Scrum Solo Application in a Project with a Strong Integration Component

Joao N. BRITO
Primavera BSS, Braga, Portugal, jnabrito@gmail.com

Carlos REBELO
University of Minho, Braga, Portugal, carlosrebelo-7@hotmail.com

Miguel A. BRITO
Centro Algoritmi - University of Minho, Braga, Portugal, mab@dsi.uminho.pt

Abstract

In Portugal, 90% of software developing companies are micro companies. In 2002, in the Brazilian market, about 60% of the software developing companies would start their activities with just a single developer (Pagotto, Fabrti, Lerario, and Gonçalves, 2016). The Scrum Solo methodology was developed having in mind, that there is a big need for organizing and managing the software development of teams comprised by a single individual. The underlying complexity of big systems and the underlying integration complexity makes the usage of techniques and processes that guarantee the control of the project vital. The methodology described in this document is validated with two application cases in a context of development. The first one is an extensive project, with strong integration component, and the second one the development of an application for process dematerialization. The Scrum Solo methodology performed well in both cases despite their differences. Nevertheless, more cases should be analyzed with the emphasis in using different contexts namely other organizations.

Keywords: Scrum Solo, Scrum, software development process.

Introduction

In Portugal, 90% of software development companies are characterized as micro-enterprises ("Lista de Empresas de Desenvolvimento de Software em Portugal", 2018). In the Brazilian market, in 2012, about 60% of software companies started their activities with only one developer (Pagotto, Fabrti, Lerario, and Gonçalves, 2016).

Agile methodologies are widely used by many teams nowadays. Among these, Scrum is the most widely practiced Agile method/framework (Srivastava et al, 2017). From the respondents to the survey referenced in the 14th Annual State of Agile Report (2020), at least 75% of respondents practice Scrum or a hybrid that includes Scrum. However, as stated before, many of those teams, are a single person team. This was the main motivation to adapt the Scrum methodology to solo teams, thus Scrum Solo.

In this article, Scrum Solo is described, explaining the main activities, the actors involved, and the artifacts resulting from the process. Furthermore, two validation cases were developed and are also described in this paper.

These two projects where developed over a period of roughly six months each, for one of the biggest Portuguese's software development company. The first one's goal was to create a tool that generates code from a REST Web service (REpresentational State Transfer) automatically, based on the existing code of an ERP system (Enterprise Resource Planning), this way, removing the need of having 2 big teams in parallel, developing both the ERP, and the Web API solution. The second one's goal was to develop a mobile application prototype, which integrates with the aforementioned ERP through the Web API, contributing to the dematerialization of processes such as a salesperson writing down sales details on paper, and typing it afterwards at the office.

This cross-platform mobile application targets sellers which are away from the company and cannot access the ERP directly, adding mobility and flexibility. With the conclusion of this prototype, they were able to access, create, edit, and delete different entities such as items, documents, suppliers, and customers, saving them the time of, for example, having to write everything in paper and typing it afterwards at the office.

Scrum Solo Methodological Approach

Scrum Solo is an adaptation of Scrum and PSP (Personal Software Process) for solo software development. Based on figure 1, it is possible to see the similarities with Scrum (Pagotto, Fabrti, Lerario, and Gonçalves, 2016).

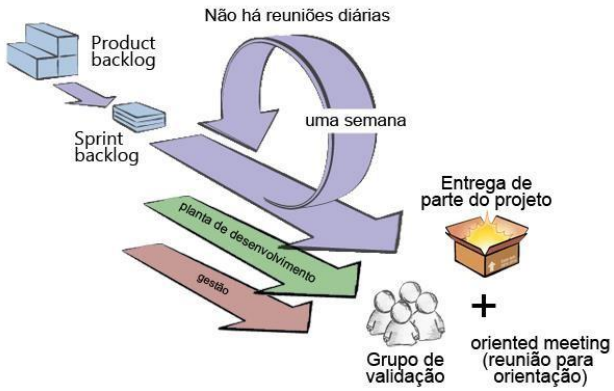

Fig 1. Scrum Solo (Pagotto, Fabrti, Lerario, and Gonçalves, 2016)

Scrum's goal is to provide a convenient process for project development, presenting an empirical approach, which applies ideas based on the theory of industrial process control to software development, introducing additional ideas such as flexibility, adaptability and productivity (Soares, 2007).

Scrum was designed to allow normal developers to organize and form high development teams (Sutherland, Viktorov, Blount, and Puntikov, 2007). Scrum assumes that the systems development process is complicated and unpredictable, which can only be described as a general progression (Schwaber, 1997).

The workflow begins with the creation of a product backlog, which contains all the tasks necessary to implement the features desired by the customer. From this group of tasks, some are selected to be developed during the sprint (within a week). This sprinting process is iterative until the project's completion date arrives, or until all tasks are completed. During sprints, it is necessary to have management activities, which aim to plan and monitor the development of the project. After completing the tasks, the project is delivered, which will be reviewed by an evaluation group and, eventually, an orientation meeting will be held with the people who will use the developed product (figure 1).

This methodological approach has four main activities: definition of requirements, sprint, deployment, and management. The definition of requirements (figure 2) aims to define the scope of the product, the product backlog and, if possible, a software prototype. These objectives will be achieved with the help of customer input and advisor that will assist the developer to achieve his goal.

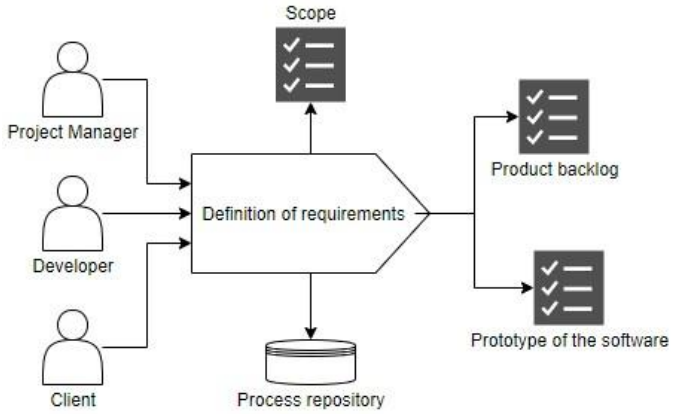

Fig 2. Definition of requirements (Adaptation from Pagotto, Fabri, Lerario, and Gonçalves, 2016)

Sprints, represented in figure 3, are the activity where the development of a set of tasks selected at the beginning of the same, from the product backlog, occurs. These tasks must be completed within one week, resulting in a sprint backlog, part of the product, development plan, and minutes.

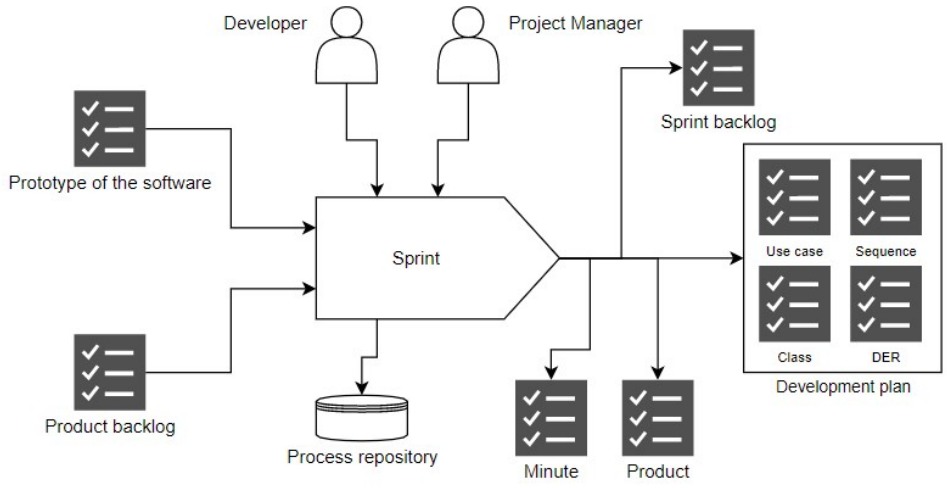

Fig 3. Sprint (Adaptation from Pagotto, Fabrti, Lerario, and Gonçalves, 2016)

During the sprints it is necessary to manage the status of the project. The project management activity has this objective. Using the product backlog, the developer, advisor, and client must discuss the status of the project. From this reflection, the project's analytical structure (WBS), project schedule and cost table will be generated.

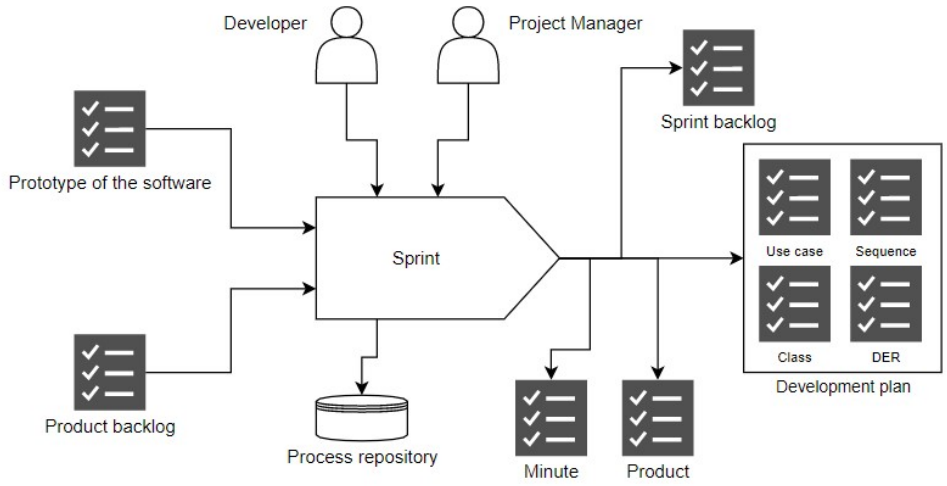

Fig 4. Project management (Adaptation from Pagotto, Fabrti, Lerario, and Gonçalves, 2016)

Finally, as a last task, deployment is the activity in which the product is made available for use by the customer. It is also important, after delivery, to have an orientation meeting so that those involved understand in more detail how the product works.

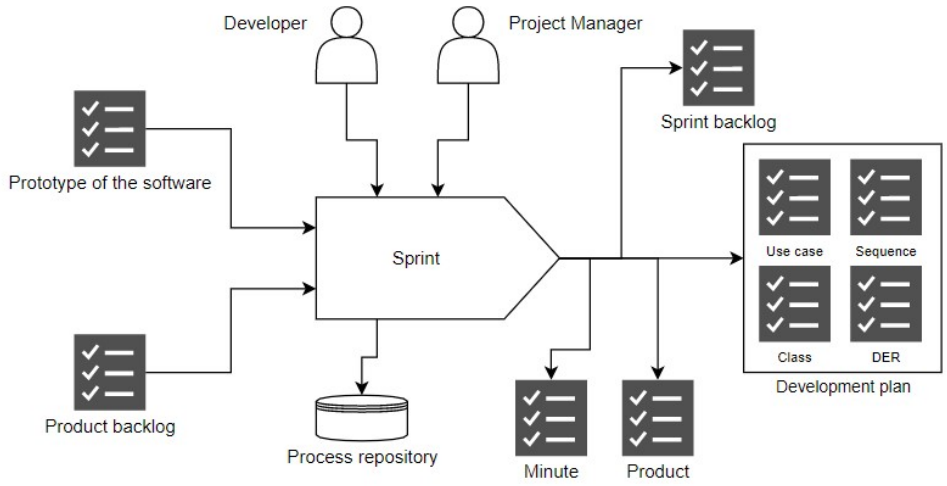

Fig 5. Deployment (Adaptation from Pagotto, Fabrti, Lerario, and Gonçalves, 2016)

There are several actors in this process, namely the developer, who is responsible for developing the product and following the methodology; the client (product owner), who will be the owner of the final product, can be made up of one or more individuals who will interact with the product (example: managers, accountants, others ...); the advisor, who is responsible for helping to guide the developer throughout the process, so that the requirements are well met; and, finally, the validation group, made up of the end users of the product, who aim to evaluate the product and provide feedback.

The various artifacts generated throughout the process are as follows:

Scope: it serves to characterize the scope of the process and the aspects inherent in mapping the problems of the product owner, describing the scope of the product, the customer profile, and the functional requirements.

- Software prototype: consists of images of the user interface with the product. It is advisable to use the ".png" format and point the name of the product backlog item that represents the image.

- Product backlog: is a list of features that must be implemented in the software. Each line in this list must contain a feature code, description, insertion date and selection date for the sprint backlog.

- Process repository: it serves to store all documents related to the entire product development process in the cloud. It is important that the developer organizes the repository folders well and follow a set of rules for that purpose.

- Sprint backlog: it is a list of all the features that must be implemented in a given sprint. The sprint backlog must also store the feature code.

- Product or part in operation: a version of the product that gives the customer the possibility of obtaining a return on the investment previously made at the time he asked for its development.

- Minute: it is used to record the moments when a feature is implemented. Used in the sprint and delivery: in the sprint, the functionality is validated by the advisor; delivery is validated by the customer.

- Development plan: aims to gather the artifacts used in the specification of the functionalities. Scrum Solo suggests the use of diagrams of use cases, sequence, classes, and entity and relationship, always with the possibility of using others, if necessary, for the case.

- Analytical structure of the project: a hierarchical structure that subdivides the work of a project into smaller and more readable components.

- Chronogram: aims to organize work packages sequentially within a certain time frame. The person responsible for the execution of each activity can be appointed. It is suggested to use the schedule in the format of a Gantt diagram.

- Cost table: aims to map the effective cost generated during the execution of the project to be compared with the budget previously made.

A search was also made for “Scrum Solo” through the Google academic platform, however, the state of the art is still underdeveloped. Fonseca (2016) only generically describes what Scrum Solo is as a “methodology for dynamic project management that also allows the development of software in an agile way”. Similarly, Sanches Kleim (2016) describes that Scrum Solo was developed with the scope of helping with the organization of software development by individuals, based on the good practices described by PSP and Scrum.

Application of Scrum Solo in an Extensive Project, with Strong Integration Component

The Scrum Solo methodology consists of an iterative and incremental process, which unites the best practices defined by the Personal Software Process (PSP) and Scrum, to promote the smooth functioning of the development of a project. This methodology was applied to the development of a project whose objective was to be able to create a tool that generates code from a REST Web service (REpresentational State Transfer) automatically, based on the existing code of an ERP system.

This project allows users to access the ERP through a Web service and, eventually, through a dedicated website, which consumes the service. However, the value of this project lies in the fact that the conversion of the ERP code (developed over 25 years, by multiple teams, with about 4.5 million lines of code today) is done automatically, saving an enormous amount of resources in its creation and maintenance, since the ERP is constantly in development, the Web service would have to monitor, adding a level of complexity in the coordination of the ERP and Web service development teams.

In projects of this magnitude, there are added risks associated with work, which do not exist in traditional software development projects. The identified risks involve limiting access to

the original ERP code and its complexity; code incompatibility between the ERP and Web service (the use of certain mechanisms for the use of assemblies by the ERP, caused a lot of problems due to the incompatibility of these mechanisms with the Web, resulting in the need to change the original ERP code); there were also problems in working with the version of ERP that was under development, sometimes bugs were found, resulting in time spent looking for the source of the problem; finally, the need to change the machine also proved to be a setback, due to the dimension of the project and the configuration of several components to support development.

The Scrum Solo methodology, although it has a project management task, does not respond well to these cases of high dependence on external factors, in this case, the ERP's underlying services. It is proposed to create a risk mitigation task, equivalent to a low percentage of the time spent per sprint, which can be replaced by a development task if time is available. This mitigation task can, for example, be added at the end of all sprints, costing two to four hours.

Scrum Solo goes through an initial moment of defining requirements, followed by sprints where the program is developed, followed by the deployment of the program to the client. During the development process it is also necessary to have project management activities. Next, the tasks developed for the particular case, previously mentioned, will be exposed.

During the definition of requirements (figure 2), several meetings were held with the client and advisor, to obtain an understanding of the necessary requirements of the application. This process resulted in a document called scope, which contains the definition of the scope of the problem and the customer's profile; and a document called product backlog, with 32 activities that, after completion, resulted in a product that will be delivered to the customer.

The Web API has the peculiarity of not having a traditional user interface. Instead, its interface is done through URL's ("{Domain}/api/{class}/{method}/{parameters}"). In other words, the software prototype is made up of this generic standard for the interface, which will then be used during code generation.

The process repository is divided into two parts. The artifacts generated regarding the use of Scrum Solo were filed in a folder in the cloud, using Google Drive services. The code developed, was saved on the client's local network, and was managed through the TFS tool (Team Foundation Server).

It was defined that the sprints (figure 3) would consist of five working days of development, however, a sprint does not necessarily represent a week, due to the interruption of development for external reasons, such as writing scientific articles, holidays, events, among others. This aspect brings a uniformity that will have an impact on the representation of the sprints, since everyone has the same amount of working time.

It is understood that, according to the work Pagotto et al. (2016), after a sprint, a development plan is supposed to be generated. In this case, the Web API already respects all the business rules imposed by the ERP. This fact comes from the exclusive use of the ERP business layer in the Web API, that is, all business rules will be respected, just like in the ERP, making the elaboration of the artifact unnecessary.

As for the product, new versions were saved on TFS regularly, allowing access to the code by authorized persons, and having access to the records of all changes made.

The sprint backlog aims to identify which product backlog tasks were executed in a given sprint, its start date, and the time spent on its completion. Although it is important for the future to have this artifact well documented and readable by people other than the developer, there is no need for complex tools for its management, a simple sheet of paper or a word document is sufficient (figure 6).

The development of the project was completed in June 2018, with the deployment activity (figure 5) already completed.

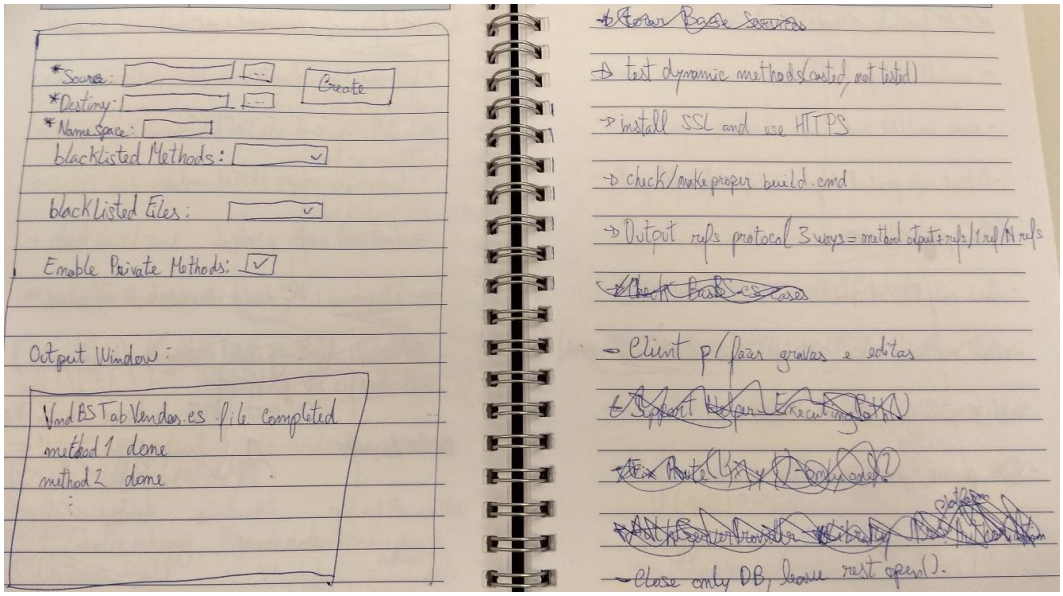

Fig 6. Example of a sprint backlog

Application of Scrum Solo in a Project, with Strong Component of Process Dematerialization

The Scrum Solo methodology was also used in another project, which aimed to achieve a process dematerialization of already existing processes. The solution was to develop a mobile application prototype, which integrates with the ERP mentioned in the previous example.

This cross-platform mobile application targets sellers which are away from the company and cannot access the ERP directly, adding mobility and flexibility. With the conclusion of this prototype, they were able to access, create, edit, and delete different entities such as items, documents, suppliers, and customers, saving them the time of, for example, having to write everything in paper and typing it afterwards at the office.

The major benefits are the highly performant load times and the ability to access all the above-mentioned features even offline (keeping in mind that some information might be outdated because it has not been synchronized).

With any project, comes different difficulties. In this case, the main one with the ERP and its Web API was the incomplete/undetailed documentation, which lead to adversities while trying to integrate different information and setting up a test server for deployment testing. With these issues, the project activities were adapted accordingly, and they were reported to the product team which made many improvements based on the inputs given.

As a side benefit, being one of the first projects using the ERP's Web API, a series of optimizations and quality of life updates were done in parallel with the development of this project. With Scrum Solo, it was easy to adapt to the constant changes to the Web API.

To carry out this project, the risk mitigation task referred to in the previous project was followed, being implemented in order to monitor the progress of the project, serving to carry out a retrospective of the prototype in relation to the scope. This task was divided into two distinct parts. The first part was dedicated to the identification of possible features that could be explored through those already carried out in the previous sprint, enhancing the responsiveness of the prototype in relation to the respective application processes.

One of the main goals of the development, was to use concept of Rich Internet Applications (RIA). RIAs combine the Web's lightweight distribution architecture with desktop applications' interface interactivity and computation power, and the resulting combination improves all the elements of a Web application (data, business logic, communication, and presentation) (Fraternali, Piero, Rossi, and Sánchez-Figueroa, 2010).

In this case, the product backlog was constructed by further detailing an already existing document, provided by the company. The sprints were done weekly, independently of having one or five workdays. Despite this fluctuation in length, the methodology was able to be customized to fit the project management needs.

At the end of each sprint, a functional prototype was presented, containing all the new and previous features developed, maintaining a complete artifact that allows viewing the prototype's state in relation to its final target.

Conclusion

After describing the main activities, their actors and artefacts resulting from the process, and the use of the Scrum Solo methodology, it is conclusive that a good adaptation was really made from the methodologies on which it was based. However, as the name "methodology" indicates, Scrum Solo is not exempt from adaptations to the specific cases of each individual and each project, such as setting the sprints as one week (regardless of having one or five workdays), or setting the sprints as five days, even though they might not be consecutive.

The number of scientific articles on Scrum Solo is still scarce, in addition to the work of Pagotto, Fabrti, Lerario, and Gonçalves (2010), the few references found are about specific applications of the methodology and were described in section 2.

Regarding the application of this methodology to cases described, the success obtained is notable. It was possible to organize and manage two projects carried out separately, lasting half a year

each from start to finish without jeopardizing the delivery date or any requirement requested by the organization, despite some unexpected problems from the ERP's underlying services, which were fixed after giving feedback.

Finally, it is proposed to create a risk mitigation task, equivalent to a low percentage of the time spent per sprint, which can be replaced by a development task if time is available. This mitigation task can, for example, be added at the end of all sprints, costing two to four hours.

Acknowledgements

This work has been supported by FCT – Fundação para a Ciência e Tecnologia within the R&D Units Project Scope: UIDB/00319/2020 "This work has been supported by FCT – Fundação para a Ciência e Tecnologia within the R&D Units Project Scope: UIDB/00319/2020"

References

14th Annual State of Agile Report. [Online], [Retrieved August 16, 2020], https://explore.digital.ai/state-of-agile/14th-annual-state-of-agile-report

Fonseca, J. W. (2016). A informatização na coleta de dados para controle de doenças epidemiológicas: um estudo de caso com a Dengue.

Fraternali, P., Rossi, G., and Sánchez-Figueroa, F. (2010). Rich internet applications. IEEE Internet Computing, 14(3), 9-12.

Lista de Empresas de desenvolvimento de software em Portugal. [Online], [Retrieved May 10, 2018], www.empresite.jornaldenegocios.pt/Actividade/SOFTWARE

Pagotto, T., Fabri, J. A., Lerario, A. and Gonçalves, J. A., 'Scrum Solo: Software process for individual development,' 2016 11th Iberian Conference on Information Systems and Technologies (CISTI), Las Palmas, 2016, pp. 1-6, doi: 10.1109/CISTI.2016.7521555.

Sanches Kleim, E. C. (2016). Um estudo do Scrum em um projeto com uma equipe pequena.

Schwaber K. (1997) SCRUM Development Process. In: Sutherland J., Casanave C., Miller J., Patel P., Hollowell G. (eds) Business Object Design and Implementation. Springer, London. https://doi.org/10.1007/978-1-4471-0947-1_11

Soares, M. (2004). 'Metodologias ágeis extreme programming e scrum para o desenvolvimento de software,' *Revista Eletrônica de Sistemas de Informação*, 3(1), ISSN 1677- 3071 doi: 10.21529/RESI.

Srivastava, A., Bhardwaj, S. and Saraswat S. (2017) 'SCRUM model for agile methodology,' International Conference on Computing, Communication and Automation (ICCCA), Greater Noida, 2017, pp. 864-869, doi: 10.1109/CCAA.2017.8229928.

Sutherland, J., Viktorov, A., Blount, J. and Puntikov, N., 'Distributed Scrum: Agile Project Management with Outsourced Development Teams', 2007 40th Annual Hawaii International

Conference on System Sciences (HICSS'07), Waikoloa, HI, 2007, pp. 274a-274a, doi: 10.1109/HICSS.2007.180.