Abstract. Discussions about compatibility between agile methods and software quality models have lately gained significant attention. This is especially true for the requirements engineering area, where agile methods neglect some of the documentation and control procedures traditionally used in its process. This work proposes an extension to the agile method Scrum as an attempt to provide compatibility between Scrum and CMMI in Requirements Management and Development process areas. To achieve that, Scrum is analyzed and some guidelines are defined in order to fulfill the requisites of both process areas that are currently not addressed in Scrum.

1. Introduction

Agile methods have recently required the software engineering area to think over some of its development practices. This is especially true for the requirements engineering area, where agile methods neglect some of the documentation and control procedures traditionally used in its process. Such methods do not mention how requirements should be documented or how to keep requirements traceability. The agile principles presented in the agile manifest show that some of the practices related to requirements engineering are important, such as the understanding of requirements. However, they are against the idea of generating too much documentation as they claim some documentation will never be read.

Discussions about compatibility between agile methods and software quality models, mostly in requirements engineering area, have become more common. [Boehm 2002] emphasizes the difference in the meaning of quality work between agile methods and quality models. In quality models, quality is mostly defined by the conformance to processes and specifications, while in agile methods, quality is determined by customer satisfaction. Furthermore, software quality models, as SW-CMM (Capability Maturity Model for software) and CMMI (Capability Maturity Model Integration), were created for large organizations that usually apply rigid quality standards.

[Turner 2002] comments that, despite the existence of distinct characteristics between agile methods and CMMI model, both have specific plans to try to produce software with quality in the organization. [Paulk 2001], [Boehm 2002] and [Paetsch 2003] also point out to a recent discussion: Is Agile Software Development compatible with the CMMI model? They present differences and similarities in both approaches and consider that software engineering is going through a new phase called Traditional Software Development versus Agile Software Development. In fact, this is currently a polemic discussion and there has been no agreement whether agile methods are compatible to software quality models such as CMMI or not.
This work proposes an extension to the agile method Scrum, called xScrum, as an attempt to provide compatibility between Scrum and CMMI in Requirements Management and Development process areas. To achieve that, Scrum is analyzed and some guidelines are defined in order to fulfill the requisites of both process areas that are currently not addressed in Scrum. Scrum was chosen because it is the agile method with the better capacity to be adapted to larger projects [Boehm 2002] and because it is deficient in mechanisms to control the requirements process.

According to [Chrissis 2003], the purpose of CMMI is to establish a guide for an organization to improve its process and its capability for managing, acquiring and maintaining products and services. The goal of the agile method Scrum is to define a process for object-oriented software development focused on people and it is indicated for environments where requirements rapidly change. Scrum is considered a specific method for managing the software development process and it is based in principles such as small teams with at most seven people, requirements that are unstable or unknown and short iterations.

This paper is organized as follows. Section 2 analyzes the conformance of Scrum to Requirements Management and Development process areas of CMMI. Section 3 presents guidelines that extend Scrum in order to fulfill the practices of those CMMI process areas. Section 4 some conclusions are presented.

2. Scrum Conformance to CMMI

The agile method Scrum was evaluated according to the process areas REQM and RD of the Engineering category of CMMI. Each typical work product resulting from specific practices of this process areas was analyzed to determine if Scrum can obtain that given typical work product in conformance to CMMI. In order to do that, an ordered scale that rates how Scrum addresses such specific practices was defined. The rating scale contains three categories:

- NA: Not Addressed – there is very little evidence that the specific practice is present in Scrum;
- PA: Partially Addressed – there are evidences that the specific practice is present in Scrum;
- A: Mostly Addressed – there are strong evidences that the specific practice is present in Scrum.

For each specific practice in question, Scrum was rated according to the three categories above, as a result of an individual analysis of typical work products in each practice. The results and interpretation of this analysis are presented next for both process areas REQM and RD.

2.1. Requirements Management Process Area

In the following, Scrum is rated according to specific practices of the process area REQM in the Engineering category of the CMMI model.

SP 1.1-1 Obtain an Understanding of Requirements. The Product Backlog addresses this specific practice, therefore, the specific practice REQM SP1.1-1 of the CMMI model is considered Addressed in Scrum.

SP 1.2-2 Obtain Commitment to Requirements. An assessment of the impacts requirements will have on a project is performed by the Product Owner, and the documentation about requirements is registered within accomplished activities during the definition of the Product Backlog and the Sprint Backlog. That way, both work products in practice REQM SP1.2-2 of the CMMI model are addressed by Scrum. Therefore, this practice is considered Addressed in Scrum.
SP 1.3-1 Manage Requirements Changes. During Sprint, the Daily Scrum is used to keep a status of which requirement is being developed, thus addressing item 1 of this specific practice. Scrum does not explicitly state that a requirements database or some other alternative means for storing requirements should be utilized. It also does not mention that there should be a database to support decision making in regard to requirements. Therefore, both items 2 and 3 of the specific practice REQM SP 1.3-1 of the CMMI model are not addressed by Scrum. This practice is then considered Partially Addressed by Scrum.

SP 1.4-2 Maintain Bidirectional Traceability of Requirements. Scrum does not provide a traceability matrix and it also does not implement a requirement tracking system. Thus, the specific practice REQM SP 1.4-2 of the CMMI model is considered Not Addressed by Scrum.

SP 1.5-1 Identify Inconsistencies between Project Work and Requirements. Scrum does not explicitly mention documenting project inconsistencies. On the other hand, it includes corrective actions that must be taken after the deliver of the Product Increment, which is originated from the final Sprint results. Therefore, the specific practice REQM SP 1.5-1 of the CMMI model is considered Partially Addressed by Scrum.

2.2. Requirements Development Process Area

Next Scrum is rated according to specific practices of the process area RD in the Engineering category of the CMMI model.

SP 1.1-1 Collect Stakeholder Needs. Stakeholder needs are all listed in the Product Backlog of Scrum. The Product Owner is responsible for creating and including the list of Stakeholder needs into the Product Backlog. Therefore, as stakeholder needs are identified and collected, the specific practice RD SP 1.1-1 is considered Addressed by Scrum. This practice does not define work products and is applied only in the Continuous representation of the CMMI model.

SP 1.1-2 Elicit Needs. Scrum does not propose the use of any techniques to elicit needs, such as use cases, prototypes, JAD (Joint Application Development), etc. Therefore, the specific practice RD SP 1.1-2 is considered Not Addressed by Scrum.

SP 1.2-1 Develop the Customer Requirements. In Scrum, the development of customer requirements is monitored by the Product Owner, whose assignment is to determine whether requirements are being developed according to what was requested by the customer. Customer constraints on the conduct of verification and validation, are conducted by both Product Owner and Scrum Master, respectively. That way, as all work products are addressed, the specific practice RD SP 1.2-1 is considered Addressed by Scrum.

SP 2.1-1 Establish Product and Product-Component Requirements. In Scrum product and component requirements are detailed in the Product Backlog, along with all functionality, features, infrastructure, architecture and technology the product must offer. The definition of costs and budgets for the project being developed in Scrum is pointed out by [Schwaber 2002], thus indicating that the work product of item 1 is also addressed by Scrum. Therefore, the specific practice RD SP 2.1-1 is considered Addressed by Scrum.

SP 2.2-1 Allocate Product-Component Requirements. The allocation of requirements to product components is not a detailed process in Sprint as Scrum does not define techniques for this phase. Even though the work products of this specific practice may be present during development, these are not specified by Scrum. The specific practice RD SP 2.2-1 is then considered Not Addressed in Scrum.
SP 2.3-1 Identify Interface Requirements. Requirements for interfacing with other systems are not detailed in Scrum. Therefore, the specific practice RD SP 2.3-1 is considered Not Addressed.

SP 3.1-1 Establish Operational Concepts and Scenarios. With the Product Backlog, work products 1 and 2 of this specific practice are addressed by Scrum. Also, since the Scrum Master makes available to stakeholders all concepts utilized during the project, work product 3 is considered partially addressed by Scrum. Although the elaboration of scenarios is not detailed by Scrum, the sequence of events handled by the system is described, thus partially addressing work product 4. Given that Scrum does not define techniques to elicit requirements, such as use cases, work product 5 is not addressed. On the other hand, new requirements can be added after the Sprint Review Meeting, when the Scrum Master reports work results to other participants. At that point, in case a new requirement is identified, it is added to the project, thus addressing work product 6. It is not very clear which rate must be assigned to Scrum for this specific practice. To make that easier, the rating assigned to work product 1, which is the closest to the goal of this specific practice, was applied to the practice as whole. Therefore, the specific practice RD SP3.1-1 is considered Addressed by Scrum.

SP 3.2-1 Establish a Definition of Required Functionality. In Scrum, the functional architecture is indicated when the Product Backlog is created. However, it is up to the developer to apply object oriented analysis techniques such as use cases and activity diagrams, as well as the identification of services. Since work products 2 and 3 are not addressed, this specific practice RD SP3.2-1 is considered Not Addressed by Scrum.

SP 3.3-1 Analyze Requirements. Requirement analysis is present in Scrum. However, it does not produce a defect report as suggested by work product 1 of this specific practice. Work product 2, on the other hand, is one of the most important recommendations in Scrum, as well as in other agile methods. Also, work product 3 is addressed in Scrum because requirements are developed according to stakeholder needs so that key or fundamental requirements are then elicited. With regard to work product 4, requirements analyzed during the elaboration of functionalities do not refer to technical performance measures to be followed during project developments. Since it is not clear this practice is addressed in Scrum, the rating for the work product closest to its goal was taken into account, which in this case is work product 2. Therefore, the specific practice RD SP3.1-1 is considered Addressed by Scrum.

SP 3.4-3 Analyze Requirements to Achieve Balance. Scrum does not define models, simulations or prototypes to analyze risks in requirements. Therefore, the specific practice RD SP 3.4-3 is considered Not Addressed in Scrum.

SP 3.5-1 Validate Requirements. The results of requirements validation, the only work product of this practice, are presented to the Scrum team before they start the development process. Therefore, the specific practice RD SP 3.5-1 is considered Addressed in Scrum.

SP 3.5-2 Validate Requirements with Comprehensive Methods. This specific practice differs from the previous only because it requires a method for requirements validation. The meeting known as Post Sprint Demonstration and Meeting is the method utilized in Scrum to validate requirements. Therefore, the specific practice RD SP 3.5-2 is considered Addressed in Scrum.
3. xScrum: Extending Scrum to Conform with CMMI

This section proposes extensions to the method Scrum so it can address all specific practices in the process areas REQM and RD, thus becoming fully compliant to CMMI in those areas. For each specific practice not addressed or partially addressed by Scrum (according to the ratings presented in the previous section), a guideline is proposed with the necessary additions in order to address that practice.

Guideline 1: Manage Requirements Changes

This guideline aims to address the specific practice REQM SP 1.3-1 of CMMI. In order to manage changes in requirements, this guideline suggests the creation of a document called Requirements Record. This document (i) records new requirements or the changes in existing ones and (ii) keeps a log of the impact caused by each new or changed requirement on others requirements. The Scrum Master is responsible for maintaining this document, which is used during Sprint after the definition of the Product Backlog. The proposed document consists of three sections. The first section (Revision Log) contains the general data about a given requirement: date, author's name and change operation (inclusion, modification or exclusion) and revision number (i.e. version). The second section (Requirements Descriptions) is the most important one. It keeps specific data about each requirement, such as its type, priority, source, version, etc. Finally, the third section (Impact on Requirements) register the impacts of this change on other requirements in the system.

Guideline 2: Identify Inconsistencies between Project Work and Requirements

The purpose of this guideline is to address the specific practice REQM SP 1.5-1 of CMMI. It proposes the use of another document called Component Product Backlog. This document is utilized to identify inconsistencies between existing project work and requirements, and it also helps the detection of deviations, as well as the visualization of inconsistencies and the identification of design elements that are out of the development scope. The proposed document consists of a date, the requirement identification and a brief description of the inconsistency of such requirement in relation to components. The Scrum Master is the responsible for maintaining this document, which is created during the Sprint and validated during the Sprint Review Meeting.

Guideline 3: Maintain a Traceability Matrix

The purpose of this guideline is to address the specific practice REQM SP 1.4-2 of CMMI. It recommends the creation of a traceability matrix which records the relationships among the requirements, between requirements and stakeholders and between requirements and project modules [Sommerville 2003]. The traceability matrix, along with documents proposed in guidelines 1 and 2, acts as a requirement tracking system (which is the other work product of SP 1.4-2). The proposed traceability matrix contains two sections. The upper section (Review Log) contains a date, a project code and a sprint code where requirements will be traced. The lower section (Requirements Traceability) is used to track any inconsistencies ("I") between a given component and requirement, dependencies ("D") between two requirements, or if a component allocates ("A") another component. This matrix serves as a basis to evaluate the impact of changes and to estimate the amount of effort and cost for modifying a project artifact. To avoid maintaining a large traceability matrix, it is possible to define one matrix for each Sprint, referring to requirements and artifacts defined in other Sprints whenever needed. The Scrum Master is responsible for maintaining the traceability matrix, which must be used during the Sprint.

Guideline 4: Allocate Product-Component Requirements

The purpose of this guideline is to address the specific practice RD SP 2.2-1 of CMMI. The idea is to define, for each project work, a document to keep information about product compo-
ments and the requirements allocated to these components. This document also contains information about temporary requirements, derivations of requirements and their relationships. The proposed document is called Requirements Allocation and it consists of two parts. The first one contains general data about the component allocation: date, project code plus component, and a brief description of the component. The second part contains the identification, description, derivations, and relationships of each allocated requirement. The Scrum Master is the responsible for maintaining this document, which is used during the Sprint Review Meeting.

**Guideline 5: Identify Interface Requirements**

The purpose of this guideline is to address the specific practice RD SP 2.3-1 of CMMI. It suggests the use of a document that keeps details about software requirements, including constraints on product features, interfaces with others applications, a description of the domain and additional information about the product matter. This document is called View Documentation and it serves as a sort of contract between users and developers, so it must be written in high level language. The proposed document consists of two sections. The first section (General Data) contains the revision log including its date, author’s name and revision (i.e. version). The second section (View Documentation) contains a date, project code, general description of the product, product view, and name of the person responsible for the project. The Scrum Master and the Client are responsible for maintaining this document, to be used during the elaboration of the Product Backlog.

**Guideline 6: Utilize a Technique to Elicit Requirements**

The purpose of this guideline is to address the specific practice RD SP 1.1-2 of CMMI. It proposes to use of a technique to elicit requirements called JAD (Joint Application Development). This technique was chosen because it is strongly based on meetings like as in Scrum and also because it has been applied in another agile method called ASD (Adaptive Software Development) [Highsmith 2002]. JAD has been successfully used in ASD [Highsmith 2002] and is a group dynamic technique originated at IBM laboratories in the end of the 60's.

**Guideline 7: Analyze Risks**

The purpose of this guideline is to address the specific practice RD SP 3.4-3 of CMMI. Since Scrum does not define any models, simulators or prototypes to analyze the risks of stakeholder needs and their constraints, the idea proposed here is to define a document that describes the risks produced by a requirement in case changes in the functional architecture occur. This document also describes the predicted impact. The proposed document is called Risk Analysis and it consists of three parts. The first part contains general information about the project and requirement in question. The second part keeps information about risk: priority, type, probability and stability. The third part contains an evaluation of risks produced in case a change occurs in the functional architecture plus what should be done. Scrum Master and Client are both responsible for maintaining this document, to be used during the Sprint.

**Guideline 8: Establish a Definition of Required Functionality**

The intent of this guideline is to address the specific practice RD SP 3.2-1 of CMMI. Scrum identifies required functionalities of the system through the Product Backlog. However, it does not mention explicitly that the use case technique should be utilized. Use cases are the main work product in RD SP 3.2-1. Hence, this guideline simply suggests the inclusion of the use case technique in Scrum.

4. Conclusion

The motivation of this work is to be able to determine if agile methods and software quality models can coexist considering they are founded on so distinct principles and ideas. To accom-
plish that, Scrum was analyzed against CMMI and the analysis demonstrated that Scrum does not completely satisfy CMMI practices, in opposite to [Paulk 2001] that related that the Requirements Management process area is addressed by Scrum. An extension to Scrum, called \textit{xScrum}, with a set of guidelines to allow Scrum to be in conformance to REQM and RD process areas of the CMMI model was then proposed. These guidelines are based on the following fundamentals: all practices of REQM e RD process areas must be addressed in Scrum; all solutions must conform to Scrum philosophy; and known software engineering practices, that can help to solve the problems found, must be applied.

In order to validate the extensions proposed two case studies were developed in a software development organization. The first case study was developed for an Internet portal dedicated to provide weather data to farmers and it included the following functionalities: (a) to maintain a database about farmers and producers; (b) to incorporate the module about farmers alimentation; and (c) to develop an authorization module. This case study took approximately 22 days. After its conclusion, some improvements were included in the initial set of extensions. The main goal of the second case study was to develop a system to obtain climatic information from a website containing weather data and insert these into a local database. This case study took 19 days and the improvements from the first case study were applied. In both case studies, the proposed extensions achieved the expected results.

An organization that uses Scrum as its software development process and desires to adapt this method to support the REQM e RD process areas of CMMI can apply the extensions proposed here, which may help such organization to get a quality certification for its development process.

As a continuation to this work, Scrum could also be extended to address remaining process areas of the \textit{Engineering} category: \textit{Technical Solution}, \textit{Product Integration}, \textit{Verification} and \textit{Validation}. It would also be interesting to perform a systematic validation of this method in others software organizations.

References


