Model-based approach for agile requirements engineering using SysML and Papyrus

B Cuesta  
1 Grupo de Investigación en Tecnología y Desarrollo en Ingenierías, Universidad Francisco de Paula Santander, Ocaña, Colombia

E-mail: byroncuesta@ufpso.edu.co

Abstract. This paper presents a model for the application of agile requirements engineering. The literature review made it possible to identify the requirements elicitation process based on standards such as capability maturity model integration, software engineering body of knowledge and how their practices can complement those of agile methodologies, to reduce risk factors and guarantee the success of software projects. This process gives importance to the continuous communication with stakeholders in an iterative process, which is maintained throughout the project and begins with the identification of the problem domain, and then gradually refining characteristics and objectives in user stories and it ends with the development of the product backlog, which will serve as input to the application agile development process. The results propose a model, indicating the different stages for requirements engineering using the agile systems modeling language and the Papyrus tool.

1. Introduction

An effective process during requirements engineering as an initial stage of the software development life cycle, affects the quality of the software to be developed. However, there are theoretical references that indicate the existence of erroneous practices in capturing the requirements that generate little effectiveness, resulting in ambiguous requirements and lack of completeness, mainly due to the lack of effective communication with the interested parties that lead to the need to reformulate them and therefore to delays that affect the project planning in relation to the delivery plan [1].

Modeling is fundamental for the project success; in this way, modeling in an effective and agile way, represents using practices that adopt an agile approach that allow improving modeling activities [2]. For Nicolás and Toval [3], an important element in the software development companies must be aligned with the application of models for their process improvement, so that they adapt to the velocity and constant changes presented by the software industry and technology. According to Lucia and Qusef [4], agile methodologies share values, practices and basic principles that facilitate facing those changes and facilitate the requirements capture and specification.

Perhaps the most important point to understand the software requirements is that a significant proportion of the requirements will change. The change must be managed by ensuring that the proposed changes go through a defined review and approval process and carefully applying the requirements tracking, the impact analysis and the management software configuration. Therefore, the requirements process covers the entire life cycle of software development [5].
2. Background

2.1. Requirements engineering themes
Requirements engineering is considered by experts as the main phase in the software development life cycle [6]. If attention is not paid to what is wanted to be produced, the establishment of requirements may be inadequate, which results in a valuable loss of time and a high repair cost [7, 8]. Understanding the requirement of a real-world problem, constitutes a challenge to attack the problems that arise in the elicitation process, which may be represented in difficulties such as: understanding the problem domain, the disagreement between the stakeholders or the inaccuracies in their formulation, among others [9-13].

The use of agile methodologies presents advantages with respect to the requirements management, due to its practices related to a high communication with the client, which allows to reduce risk factors in the requirements engineering process [14]. Having the client accessible throughout the product development represents the key to the agile approach, since it significantly reduces the possibility that there is no understanding between the parties [15-18].

2.2. Guide to the software engineering body of knowledge (SWEBOK)
The SWEBOK guide is based on standards to establish characteristics of the software requirements. A software requirement exhibits the properties to solve a real-world problem that extends itself as a feature of a set of people, who in some way are connected to the environment in which the software will operate. The requirements can be represented as a functional requirement at the system level or as a non-functional requirement and are known as restrictions or quality requirements [19]. The requirements process covers the entire software life cycle using an iterative nature. Change management, documentation and requirements maintenance are key to the success of the software engineering process.

2.3. CMMI
Capability maturity model integration (CMMI), consists of a series of high-level practices and processes that allow organizations to build a model for the improvement of their processes. CMMI divides software requirements into two process areas: The first is requirements development (RD), which includes elicitation, definition, analysis, specification and validation. The functional architecture, which includes the grouping of the functionality and its relationships, will be obtained from the requirements. The second is requirements management (REQM), which defines the change control and verification, throughout the project life cycle. Part of this management is to document the changes and maintain bidirectional traceability between the requirements and the products that are generated [20].

2.4. Agile requirements engineering
The requirements engineering process using agile methodologies, gives importance to the capture of requirements together with stakeholders, to ensure a correct and traceable process, so that it can be understood by the development team.

The user stories can have different granularity, which represents a particularity wherewith to define how detailed or general is the description of a software functionality. The first level (epics) is associated with the so-called thick granular, which are large user stories that, due to their size, will require several months of development and involve more than one release. They are characterized by the fact that they hardly fit into a single sprint and are suitable to start with, but in the end the team must refine them into stories with a lower level of granularity. The second level is associated with the themes, which arise from decomposing epics that are represented as elements or containers of a set of related user stories. From the topics that are still of thick granularity, a set of user stories called of fine granularity or of adequate size (man-day) are generated, that is, they fit in a sprint and represent the third level of granularity, being essentially a functional refinement [21].

Due to the relationship between the epics and the themes with the stakeholders and their objectives, they should be managed as items together in a context called domain backlog. The user stories obtained
through the refinement of epics and tasks as functional and non-functional requirements, are inserted as items in what is called the product backlog, which allows the development teams to create the sprint planning [21]. Each item represents a user story with sufficient characteristics to satisfy the stakeholders (minimum viable product - MVP) [22,23].

Agile modeling adopts values that allow relying on an effective communication of the development team and the project’s stakeholders. The team must strive to develop the simplest possible solution that meets all needs and get feedback on their efforts continuously and early. In addition, the team must have the courage to stick to its decisions and have the humility to admit that all opinions have value to add to the project's efforts [24].

3. Results

3.1. Proposed model for agile requirements engineering

Agile requirements engineering defines a set of phases that begin with the analysis of stakeholders and their objectives, the definition of the system context, as well as eliciting, documenting, evaluating and managing the requirements in a process marked by continuous iteration in each of the phases (See Figure 1). For the development of each of the phases, elements and practices of CMMI, SWEBOK and agile methodologies were used. It is useful to use models, through system context diagrams, requirement diagrams or other means with UML/SysML.

The systems modeling language (SysML) provides a standard modeling language for requirement diagrams and tables [25,26]. The tool selected for modeling is the Papyrus software, which belongs to the Eclipse foundation [27]. The specification of software requirements within the agile requirements engineering is defined by the following stages:

3.1.1. Analyze stakeholders. An agile project aims to recognize what is most important for the stakeholders and their position in the business. Stakeholders are all people and organizations that can directly or indirectly influence the system requirements [28]. Interested parties do not always consist of future users of the system to be developed, for this reason it is important to investigate users' real needs to identify system requirements oriented with usability and design alternatives.

3.1.2. Define the system context. A system to be developed is connected to its environment through its interactions, in what is called the system environment, which must be known in order to understand and define the system requirements [28]. The next step is to develop a system context diagram using SysML block diagrams, which will allow to demarcate the planned system and its context for a complete understanding of the requirements (see example in Figure 2).

3.1.3. Elicit requirements. Once the system objectives and context are known, general characteristics must be found that allow deriving requirements from the objectives and be associated with epics, to then
decompose into themes, and thus into a refinement process until it reaches the level of user stories, which represent specific requirements that can be implemented by the development team.

Once the epics are evaluated, it is necessary to reduce them by looking for planning units that can be done in one or two weeks (framework of a sprint). The idea is to find more detail to identify sets of user stories that can be grouped into themes and generate value for the stakeholders. In a similar way and understanding that the themes are still thick, they must be broken down into user stories with a range of one to five days per person. Stakeholders must be asked how they imagine the interaction between the user and the system when refining a theme in user stories. Story cards can be used for refinement, placed on a wall that allow structuring a mapping of user stories, where each story allows to externalize the stakeholders' thoughts.

3.1.4. Document requirements. Documentation plays a preponderant role to guarantee requirement traceability and express user stories as planning units for project management. One way to communicate the outcome to stakeholders is through a high-level SysML requirement diagram that illustrates epics, themes and user stories. The Figure 3 shows an example:

![Figure 2. System context diagram elements.](image)

**Figure 2.** System context diagram elements.

![Figure 3. Requirement diagram at the epics level.](image)

**Figure 3.** Requirement diagram at the epics level.
The task refinement allows to obtain independent, value-creating and verifiable units. The description of user stories is governed by a pattern defined as follows: "As <role> I want <functionality> so that <benefit>". Another important element is the one associated with the requirement-verification relationship, called user story acceptance criteria. For this case, the following text template is used: when <action + concept>, It must be fulfilled <specification>. Figure 4 shows the visual possibilities when using SysML requirement diagrams. An acceptance criterion and how it connects with a requirement through a requirement-verification relationship can be seen.

![Figure 4. Requirement diagram at a user story level.](image)

3.1.5. Check requirements. Epics, themes and user stories must be adjusted and verified by the stakeholders in review meetings. The user stories must comply with the INVEST principle, which represents a metaphor as an acronym that serves as a mnemonic rule. The principles are associated with the need for each user story to be independent, negotiable, valuable, estimable, small and testable.

4. Discussion
The requirements express the system purpose and the context in which it will be used. In this way, the success of a project begins with the requirements engineering, being a critical point when the activities are not properly carried out. The CMMI reference model defines practices and activities for the requirement management and development, emphasizing the needs of users and stakeholders.

The model expressed as a result of this document represents a set of stages of agile requirements engineering, which is governed by the elicitation stage and its association to characterize the problem domain by finding epics from identifying objectives and advancing in a refinement process that allows the epics to be broken down into themes and these in turn into user stories in a transition line of user stories from large grain to fine grain, with a level of detail that determines the functionalities that the system must offer. It is important to map user histories to achieve a level of granularity that identifies software features of adequate size for implementation [28].

Modeling allows to communicate and document the system requirements, allowing to know the system behavior before the software development and offers advantages in relation to the quality of the final product (minimize errors) and its ease of maintenance.

Review meetings take time but allow for a better understanding of stakeholders and to learn more about their motivations and interests. As not all stakeholders have the same interests, or have conflicting interests, it is important to identify conflicts between epics and resolve them. This can mean, for example, an agreement, a commitment, a majority decision or a decision at a higher level of hierarchy [28].
5. Conclusions
In the context of requirements engineering, the use of agile world techniques or ceremonies and CMMI and SWEBOK practices, allow to make higher quality software, through mitigating generic problems of the elicitation process, such as poor communication with stakeholders, the vague or open definition of requirements, and not understanding that during the software development life cycle the requirements are always changing. It is important to establish for each requirement a follow-up that allows to verify traceability from its obtaining until its implementation in the software code.

User stories can be described with a different level of abstraction and associated with clearly definable properties of a system that always create value for one or more stakeholders. The user story functionality is expressed in its granularity level.

The use of models allows an effective communication of the problem domain to stakeholders, through the use of requirement diagrams.

References
[1] Lace K and Kirikova M 2018 24th International Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2018) (Utrecht) Vol 2075 (Netherlands: CEUR Workshop Proceedings)
[2] Ambler S 2002 Agile modeling: Effective practices for extreme programming and the unified process (New York: John Wiley & Sons, Inc.)
[3] Nicolás J and Toval A 2009 On the generation of requirements specifications from software engineering models: a systematic literature review Information and Software Technology 51(9) 1291-1307
[4] Lucia AD and Qusef A 2010 Requirements engineering in agile software development Journal of Emerging Technologies in Web Intelligence 2(3) 212-220
[5] Bourque P and Fairley R 2014 Guide to the Software Engineering Body of Knowledge (SWEBOK(R)) (Los Alamitos: IEEE Computer Society)
[6] Sommerville I 2005 Ingeniería de software (Madrid: Pearson Educación)
[7] Westfall L 2011 Las fallas en la ingeniería de requisitos Revista Ingenierías USBMed 2(2) 40-47
[8] Schmid K 2014 6th International Conference, SWQD 2014 (Vienna) (New York: Springer) pp 85-99
[9] Granollers T 2007 MPIu+a. Una metodología que integra la Ingeniería del Software, la Interacción Persona-Ordenador y la Accesibilidad en el contexto de equipos de desarrollo multidisciplinares (Lérida: Universitat de Lleida)
[10] Fernandes J and Machado R 2016 Requirements in engineering projects (New York: Springer)
[11] Katina P, Keating C and Jaradat R 2014 System requirements engineering in complex situations Requirements Engineering 19(1) 45–62
[12] Memon R N, Salim S and Ahmad R 2014 Analysis and classification of problems associated with requirements engineering education: towards an integrated view Arabian Journal for Science and Engineering 39(3) 1923–1935
[13] Ramíez J, Giraldo W and Anaya R 2017 Una propuesta metodológica para mejorar la comunicación en ingeniería de requisitos Revista EIA 13(26) 121-139
[14] Sillitti A and Succi G 2005 Requirements engineering for agile methods Engineering and Managing SoftwareRequirements (New York: Springer) pp 309-326
[15] Paetsch F, Eberlein A and Maurer F 2003 Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (Linz) (Los Alamitos: IEEE Computer Society) pp 308-313
[16] Pichler R 2010 Agile product management with Scrum: Creating products that customers love (Reading: Addison-Wesley)
[17] Amyot D, Anda A, Baslyman M, Lessard L and Bruel J 2016 IEEE 24th International Requirements Engineering Conference (RE) (Beijing) (Los Alamitos: IEEE Computer Society) pp 329-334
[18] Bahill A and Henderson S 2005 Requirements development, verification and validation exhibited in famous failures Systems Engineering 8(1) 1-14
[19] Adams K 2015 Introduction to non-functional requirements. Nonfunctional requirements in systems analysis and design. Topics in safety, risk, reliability and quality (New York: Springer) pp. 45–72
[20] Chrissis M, Konrad M and Shrum S 2003 CMMI guidelines for process integration and product improvement (Boston: Addison-Wesley)
[21] Larman C 2004 Agile and iterative development: A manager’s guide (Boston: Addison-Wesley)
[22] Leffingwell D 2010 Agile software requirements: Lean requirements practices for teams, programs, and the enterprise (Boston: Addison-Wesley)

[23] Patton J and Economy P 2014 User story mapping: Discover the whole story, build the right product (Sebastopol: O’Reilly Media, Inc)

[24] Beck K and Gama E 2000 Extreme programming explained: Embrace change (Boston: Addison-Wesley)

[25] Weikens T 2008 Systems engineering with SysML/UML: Modeling, analysis, design (Massachusetts: Morgan Kaufmann)

[26] Friedenthal S, Moore A and Steiner R 2014 A practical guide to SysML: the systems modeling language (Massachusetts: Morgan Kaufmann)

[27] Arias E, Sufran E, Uranga J and Barrientos C 2017 XXIII Congreso Argentino de Ciencias de la Computación (La plata) (New York: Springer) pp 822-831

[28] Lamsweerde A 2009 Requirements engineering – From system goals to UML models to software specifications (New York: John Wiley & Sons, Inc.)