Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 164 (2019) 491–497

Procedia Computer Science

www.elsevier.com/locate/procedia

CENTERIS - International Conference on ENTERprise Information Systems / ProjMAN - International Conference on Project MANagement / HCist - International Conference on Health and Social Care Information Systems and Technologies

Proposal of a Visual Environment to Support Scrum

Fidel Kussunga[a,*], Pedro Ribeiro[b]

[a]*Department of Industrial Electronics, University of Minho, Campos de Azurém, Guimarães 4800 – 058, Portugal*
[b]*Information Systems Department, University of Minho, Campos de Azurém, Guimarães 4800 – 058, Portugal*

Abstract

Agile methodologies are increasingly considered important in the current context because of their focus on business benefit, strong stakeholder involvement and rapid incorporation of changing requirements. The Scrum methodology presents a generic and agile model for software development, but there are some gaps regarding solution architecture and requirements modeling and traceability. Fast customer feedback and support for volatile requirements result in higher product value, however, initial requirements modelling using modelling techniques are not commonly used in agile processes such as Scum to prepare the phase of Implementation of the software project.
This article presents a first proposal for a visual environment to support the product / sprint backlog, helping to prioritize, track and decide on the most valuable, aligned and quickly deployable requirements.

Keywords: Visual environment; Software development; SCRUM; Software modeling; Agile methodologies.

* Corresponding author. Tel.: +351- 912-078-735; fax: +351-969-48-05-58.
E-mail address: a72671@alunos.uminho.pt

1. Introduction

Software systems are becoming increasingly complex due to diverse and changing customer requirements [1]. At the same time, the market is very competitive. Therefore, software teams must develop software products to meet all requirements in a predetermined time and at a defined cost. Traditional methods of software development are no longer adequate. These methods are burdensome processes that pay more attention to industry and standardization. This usually leads to a longer development cycle and an additional cost [1]. The emergence of agile methods in recent years has attracted increasing attention, being used successfully to improve software project processes.

Agile methodologies from the Agile Manifesto [2] are lighter methods for software projects compared to most traditional methods. Agile methodologies feature self-organized teams that are empowered to achieve specific business goals. Agile methodologies focus their attention on fast and frequent deliveries of partial solutions in an iterative and incremental way [1]. Agile methodologies have been proven to deliver high quality products in less time, resulting in greater customer satisfaction.

Scrum is an approach to project development that does not require the completion of the global technical specification, as in the traditional model [3]. Scrum is a methodology for managing the development of information systems, which places a strong emphasis on fast deliveries on the development process. In addition to managing software development projects, the methodology is used by software support teams as well as an approach to managing software development and maintenance [3]. The agile Scrum methodology is based on the division of projects into iterations (Sprints). Each Sprint includes five phases, which are [4]: (1) initialize, (2) analyze, (3) design, (4) perform, and (5) test a set of User Stories. A User Story describes the functionality that will be valuable to the user / client of a system or software.

The UML modeling language contains many different diagrams from object-oriented systems analysis and design and is a standard language in the software development market. This rich modeling tool provides practices at the technical level at every stage of software development. In addition, UML modeling is independent of process models.

An architectural transformation approach, called the Four Step Rule Set (4SRS), has been used and employs successive transformations of software architecture [5] to satisfy the elicited user requirements. The 4SRS technique is essentially based on the mapping of use case diagrams in object diagrams. The iterative nature of the technique and the use of graphic models are important issues to ensure that the final architecture reflects user requirements [5].

The purpose of this article is to present a visual environment approach to support Scrum, using modeling techniques. For this publication, a modeling technique based on UML was adopted because of the strong formalization and its recognition by the scientific community. The approach maps UML modeling technologies to Scrum practices and provides detailed descriptions of how it can be applied to a real project.

2. Overview of the approach

In this section, we present the proposed approach linking it to the Scrum ceremonies (figure 1). The approach includes a block of model transformation, i.e., User Stories in Use Cases and the transformation of Use Cases to the Architectural model (using 4SRS technique).

The life cycle of the visual environment is divided into two main phases, which are Sprint Planning, using the UML modeling technique and Sprint Execution. The following guidelines are provided in the Scrum methodology, the Product Backlog must first be created, which is a list of User Stories (short descriptions of all the functionalities required in the software product), the Product Owner is responsible for prioritizing the Product Backlog [6]. There is few orientations on how to analyze and create a Product Backlog. Whereas, the analysis of software requirements is extremely difficult, and is the secret to successfully start any agile software development [1]. Therefore, in this approach to a visual environment, the Use Case modeling technique / tool was introduced in the agile requirements analysis phase. The Sprint implementation, the project organization, and the Scrum framework template are followed. The second phase are called Sprint Execution with UML modeling.

The first phase consists of Product Backlog (User Stories list), Sprint Planning, Select Product Backlog (Transformation - US-> UC), 4SRS, Global Architecture, Product Backlog (with Use Case Diagrams) and the second phase consists of Sprint Backlog, Sprint and Sprint Review. The description of each of these elements can be found in section 3 of this article.

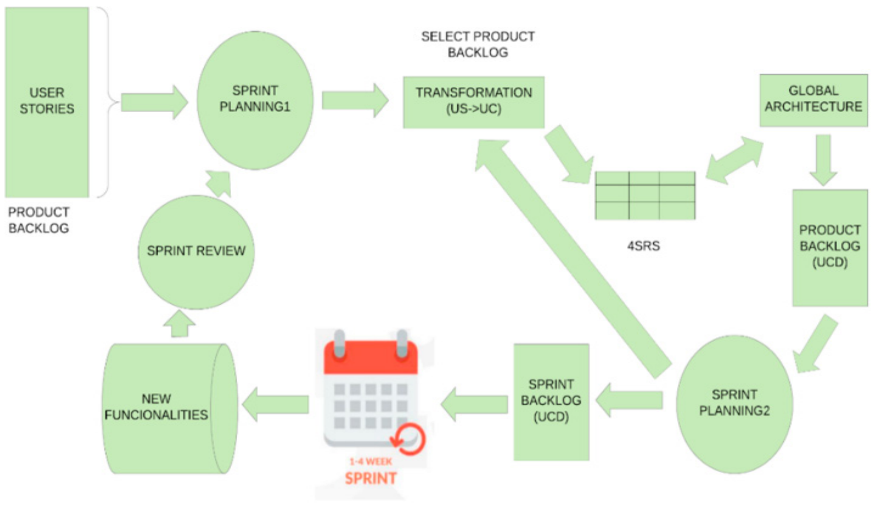

Fig.1. Visual environment approach process.

In Figure 1 the activities and artifacts of these two steps are illustrated. The main artifacts of the agile requirements analysis phase are the Product Backlog and the Use Case models resulting from the User Stories transformation process. The second phase is implemented through a series of iterations called Sprints. Each Sprint deliver a potentially usable product increment [1].

The proposed approach is inspired by Scrum, UML and object-oriented methods, so it is user and architecture focused, as well as iterative and incremental oriented. User oriented development is one of the commonly adopted practices in software development [1], which provides an excellent solution for inaccurate, incomplete, or inconsistent requirements. In this proposal, the user stories and use cases are central. User stories are converted into use cases and use cases are converted into the global architectural model using the 4SRS technique, the details are described in the next section.

Use Case models and User Stories are important in the requirements collection process and although there are no distinct phases (such as gathering requirements phase or design phase) in agile processes, there is a need for efficient and effective requirements collection, since the industry develops software for customers with low level of knowledge of software development [7], [8].

3. Proposed Approach Process

The proposed visual environment to support Scrum ceremonies, including functions, activities, artifacts, and implementation model, will be detailed in this section.

3.1 Sprint Planning

According to the Scrum Guide [6], work to run on Sprint is planned in Sprint Planning. This plan is created involving all Scrum roles. The team prioritizes the elements of the Product Backlog to be implemented and transfer these elements of the Product Backlog to the Sprint Backlog, that is, the list of features to be implemented. In our approach, the plan is created based on the Product Backlog that results from the transformation processes (transformation of User Stories in Use Cases and application of the 4SRS method). Sprint Planning2 is a task of extreme importance for this proposal since it is in this meeting that the details of the requirements are analyzed carefully. The missing details are checked in the global architecture as new Use Cases are created and moved to the next step as soon as everything is set (Sprint Backlog stage), otherwise it returns to the initial block and proceed to adjust the requirements. This process is repeated and ends as soon as satisfactory results are obtained. Although the approach presents two blocks of Sprint Planning, only a single description is made since both blocks have the same function.

3.2 Transformation (US-> UC)

In this block, the Product Backlog is selected (in the form of User Stories) and then converted according to the priority into Use Case diagrams. The selection of requirements at this stage is responsibility of the Product Owner who also provides the User Stories explanation. In the proposed prototype (see section 4) a text file containing the list of Product Backlog items is read and using the PlantUML tool (installed in the Eclipse environment) we indirectly convert (using Epics) User Stories into Use Cases.

3.3 Four Step Rule Set (4SRS)

4SRS is a method that allows the transformation of user requirements into a representation of the architectural model [9] by applying a set of four steps. The 4SRS receives as inputs a set of Use Cases that describe the requirements for the specific process (s) that addresses the initial question, what are the activities performed by people or machines, in the context of the system. Use cases are refined through several iterations of the 4SRS. At the end of the execution of all the steps that constitute the method, we obtain as output, a global logic architecture [9].

Utilization of Use Case diagrams to manage requirements is mandatory, because 4SRS method uses Use Cases as input, to derive the logical architecture [10]. Object Management Group (OMG) defines a Use Case as: "the specification of a set of actions performed by a system that produces an observable result that is typically of value to one or more actors or other system stakeholders"[14]. Use Cases allow you to simply express the system requirements and the connection between the system actors and the functionalities.

3.4 Global Architecture

Global architecture is the output of the application of the 4SRS method. The global architecture in this work is the set of multiple object diagrams that result from the process of transforming Use Cases diagrams into an architectural model after the four steps of the 4SRS methods are recursively executed.

3.5 Product Backlog

The product owner is the person who makes the decisions as to what requirements the product will have. It is the one that prioritizes the requirements and manage the Product Backlog [6], [11], [12]. To do this job, the Product Owner role requires an individual with certain skills and characteristics, including availability, business experience, and communication skills. In this proposal, the process of automatically transforming User Stories into Use Cases and applying the 4SRS method can help reduce the complexity and improve the efficiency of its work (it is intended to implement the automation of the process of extracting class diagrams from Use Cases). More specifically, a Use Case for this proposal is a Product Backlog feature and all the Use Cases diagrams that make up the Global Use Cases model constitute the Product Backlog. The Product Owner function is to prioritize User Stories (User Stories are prioritized before being converted into Use Cases). This approach proposes a Product Backlog different from traditional ones, as it is visually supported by Use Cases.

3.6 Sprint

The goal of each Sprint is to convert User Stories into Code Delivered and Tested. The Sprint Backlog can be maintained as a physical task board on the wall for quick and frequent tracking. During Sprint, the Scrum team divides each User Stories into small units called tasks as needed and estimates how many stories points each task will take the team to complete the task. The team is self-organized and self-managed. All team members contribute in whatever way they can to complete the set of work they have collectively agreed to complete on Sprint.

In the Sprint implementation, UML technologies are applied. Both the static and dynamic characteristics of the system are analyzed and then the coding can be performed correspondingly [1]. Model diagrams can be drawn superficially on the board or even on paper (general case). To maintain agility and programming as fast as possible. For this approach, UML models (Use Cases diagrams and Class diagrams) are used.

3.7 Sprint Review

At the end of each Sprint [6], the team performs a Sprint review, during which new features are demonstrated to the Product Owner or any other stakeholder who wishes to provide feedback. The project is evaluated against the Sprint goal determined during Sprint Planning. This type of feedback loop in software development can result in reviewing or adding items to the Product Backlog, and it's an opportunity to identify anything to improve.

4. Prototype software

A contextual diagram of the architecture of the software prototype is presented below in Figure 2, together with the manual process of transforming User Stories into Epics and the existing process (Eclipse and PlantUML). The prototype model is based on [8].

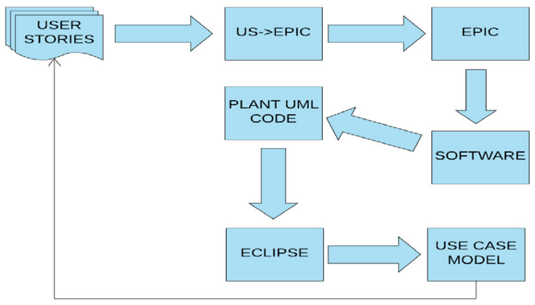

Fig. 2. Context diagram of software prototype architecture.

4.1 EPIC

An Epic is a larger User Story that is too large to be implemented in a single iteration, so it needs to be broken down into smaller User Stories. In this work the use of Epics was adopted due to the direct relation that these present with Use Cases according to the study presented in [7]. Epics are obtained from a set of User Stories in a non-automatic process. User Stories are not covered in this section since they have been discussed previously.

4.2 PlantUML code

This block presents the code of the PlantUML tool generated through developed software. This PlantUML code implemented in an untitled file in the Eclipse environment outputs the diagram representing the desired Use Case model.

PlantUML is a tool that allows users to generate UML diagrams from a simple text language [13]. The language used by PlantUML is called application specific language because it only works for the PlantUML tool. PlantUML itself is an Open Source software, and there are factory UML plug-ins for various common software platforms such as Eclipse, NetBeans, Microsoft Word, LaTeX, etc. [13].

4.3 *Eclips*e

Eclipse is a development environment for Java programs, but it is possible to use Eclipse for development in other programming languages such as C / C ++ from installing plug-ins. In Eclipse you can create a project, perform the encodings, and check execution logs.

4.4 Use Case Model

Use Case Model is the result of transforming Epics into code from the PlantUML tool. This model is the result that we intend to achieve with the development of this software prototype. The following Figure shows the diagram of Use Case automatically generated by the prototype as well as its Epic as a form of validation of the concept of the process of transformation of User Stories in Use Cases.

As an Operator I want to register customers, so customers can be counted.

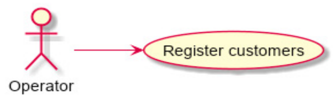

Fig.3. Use Case model generated by the prototype and its respective Epic.

5. Conclusion

This article presents a visual framework approach to support the Scrum agile methodology ceremonies and a prototype required for the generation of User Use Cases from User Stories. The 4SRS method was used to derive a logical architecture, which is then modularized, refined and used as input to the Product Backlog structure, that is, Epics, User Stories, and acceptance criteria. Finally, an Epic and its Use Case model generated by the prototype was presented. It is suggested to follow all the detailed steps to maintain the entire development process in an agile way.

Acknowledgments: This work has been supported by national funds through FCT – Fundação para a Ciência e Tecnologia within the Project Scope: UID/CEC/00319/2019.

References

[1] X. Yaohong and F. Jingtao (2014) "Research on Software Development Process Conjunction of Scrum and UML Modeling,".

[2] K. Beck *et al.* (2001) "Manifesto for Agile Software Development Twelve Principles of Agile Software,".

[3] J. Doronina and E. Doronina (2018) "MODELS OF IT-PROJECT MANAGEMENT," *Int. J. Comput. Sci. Inf. Technol.*, vol. 10, no. 5.

[4] M. Elallaoui, K. Nafil, and R. Touahni (2018) "Automatic Transformation of User Stories into UML Use Case Diagrams using NLP Techniques," *Procedia Comput. Sci.*, vol. 130, pp. 42–49.

[5] N. Santos, J. Pereira, F. Morais, J. Barros, N. Ferreira, and R. J. Machado (2018) "An agile modeling oriented process for logical architecture design," in *Lecture Notes in Business Information Processing*, vol. 318, pp. 260–275.

[6] K. Schwaber and J. Sutherland (2017) "Scrum Guide," vol. 19, no. 6, p. 504.

[7] R. Madanayake, G. K. A. Dias, and N. D. Kodikara (2016)"Use Stories vs UML Use Cases in Modular Transformation," *Int. J. Sci. Eng. Appl. Sci.*, no. 3.

[8] R. S. Madanayake, G. K. A. Dias, and N. D. Kodikara (2017) "Transforming Simplified Requirement in to a UML Use Case Diagram Using an Open Source Tool," *Int. J. Comput. Sci. Softw. Eng.*, vol. 6, no. 3, pp. 2409–4285.

[9] N. Ferreira, N. Santos, R. J. MacHado, and D. Gašević (2012) "Derivation of process-oriented logical architectures: An elicitation approach for cloud design," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7343 LNCS, Springer-Verlag, 2012, pp. 44–58.

[10] N. Santos *et al.* (2018) "Decomposing monolithic to microservices architectures: a modeling approach," no. August.

[11] Y. Wautelet, S. Heng, M. Kolp, and I. Mirbel (2014) “Unifying and extending user story models,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8484 LNCS, pp. 211–225.

[12] K. Schwaber and J. Sutherland (2011) “The Scrum guide,” vol. 2, no. July, p. 17.

[13] Plant UML a free UML Diagramming tool.(2019) “Commons:Wiki Loves Love - Wikimedia Commons.” [Online]. Available: https://commons.wikimedia.org/wiki/Commons:Wiki_Loves_Love_2019. [Accessed: 07-Feb-2019].

[14] OMG, "OMG Unified Modeling Language TM (OMG UML)", March 2015.