in real application is viewed as framework.

On the other hand, the questions on historical development methodologies usually emphasise that the old methodology is inadequate and the new methodology has corrected the shortcomings of previous ones and brought new and advanced ideas. However, it should be noted that most of today's operating systems, solutions for word processing, all serious RDBMS systems, most Mail servers, and all major ERP systems and so on are created using the old methodologies. It should be borne in mind that today, with agile methodologies, there is very little new software developed, and most applications undergo the development stage, when more and more serious systems are concerned. Some of them are working on new versions for improvement and enhancement, minor finishing and some refactoring in the existing systems that have been developed through older methodologies.

The question is whether the new methodology exactly is much better the traditional methodologies and which are the characteristics that distinguish it compared to the traditional ones. In theory, agile methodologies usually allocate several key areas, such as:

- Management team;
- Changes required by the client;
- Product quality;
- Definition of architecture;
- Documentation;
- Return on investment.

According to these key areas, Tables I–IV give a comparison of the practical vision of two methodologies, SCRUM and Microsoft Solutions Framework.

TABLE I

COMPARISON OF MICROSOFT SOLUTIONS FRAMEWORK TO SCRUM BY KEY AREAS

Microsoft Solutions Framework	**SCRUM**
Basic principles	
Microsoft Solutions Framework is a combination of waterfall and spiral models.	SCRUM uses an iterative and incremental development process.
MSF is formed after analysing the key characteristics of successful projects, which were then incorporated to the MSF framework.	SCRUM is an agile methodology for software development that is ideal for projects with rapid changes or with very urgent requirements.
Iteration is the basic unit of development in Microsoft Solutions Framework.	Sprint is the basic unit of development in SCRUM. With the progress of SCRUM, a project is going through a series of iterations called “sprint”. SCRUM is sometimes said to be a thin framework [16] rather than a strict methodology.
The duration of an iteration with MSF is between 30 and 90 days and in between changes are not permitted.	The maximum duration of sprint at SCRUM is 30 days and in between changes are not permitted.
Management	
According to MSF, complete team is managed by program management, which often has a dual role, Project Manager and Solutions Architect. This role is responsible for planning, implementation and monitoring of plans (using the defined control points) and reporting to the master formal management.	Basically, the team is managed autonomously and partially by SCRUM Master who provides training and guidance in the SCRUM approach, monitors the team work, and attempts to remove any obstacles to productivity and reports to a product owner. At the beginning of the project, according to the SCRUM methodology, formal management chooses the owner of the product (Product Owner) and SCRUM Master, who then elect the remaining team members.
In MSF, there are 6 role clusters and each role cluster has one or more functional areas and each functional area may have one or more responsibilities and the responsibilities have more tasks. One role is not equal to one person responsible for it. One role may be supported by multiple persons or depend on the size of the project, one or more roles can be taken care by one person.	SCRUM team is cross-functional. Apart from developers, the SCRUM team will include software testers, front-end designers etc. The SCRUM team members have different skill sets and they train each other so that no single team member will be at risk for the delivery of the project. All team members will help each other for smooth completion of the sprint.
Change requests	
Microsoft Solutions Framework supports the amendment request between iterations.	SCRUM supports the amendment request among sprints.
During the Iteration with MSF (between 30 and 90 days), changes are not permitted. However, the client (purchaser of work) can be interrupted if it is estimated that certain changes need to be implemented in particular iteration.	During the Sprint at SCRUM (maximum 30 days), changes are not permitted. However, the product owner can interrupt the sprint if it is estimated that certain changes need to be implemented in particular sprint and sprint can also be stopped by him/her.
Quality	
MSF provides a testing role to a developer who provides feedback on the objectives relating quality solutions and determines the actions that will be necessary for reaching the level of quality. The testing applies decisions on the objectives, relating to quality, strategy testing and acceptance criteria to be used for quality measurement. Testing role also carries bug tracking in accordance with defined internal procedures at the level of business system.	In SCRUM when comes to testing, it is conducted by members of development team or sometimes a dedicated Quality Analyst who is part of the SCRUM team. The attitude of using development team members for testing in practice is not realistic because testing entails specific organisation and documentation, specific tools and technology and specific knowledge and skills, but also the specific mind-set of the people who carry out testing.
Defining architecture	
According to MSF, a program manager or dedicated architect is in charge of defining the architecture. Solution and technical architecture are carried over by them. After defining the architecture, the whole team works on the solution.	In agile methodologies, principles of agile architecture are mainly the responsibility of the team. SCRUM does not support formally defining the functions or roles in the team, such as architect, developer, tester, etc. In such situations, people with extensive knowledge and experience and therefore authority will always stand out in a team as technological leaders of the team and their opinion will always have extraordinary weight on the whole team.
Documentation	
Full documentation cycle. When it comes to design, MSF predicts creation of functional specifications. Since there is no pre-defined form in the MSF framework, the way of documenting defines not only the business system level and internal standards, but also the level of each project.	Generating a minimum of documentation is positioned as an advantage. Tentatively documentation can be divided into three groups: project documentation, project management documentation and instructions for use and maintenance. When the project documentation is concerned, in SCRUM usually two types of documents are created: Project Backlog and Sprint Backlog. Both types of documents are written in very short sentences that clearly define the required functionality. When the project or team management documentation is concerned, SCRUM involves two types of documents: Sprint Burndown chart and Release Burndown chart. Sprint Burndown chart is used to monitor the progress of concrete Sprint day, while Release Burndown chart is used to monitor the progress of the project after sprints. The problem arises in large projects with more SCRUM teams when necessary to compare the results of multiple teams or to present their joint effect. The problem is calculated for Y axis which usually displays weight points (Story Points) who burned the previous day (Sprint Burndown Chart) or in the previous sprint-in (Release Burndown Chart). When it comes to the documents related to the maintenance of the production and use of concrete solutions, according to SCRUM, the team writes documentation maintenance solutions and user instructions in compliance with the defined client requirements.

TABLE II

COMPARISON OF MICROSOFT SOLUTIONS FRAMEWORK TO SCRUM BY PRINCIPLES

Microsoft Solutions Framework	**SCRUM**
1) Work toward a shared vision: It enables agility so that the team members are able to make informed decisions quickly in the context of achieving a vision.	Individuals and interactions over processes and tools: SCRUM is a team-based approach to delivering value to the business. Team members work together to achieve a shared business goal. The SCRUM framework promotes effective interaction between team members, so the team delivers value to the business.
2) Deliver incremental value: Determine optimal increments and make sure what is delivered has optimal value to stakeholders. Working software over comprehensive documentation: Focus on delivering valuable items sooner. SCRUM requires a working, finished product increment as the primary result of every sprint.	A SCRUM team's goal is to produce a product increment every sprint. The increment may not yet include enough functionality for the business to decide to ship it, but the team's job is to ensure the functionality present is of shippable quality.
3) Stay agile, expect and adapt to change: Change can happen and the organisation should be able to adapt and adjust to a change. Having an agile way to handle a change helps minimise common disruption caused by a change.	Responding to change by following a plan: SCRUM teams make frequent plans. For starters, they plan the current sprint. However, the team's goal is not to blindly follow the plan; the goal is to create value and embrace a change. SCRUM teams constantly respond to a change so that the best possible outcome can be achieved.
4) Partner with internal and external customers: This increases project success because the internal and external customers are involved in development of a project.	Customer collaboration over contract negotiation: SCRUM is a framework designed to promote and facilitate collaboration. The team, especially the product owner, collaborates with stakeholders to inspect and adapt the product vision so that the product is as valuable as possible.
5) Establish clear accountability and shared responsibility: Team member accountability leads to higher quality and all team members should share responsibility for the overall solution and its deliverables.	Self-organising teams: A group of motivated individuals, who work together toward a goal, have the ability and authority to take decisions and readily adapt to changing demands. They work for themselves, manage their work, do not require command and control and they communicate more with each other, they continuously enhance their own skills etc. Unlike MSF, an induvial is not accountable instead the whole team is accountable and shares the responsibility.
6) Foster open communications: Share information among team members and the enterprise.	
7) Empower team members: Empowering team members is important in an ever-changing environment to help create a high performance team.	
8) Invest in quality: Quality should be checked and it should be proactively incorporated into a delivery lifecycle. Quality of the product should be considered important during and after development of a product.	Definition of Done: It is a simple list of activities (writing code, coding comments, unit testing, integration testing, release notes, design documents, etc.) that add verifiable/demonstrable value to the product. Focusing on value-added steps allows the team to focus on what must be completed in order to build software while eliminating wasteful activities that only complicate software development efforts.
9) Learn from all experiences: This should help improve the processes and minimise issues. This should happen at all levels, such as project level, individual level and organisation level.	Sprint retrospective: The sprint retrospective is an important mechanism that allows a team to continuously evolve and improve throughout the life of a project.
	Time boxing: Fixed time period for each planned activity.
	Empirical process control: There is transparency in SCRUM process with an idea of inspection through early feedback from the customer and then adapting to transparency and inspection by bringing improvements.

Microsoft Solutions Framework has a number of roles when compared to SCRUM. Table III shows the mapping of roles between MSF and SCRUM.

TABLE III

MAPPING OF MSF AND SCRUM ROLES

Microsoft Solutions Framework	**SCRUM**
Product Management	Stakeholders
	Product Owner
Program Management	
Testing Release / Operations	
Architecture	Development team
User Experience	
Development	
	SCRUM Master

As seen in Table III, in SCRUM the product owner acts as the representative of the product who performs product and program management and he is also responsible for the quality and operations of the project. The SCRUM team is multi-disciplinary and cross-functional. The skill level varies from developer to tester and user interface designer etc. The SCRUM team will work as a team to finalise the architecture, to test the product, to bring better user experience. In MSF, activities are handled by separate roles. The analysis on mapping of roles within MSF, rational unified process [17], SCRUM and XP made by the authors is published in [18].

Table IV shows the mapping between MSF and SCRUM process. Almost all of the high level processes of MSF can be mapped with SCRUM processes but have an additional process when compared to MSF. It is the review and retrospect process. This process takes place during the end of every sprint and it is actually a meeting where SCRUM teams, product owner and stakeholders are present and the sprint is reviewed. During the meeting, the events that went wrong or well, as well as possible improvements are discussed and then the actions are taken to improve the upcoming sprints.

TABLE IV

MAPPING OF MSF AND SCRUM PHASES AND ACTIVITIES

MSF	**SCRUM**
Envision	Initiate
Plan	Plan and Estimate
Develop	Implement
Stabilise	Implement (Part of Implementation Process)
(No near equivalent process)	Review and retrospect
Deploy	Release

In the MSF, most of the project management responsibilities are covered by the role of program manager, which owns all of the project management area of the whole project. MSF is a highly scalable framework, so it supports the team of any size from small, large to complex. In larger projects, management occurs at multiple levels. In complex projects, a project manager or a project management team will be required. Since MSF is a highly scalable framework when dealing with large or complex projects, the project management can be implemented in two ways, feature and function team. In the case of the feature team, there is a team leader who performs the project management duties and in the function team there are multidisciplinary sub-teams and each team has a project management role.

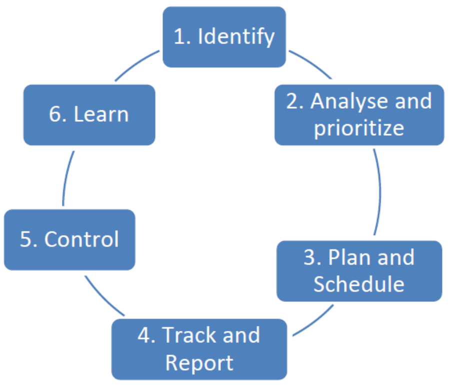
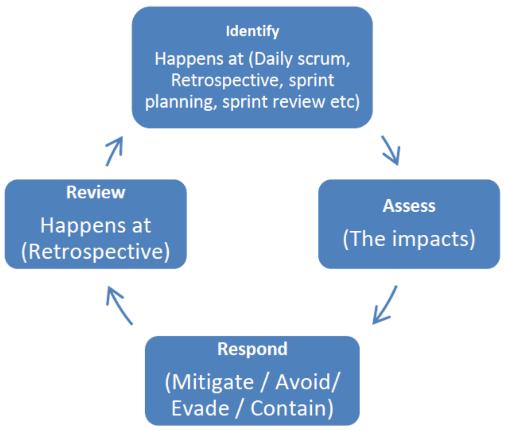
SCRUM is an agile way of managing software project. Surprisingly, there is no role as a project manager in SCRUM, but a project manager can be at the organisation level or from stakeholders. SCRUM team is supported by a SCRUM master and product owner. A SCRUM master or a product owner is not a project manager and the team does not need to report to them. SCRUM team is self-managed and the project is managed by the team. This management occurs through daily SCRUM meetings, sprint planning, sprint review and sprint retrospective. SCRUM project management is different from traditional project management. There should be made some adjustments in activities, artefacts and the roles within the project team. All the above-mentioned will work for small and co-located teams but in the case of large projects with backlogs and remote or distributed teams visibility is very important, so many organisations mostly introduce some software tools to centrally manage projects and this allows for distributed team collaboration. Table V shows the comparison results according to risk management in both software development processes.

Finally, in order to make a choice between methodologies for software development based on existing real projects, usually the ability of addressing the key issues by the methodology is important. For the purposes of qualitative comparison of SCRUM and MSF methodology, the following critical questions are listed and can be quantified in the range between 0 and 1 for each methodology:

1. Are the processes of methodologies clearly defined?
2. Does the client have a quick return on investment (ROI)?
3. Is it possible to change requirements for the duration of iteration?
4. Is it possible to change requirements between sprints or iteration?
5. Is documentation clearly defined?
6. Does duration of the iteration (sprint) have specified time?
7. Are roles clearly defined?
8. Are responsibilities of the roles clearly defined in small projects?
9. Are responsibilities of the roles clearly defined in large projects?
10. Does the development team work only on development tasks?
11. How is the project implemented and how does it impact team members?
12. Is project manager’s role defined?
13. Who defines architecture solutions?
14. Who does the testing?
15. Who define, monitor and manage risks?
16. Who assigns tasks to team members?
17. Are members of development team defined by specialisations?
18. Who will work on release management and deployment?
19. Who will work on creating software builds?
20. How many hours does the team work in full composition during the duration of iteration?

TABLE V

COMPARISON OF MICROSOFT SOLUTIONS FRAMEWORK TO SCRUM BY RISK MANAGEMENT

Microsoft Solutions Framework	**SCRUM**
There is a discipline in MSF to deal with risks in the project. MSF has the process of continuous identification of risk in the project. These risks will be prioritised and sent through a six-stage process to detonate them. In MSF, some strategies are implemented to deal with the risks throughout the project. MSF risk management process is as follows and is self-explanatory:  1. Identify 2. Analyse and prioritize 3. Plan and Schedule 4. Track and Report 5. Control 6. Learn	There is no specific discipline for SCRUM to deal with risks but like in MSF it is a continuous process in SCRUM. In daily meetings and almost all processes of SCRUM and mainly in review and retrospective meeting, the risks, the things which went bad, the things which need to be improved will be added to backlogs and dealt. This way, the risks are solved. The following schema shows the way the risks are managed in SCRUM:  Identify Happens at (Daily scrum, Retrospective, sprint planning, sprint review etc) Assess (The impacts) Respond (Mitigate / Avoid/ Evade / Contain) Review Happens at (Retrospective)

IV. The Proposed Hybrid Model

Figure 7 shows an interactive view of the proposed Hybrid Agile Framework project lifecycle. The Hybrid Agile Framework takes SCRUM construction lifecycle (shown at the top of Figure 7) and extends it to show the full delivery lifecycle from the beginning of the project to the release of the solution to production. The Hybrid Agile Framework lifecycle is organised into three distinct phases with explicit milestones and it is shown in the context of Solution Planning and Application Management.

The Hybrid Agile Framework strives to provide sufficient guidance for consultants to understand the process framework without being overly prescriptive. Therefore, the Hybrid Agile Framework is goal-driven. A goal-driven, suggestive approach provides just enough guidance for delivery teams and is flexible so that teams can tailor the process to fit in the situation they find themselves. The mainstream agile mantra is that Agile Software Development is iterative. The Hybrid Agile Framework recognises that from the point of view of the development team's daily rhythm, the work proceeds iteratively. Each day the development team is likely to iterate back and forth between modelling, testing, coding, and management activities, but the release rhythm proceeds through different project phases. In the beginning, you focus on initiation or start-up activities, in the middle you focus on construction activities, and in the end you focus on deployment activities.

The Hybrid Agile Framework addresses the project lifecycle from the point of initiating the project through construction to the point of releasing the solution into production (shown at the bottom of Figure 7). You will notice each Sprint is not the same. Projects evolve and the work emphasis changes as you move through the lifecycle. To make this clear, the project is partitioned into phases with light-weight milestones to ensure you are focused on the right things at the right time, such as initial visioning, architectural modelling, risk management, and deployment planning.

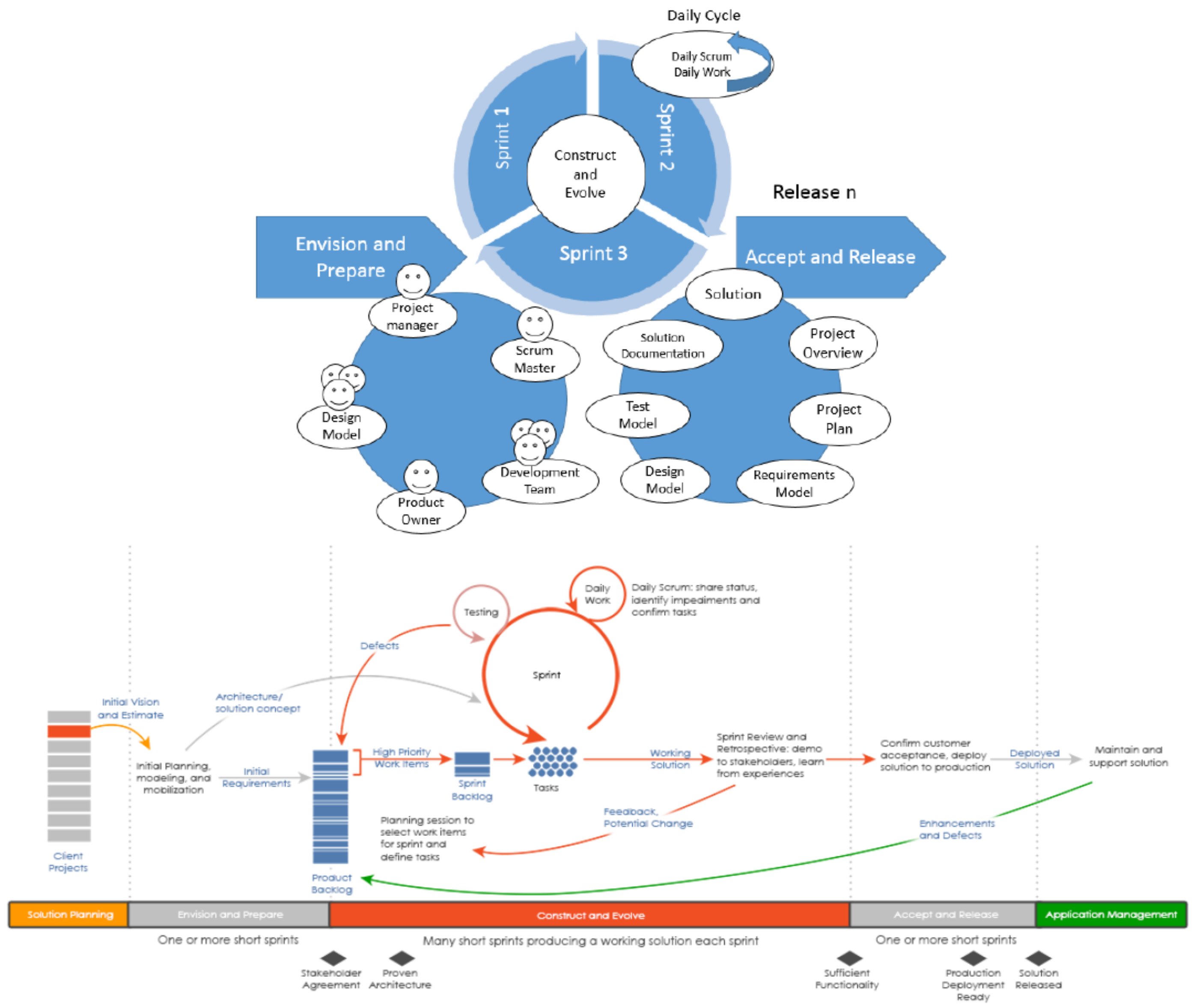


Fig. 7. Hybrid Agile Framework.

The Hybrid Agile Framework lifecycle consists of the following phases:

Solution Planning – a core activity during the sales process. The purpose of Solution Planning is to define the solution, and delivery approach will provide to the client. This includes confirming scope, assessing capabilities and constraints, confirming the architecture / solution concept, defining the solution strategy, and defining the delivery mobilisation approach.

Envision and Prepare – the goal is to identify the stakeholders and their success criteria, develop or confirm the project vision and assess whether the vision is technically feasible. It is necessary to understand the high-level cost estimate, schedule, and risks associated with the project. Stakeholders' agreement around the vision is obtained, the initial development team is mobilised, and the project environments are prepared. If Solution Planning is conducted properly, most of the envisioning work is already completed. In this case, the phase of Envision and Prepare mainly focuses on mobilising the initial development team and preparing the project environments.

Construct and Evolve – the goal is to establish the baseline of the solution architecture and then complete the development of the solution based upon the baseline architecture. The focus of the initial Sprints in the Construct and Evolve phase is to prove the architecture. The remaining functionality is then developed in subsequent Sprints until the features for the current release are complete.

Accept and Release – the goal is to confirm client acceptance of the solution, prepare for a smooth deployment of the solution to production, and then deploy the solution to production, rolling it out to the end users.

Application Management – once the solution is deployed to production, it is transitioned to the support organisation responsible for operating and maintaining the solution.

From a development perspective, each Sprint provides an increment of functionality to the product. The end of each Sprint corresponds to a checkpoint where the development team demonstrates to stakeholders that the objectives of the Sprint are met. From a management perspective, the software lifecycle is decomposed over time into three sequential phases, each concluded by a major milestone.

These milestones provide evaluation criteria at the end of each phase. Each phase is a span of time between two major milestones and has specific focus and objectives. At the end of each phase, assessment is performed to determine whether the objectives of the phase have been met. A satisfactory assessment allows the project to move to the next phase. When a milestone is not met, more Sprints may be performed in the current phase until the milestone is considered complete. Achieving a milestone represents objective criteria used to measure progress.

There are two additional milestones within the phases. The proven architecture milestone is reached early in the phase of Construct and Evolve, normally within one or two Sprints. At this milestone, the stakeholders agree that the architecture is stable and sufficient to satisfy the requirements and the architecture has been prototyped where appropriate to address major architectural risks. The deployment ready milestone is reached late in the phase of Validate and Deploy. At this milestone, all end user / support documentation is created and all end users are trained and ready to use a new solution. The support / operations organisation is ready to support the solution, it has formally accepted it, and the system is ready to be deployed and rolled out to the end users.

The Hybrid Agile Framework embraces SCRUM at its core. It is an iterative, incremental process. The framework structures software development in cycles of work called Sprints, iterations of work which are typically 1–4 weeks long, and take place one after the other without pause. Sprints are time boxed – they end on a specific date whether the work has been completed or not, and are never extended. At the beginning of each Sprint, a cross-functional development team selects items from a prioritised list of requirements, and commits to complete them by the end of Sprint. Each workday, the development team inspects its progress, and adjusts the next steps needed to complete the work remaining. At the end of Sprint, the SCRUM team reviews the Sprint with Stakeholders, and demonstrates what it has built. The SCRUM team obtains feedback that can be incorporated in the next Sprint. The Hybrid Agile Framework emphasises having a working solution at the end of the Sprint that is really "done"; in the case of software, this means a code that is integrated, fully tested and useable.

V. Conclusion And Future Research

Clear responsibilities and shared responsibilities, empowerment of team members, focusing on business values, shared vision of the project, agility, change management, fostering open communication, learning from all the experiences and investing in quality might be considered agile principles methodology.

However, these principles are fundamental principles of Microsoft Solutions Framework and they are not different from the principles behind the agile manifesto. In the present paper, the comparison of SCRUM as one of the most common agile methodologies and MSF as a representative of traditional methodology has been performed. Considering the high coefficient of the two methodologies which can be obtained by quantifying the key issues, we can conclude that SCRUM methodology does not have primacy over MSF in terms of development of software solutions.

Theory of MSF is well structured and defined, while for SCRUM there are a lot of different opinions, contradictions and inconsistencies. In real projects, methodology is still seen as a set of recommendations (framework), and the organisation of work on the development of software solutions should be adapted to the specificities of business system and the team that will implement the project.

Microsoft Solutions Framework is a disciplined software development approach with proven and best practices, adaptable guidelines and processes, as well as it follows iterative methodology. On the other hand, SCRUM is an iterative and incremental form of agile software development and it focuses on providing value to a customer in a short period of time. Taking into consideration advantages of both methodologies, the authors propose the idea of hybrid methodology, which combines discipline and phase division borrowed from MSF and usage of sprints and minimal documentation borrowed from SCRUM.

The role aspect is beyond the scope of this paper, so in future the authors are planning to devote attention to this aspect of software development process organisation.

References

[1] *Microsoft Solutions Framework* [Online]. Available: https://msdn.microsoft.com/en-us/library/jj161047(v=vs.120).aspx

[2] SCRUM Alliance, *What is SCRUM? An Agile Framework for Completing Complex Projects*. Accessed: February 24, 2016.

[3] I. Sommerville, *Software Engineering*, 10th ed., Pearson Education, 2015.

[4] W. W. Royce, “Managing the Development of Large Software Systems,” in Proceedings of IEEE WESCON, pp. 1–9, 1970.

[5] J. Crinnion, *Evolutionary Systems Development, a practical guide to the use of prototyping within a structured systems methodology*, New York: Plenum Press, 1991.

[6] R. Pressman, *Software Engineering: A Practitioner's Approach*, Boston: McGraw Hill, 2010.

[7] K. Forsberg and H. Mooz, “7.17. System Engineering for Faster, Cheaper, Better,” *INCOSE International Symposium*, vol. 8, no. 1, pp. 917–927, Jul. 1998. https://doi.org/10.1002/j.2334-5837.1998.tb00130.x

[8] B. Boehm, “A spiral model of software development and enhancement,” *ACM SIGSOFT Software Engineering Notes*, vol. 11, no. 4, pp. 14–24, Aug. 1986. https://doi.org/10.1145/12944.12948

[9] “Manifesto for Agile Software Development,” Agile Alliance, 2001.

[10] *Software Development Models* [Online]. Available: http://istqbexamcertification.com/what-are-the-software-development-models/

[11] K. Beck, *Extreme Programming Explained: Embrace Change*, Addison-Wesley, 1999. ISBN: 978-0-321-27865-4.

[12] J. Martin, *Rapid Application Development*, Macmillan Publishing Company, 1999.

[13] The Standish Group International, *CHAOS Manifesto: The Laws of CHAOS and the CHAOS 100 Best PM Practices*, The Standish Group International, 2011.

[14] D. West and J. S. Hammond, *The Forrester Wave: Agile Development Management Tools, Q2 2010*, May 2010.

[15] VersionOne, *5th Annual Survey: 2010 - “The State of Agile Development”*, October 2010.

[16] K. Cathey, *Software Engineering Process, Apple Inc*. [Online]. Available: https://www-s.acm.illinois.edu/iCal/workshops/general/Software-Engineering-Slides.pdf

[17] P. Kruchten, *The Rational Unified Process: An Introduction*, Addison-Wesley, 2004.

[18] O. Nikiforova, V. Nikulsins and U. Sukovskis, “Integration of MDA framework into the model of traditional software development,” *Frontiers in Artificial Intelligence and Applications*, vol. 187, issue 1, 2009, pp. 229-239. https://doi.org/10.3233/978-1-58603-939-4-229

Jai Vigneshwar Alavandhar received the Bachelor’s degree in information technology from Bharath University, India, in 2011.
He received Master’s degree in applied computer systems from Riga Technical University in 2016.
Currently, he works as an Analyst Programmer at KPMG Crimsonwing Ltd, Malta.
His current research interests include agile software development.
E-mail: jaivigneshwar@gmail.com

Oksana Ņikiforova received the Doctoral degree in information technologies (system analysis, modelling and design) from Riga Technical University, Latvia, in 2001.
She is a Professor at the Department of Applied Computer Science, Riga Technical University. Her current research interests include object-oriented system analysis, design and modelling, especially the issues in model-driven software development.
E-mail: oksana.nikiforova@rtu.lv