A Semantic Wiki Approach to Enable Behaviour Driven Requirements Management

Catarina Marques-Lucena*', Carlos Agostinho**,**
João Sarraipa, Ricardo Jardim-Goncalves***

**Departamento de Engenharia Electrotécnica, FCT, Universidade Nova de Lisboa,*
2829-516 Caparica, Portugal (e-mail: c.lucena@campus.fct.unl.pt)
*** Centre of Technology and Systems, CTS, UNINOVA*
2829-516 Caparica, Portugal (e-mail: cml@uninova.pt)

Abstract: Poorly managed requirements are considered as one of the principal causes of projects failure and consequently companies struggle to find an effective solution for requirements elicitation and further management. The adoption of such solution becomes even more difficult when the collaboration between different departments (e.g. marketing and development) is necessary. To address this challenge, the authors propose a methodology for requirements management based on Semantic Wiki and Behaviour Driven Development (BDD). BDD allows developers and end-users to interoperate and encourages seamless collaboration between all project participants. It also certifies that requirements are treated properly by their associated developments through the connection of textual descriptions to functional tests. Semantic wikis can be an added value in requirements management due to their enhanced browser interface and collaborative knowledge sharing capability. They allow stakeholders to participate in requirements management independently of their location. This is of major importance to reduce the problem of lack of inputs from interested parties. Moreover, with semantic wikis adoption, end-users and ontologies can coexist in one system since wiki pages are presented in a human-readable format in parallel to their formal representation in ontologies. This knowledge representation supports companies' decision-making by allowing managers to prioritize implementations, to keep a trace of requirements evolution, and reuse implementations when new requirements enter the system.

Keywords: Requirements Management, Requirements Engineering, Semantic Wiki, Folksonomies, Behavior Driven Development

1. INTRODUCTION

Requirements Engineering concerns the analysis and documentation of requirements (Nuseibeh and Easterbrook, 2000) and its classical approach is composed by 4 processes: 1) elicitation; 2) analysis; 3) specification; and 4) validation (Finkelstein and Emmerich, 2000). Requirements Engineering is needed for planning the development process, assessing the impact of changes and testing the acceptance of outcomes (Hull et al., 2010).

Well formulated requirements lead to quality products and services since they reflect the knowledge and strategies of companies' members (Rogstrand and Kjellberg, 2009). Consequently, techniques that allow all departments (e.g. marketing, developers) to contribute and collaborate in requirements management need to be implemented (Stegaru et al., 2015).

This is even more notorious in the manufacturing sector, where usually companies engage in products development together with their customers and suppliers. Here, requirements elicitation from different teams, with different backgrounds, can result in a failure to capture and translate requirements into meaningful specifications resulting in delayed productions, increased costs and ultimately customers dissatisfaction (Barson et al., 2000; Bozarth and Edwards, 1997; Hausman and Montgomery, 1997). Therefore, a proper approach that allows both business and development requirements to coexist and collaborate is required (Coutinho et al., 2013; Grilo and Jardim-Goncalves, 2013). That is the reason why Behaviour Driven Development approaches should be considered.

In this paper, an assessment related to the benefits of using Behaviour Driven Approaches in requirements managements is conducted. Then, some well-known elicitation techniques and their associated advantages are summarized. It is followed by the state of the art analysis of relevant requirements management tools. In this study the added value of using semantic wikis in requirements management is highlighted.

After this background study, the novel methodology for requirements management is presented in section 5. This methodology supported a framework for requirements management using wiki-based front-end modules (section 6). This semantic enriched framework allows both end-users (industrial partners) and technical teams to collaborate in requirements elicitation and further management. Finally, an application scenario is introduced and some conclusions and future work statements are presented.

2. BEHAVIOUR DRIVEN DEVELOPMENT

Behaviour Driven Development (BDD) is a specification technique that '*automatically certifies that all functional requirements are treated properly by source code, through the connection of their textual description to automated tests'* (Solis and Wang, 2011). Adopting a similar definition, (Tavares et al., 2010) focus on the implication of BDD as a design technique and states that BDD is used to integrate products verification and validation in the design phase using an outside-in style, which implies thinking early on what is the client acceptance criteria before going into design of each part that composes the functionality. Other authors like (Keogh, 2010), argue that BDD is relevant in the whole product life-cycle, especially in the interaction between the business and software development. In addition, is also recognized that BDD permits to deliver value by defining behaviour, and it is focuses on **learning** by **encouraging questions**, **conversations**, **creative explorations and feedback**. In (Lazr et al., 2010) work it is highlighted the value of BDD for business domain and the **interaction of business and developers**, claiming that BDD **allows developers and domain experts to speak the same language** and encourages collaboration between all project participants.

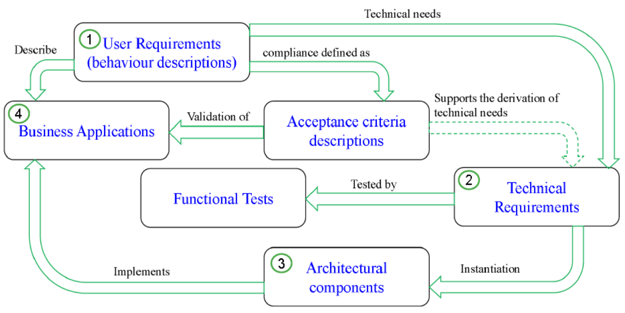


Fig. 1. From User Requirements to Business Applications: A BDD Paradigm.

Fig. 1 illustrates how the communication between the business and development teams can be achieved in products and services development. At first, the *User requirements* are elicited and managed. This is done in such a way that each requirement is written in a form of a **user story**, which structure intends to specify:

- **Who/Which** business or project role is the driver or primary stakeholder of the story (the actor o derive business benefit from the story);
- **What effect** the stakeholder wants the story to have;
- **What business value** the stakeholder will derive from this effect.

To accomplish this, the requirement user story template should take the form:

- **As a** [X]
- **I want** [Y]
- **So that** [Z]

Where Y is some feature, Z is the benefit or value of the feature, and X is the person (or role) who will take benefit of the feature.

The defined *User requirements*, are then used to derive the *Acceptance Criteria* for a specified business application. Since a story's behaviour is simply a business application acceptance criterion, if the system fulfils all the acceptance criteria, it is behaving correctly. Each acceptance criteria should take the form:

- **Given** some initial context
- **When** an event occurs
- **Then** ensure some outcomes

Using the proposed structure, user stories are written using an unambiguous language and the technical team can understand the purpose of each requirement and define some technical requirements from them. Unlike *User Requirements*, that are essentially descriptions of what the system is supposed to do, *Technical Requirements* are directives of how the system should be built, or whose components are demanded. Typically, each user requirement means one desired behaviour and should originate one or more *Technical Requirements*. These are tested by *Functional Tests*, whose should take the same form of the *Acceptance Criteria* (given, when, then).

Each *Technical Requirements* can be used to accomplish several *User Requirements*. Thus, a N:N relationship exists between user requirements and technical requirements. The same happens with the relation between *Technical requirements* and *Architectural Components*. A technical requirement can be linked to one or more architectural components, as an architectural component can be used to implement one or more technical requirements.

2.1. Requirements Engineering in the V-Model

V-Models are used to provide a graphical representation of a system development lifecycle. In general, it provides the activities to be performed (e.g. in a form of requirements and architectural components definition) and the results that have to be produced during the product development (e.g. validation through several tests) (Clark, 2009). V-Models views development in terms of layers, each layer addressing the concerns proper to the corresponding stage of development (Hull et al., 2010).

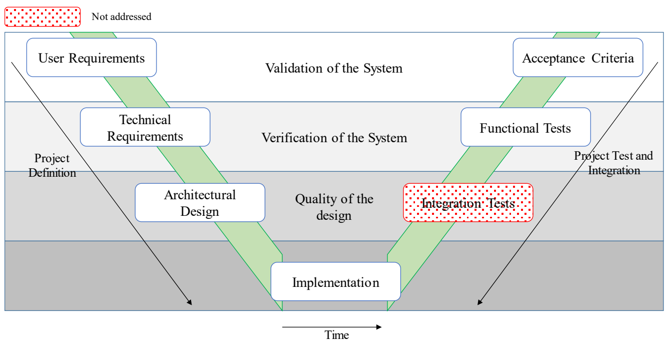


Fig. 2. Requirements Engineering in the V-Model (based on (Hull et al., 2010)).

The usage of Behaviour Driven Development approaches allows an ongoing interaction between Requirements Management and the developments phases of the system engineering. The V-Model of Fig. 2 Is used to summarize the basis relationships established between requirements and testing. Even though they are represented in the figure, the tests regarding the integration between architectural components were not considered in the proposed framework.

3. PREPARATION STAGE FOR MANUFACTURING REQUIREMENTS ELICITATION

Focusing on the manufacturing domain, and based on a previous work proposed by the authors of this article, 5 types of requirements were identified as a must: Business Requirements, Functional Requirements, User-Interface Requirements, Transition Requirements, and Non-Functional Requirements (Marques-Lucena et al., 2015). In that work the authors also concluded that the brainstorming and brainwriting creative technique cover most of the identified requirement types preconditions. However, one inherent characteristic that can be considered as a disadvantage is the low compliance related to the independency of stakeholder's location. This characteristic is the main catalyser for the authors work: to find a solution able to take the best of brainstorming and brainwriting techniques, and that reduces the impact of the stakeholder's location.

Other techniques that show good performance in relation to the presented preconditions are **focus group interviews**, **questionnaires** and **requirements workshops**. Those techniques are currently used to incentive end-users and developers in the requirements elicitation process. Thus, the output of these elicitation techniques should be written and used as input of requirements management tools, starting the brainwriting technique from here.

4. GOING BEYOND TRADITIONAL REQUIREMENTS MANAGEMENT TOOLS

Requirements management entails being able: 1) to relate many different documents to obtain a synoptic view of these document relations; 2) to retrieve information from within those documents; 3) to create special document view; 4) to handle changes made across the set of documents in a consistent manner; and 5) to accommodate diverse documents structuring requirements and document types (Finkelstein & Emmerich, 2000). Meeting those demands, several tools have been developed and released almost every day. Indeed, information about several commercial requirements management tools can be accessed from the INCOSE Survey (INCOSE, 2002). Some of the most well-known commercial tools are DOORS, RDD-100, RequisitePro, and CaliberRM. Beside those professional tools there are others largely used due its simplicity and lake of commercial costs. Examples are wikis, Microsoft Excel and Microsoft Word. Wikis can provide great value as a requirements management tool, including the incentive for stakeholder participation, the support for more consistent documentation through their simple and consistent layout, the improved search and traceability, the support for base lining and versioning requirements, and the support for collaborative requirements review (Beal, 2014).

Simple tools such as Excel and Word can be used to quickly obtain good results in determining requirements. Experience has shown that when formulating requirements, it is important to choose a practical approach that suits the company's organization and available resources. Good results can then be obtained with surprisingly simple tools. Furthermore, these tools are easy to use and do not generate any additional expenses. However, Excel and Word have its limits. For example, they require other tools for collaborative management, for instance Google Docs.

Table 1. Requirement Management tools comparative study.

● : high compliance
⦿ : partly compliant
○ : low compliance

	DOORS	RDD-100	RequisitePro	Caliber-RM	Wikis	Word and Excel
Preconditions						
Input documents enrichment/analysis	●	●	●	●	●	●
Input documents change/comparsion	●	●	●	●	●	○
Requirements Classification	●	●	●	●	⦿	⦿
Requirements Derivation (req. to req., req. to analysis)	●	●	●	●	⦿	○
History of requirements changes, who, what, when, where, why, how	●	●	●	●	●	○
Visibility into existing links from source to implementation - i.e. Follow the links	●	●	●	●	⦿	○
Access control	●	●	⦿	●	●	○
Web browser interface?	○	○	●	●	●	○
Single user/ multiple user concurrent users	●	●	●	●	●	○
Commercial?	●	●	●	●	○	○

The summary of Table 2 was built based on the previous tools' descriptions and on the survey made by (INCOSE, 2002). It represents the compliance of well-known requirements management tools, represented on the top, in relation to the characteristics that define them as a good tool.

It is possible to conclude that commercial tools have a high compliance in relation to the most of the characteristics considered. However, some companies may not be willing or even capable to afford them. In that case, solutions like wikis can be a better choice. Like represented in the table, they show similar results in comparison to commercial tools. Even though, they are still not fully compliant to some characteristics like: Requirements Classification; Requirements Derivation; and Visibility into existing links from source to implementation. Thus, the work here presented is focused on the adaptation of wiki-based modules so they can meet these demands and compete with commercial tools in requirements management.

4.1. Semantic Wikis for Requirements Management Support

Several requirements management works supported by wiki modules can be found in the literature. (Decker et al., 2007) promoted the collaboration between teams through the implementation of templates for communication establishment. However, these templates can be easily edited by stakeholders breaking the replication of requirements structure. This can be a rollback in what concerns requirements classification. The WikiWinWin approach creates a sequence of steps and instructions to guide stakeholders on requirements management work (Yang et al., 2008). During each step, the system displays one or more tools with which the team can generate, organize, and evaluate concepts and information. An identified problem is related to the need of a '*shaper'* role which function is to integrate, distil, organize & rewrite contributions of others. Hence the proper classification of requirements needs to be always supported a human entity. This could be avoided if the information was already inserted properly in the wiki interface. This is one of the challenges that the authors addressed. They also supported the derivation of requirements and their characteristics using proper templates for collaborative tagging. Thus, links can be generated between the several entities involved in requirements management (e.g. user requirements, authors, acceptance criteria). That functionality improved both requirements derivation and links visualization between requirements and implementations.

5. REQUIREMENTS MANAGEMENT METHODOLOGY

The proposed methodology for requirements management starts with a **Preparation** step, where the selection of the more appropriated elicitation methods is a key to reduce the lack of inputs from stakeholders and developers. Several elicitation methods can be selected and used together to achieve better results. In this work, the authors considered brainwriting as a must since it covers the most of the precondition for good requirements elicitation.

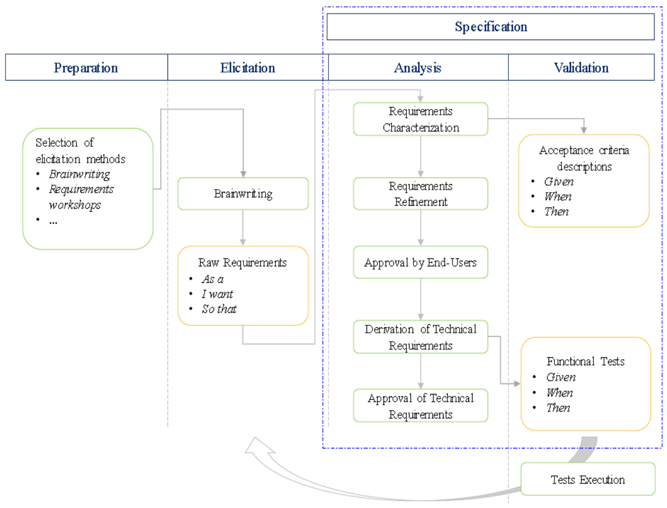


Fig. 3. Requirements Management Methodology.

After this preparation step, the requirements management methodology represented in Fig. 3 starts with the **Elicitation** process. In this process, the requirements begin as ideas or concepts. These can be defined by a single individual, but usually are defined from group's interactions. The novelty of the proposed methodology is that the elicited requirements are written as user stories describing in a comprehensible language what is expected from the system.

Analysis concerns reviewing, analysing requirements in detail, and negotiating with stakeholders on which requirements should be considered (Software Engineering, 2010). Thus, mechanisms that allow these interactions should be implemented. These include requirements approval and refinement. The analysis process also encompasses the transition from user to technical requirements and validation criteria definition. It means that, during user requirements analysis, the end-users should be able to provide the acceptance criteria for their requirements. Thus, a requirement is fulfilled when it behaves accordingly with its acceptance criteria; and the technical team should also be able to define the functional tests for the derived technical requirements.

Requirements **Specification** describes the phase, where the requirements are brought into a suitable and unambiguous form. The idea of this phase is to make requirements readable and understandable by anyone that was not involved in the elicitation and/or analysis process. The Requirements Specification phase documents the agreed requirements at a certain level (Software Engineering, 2010). This is a core process in the proposed methodology. The authors want to come up with an approach that makes sure that, since the moment requirements and validation definitions are elicited and analysed, these are already described in an unambiguous way. This can be done, as an example, using strict forms that only allow requirements to be written in a certain way. Finally, requirements **Validation** is done by checking the compliance of user and technical requirements with their defined acceptance criteria and functional tests correspondingly.

6. BDD BASED REQUIREMENTS MANAGEMENT FRAMEWORK

The proposed BDD based requirements management framework was implemented adopting the work presented by (Marques-Lucena et al., 2015). In that work, the authors argued that the more communication, involvement, and interaction of people, more is the chance for organizations to expose tacit knowledge residing in individuals' heads. To meet this concern, they proposed to use wiki modules for domain experts to expose their knowledge.

The novel proposed framework adapted the previous solution to keep-up with requirements management main steps: Elicitation; Analysis; Specification; and Validation (see Fig. 3). Similarly, to the work that grounded the proposed framework, it relies on the collaborative aspects of Semantic wikis to allow collaborative contributions and further feedback from interested parties, which might not have the technical skill for complex solutions. Furthermore, due to wikis browser interface, this works allowed stakeholders to collaboratively participate in requirements management independently of their physical location.

In the presented work, the classical wiki template was edited to incorporate as many templates as the entities that the system intends to represent (user and technical requirements, authors, etc.). This is the major contribution in terms of wiki front-ends adaptation for requirements management purpose. The mentioned templates were implemented accordingly with the BDD specifications of section 2. The usage of this templates allowed the replicability of requirements structure improving their classification and further analysis.

Fig. 4 depicts the proposed framework for requirements management which is composed by four modules: 1) wiki front-end; 2) Synchronization module; 3) Ontology; and 4) Reasoning and Decision-Making Module. These are used to implement the sub processes of Fig. 3 methodology, namely: Elicitation, Analysis, Specification, and Validation.

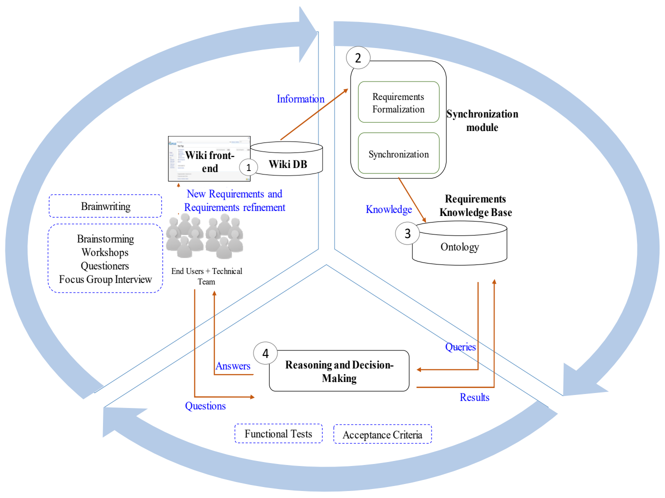


Fig. 4. Requirements Management Framework.

6.1. Requirements Elicitation and Analysis

The requirements management process is established as a cyclic process, since requirements are constantly refined while domain participants increase their contextual knowledge about the system. Consequently, requirements become even more concrete, detailed and complex during a product or service development. Thus, important characteristics that a requirements management tool should address are documents enrichment, modifiability, changes tracking and comparison. Accordingly, with the study provided in section 4, the **wiki-front end** is compliant with all these characteristics and consequently were selected to be used in requirements elicitation.

6.2. Requirements Specification

Wikis are known by being collaboratively edited by domain experts based on the knowledge consulted. However, the content of wiki-based front-ends is characterized by being human readable only. This means that its content is not formalized to facilitate computerized use (e.g. reasoning). To overcome this issue, a **synchronization module**, composed by two sub-modules is used:

– **Requirements Formalization** – Implemented following methodology of Fig. 5. By using it, wikis interface can be adapted to support the structured insertion of requirements from stakeholders despite knowing nothing about its syntax.
– **Synchronization** – This module uses Wiki Data Base to detect any changes that have occurred in the front-end since it was last run and then updates the Requirements Knowledge Base accordingly.

Finally, under the Requirements Specification process, the **Requirements Knowledge Base** is used to represent the knowledge about requirements and related elements (e.g. acceptance criteria, functional tests, authors).

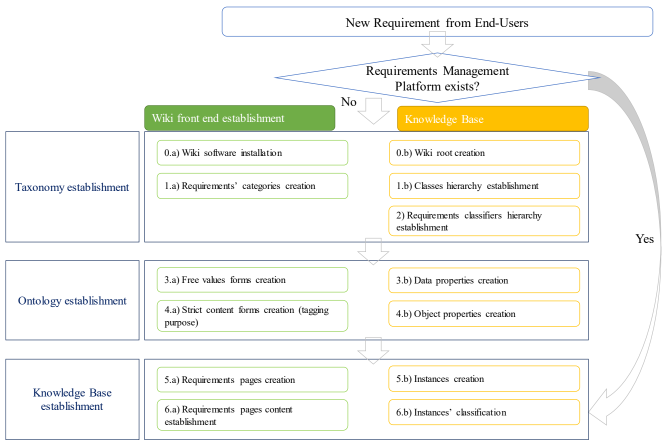

New Requirement from End-Users
Requirements Management Platform exists?
No
Yes
Wiki front end establishment
Knowledge Base
Taxonomy establishment
0.a) Wiki software installation
1.a) Requirements' categories creation
0.b) Wiki root creation
1.b) Classes hierarchy establishment
2) Requirements classifiers hierarchy establishment
Ontology establishment
3.a) Free values forms creation
4.a) Strict content forms creation (tagging purpose)
3.b) Data properties creation
4.b) Object properties creation
Knowledge Base establishment
5.a) Requirements pages creation
6.a) Requirements pages content establishment
5.b) Instances creation
6.b) Instances' classification

Fig. 5. Methodology for wiki-based front-end contents formalization.

Requirements Formalization Methodology

This methodology, illustrated in Fig. 5, encompasses the design of an adapted wiki front-end and the ontology which will handle requirements associated knowledge. The first phase of the methodology consists in the knowledge base's classes taxonomy establishment. This phase is composed by the steps 0-2. In step 0 is made the preparation of the platform to handle the domain users' knowledge. In the step 0.a) is made the software installation, and in step 0.b) is created the wiki root class in the ontology to represent the knowledge about the requirements.

The process of assigning categories to other categories, in the proposed methodologies (step 1.b)), is used to build the ontology's instances taxonomy, being the tagging between them handled as the ontological relation `is a'. In the step 1.a) are created the necessary category pages to allow the elicitation and analyses of requirements. The classification of requirements' contents can be facilitated if a classifiers taxonomy of those contents is defined (step 2). This allows to better structure the gathered knowledge and visualize relations between Knowledge Base's instances.

To guarantee the same structure between wiki articles under a category, the front-end was adapted through the development of forms. They encompass the possibility of insert free valued texts like requirements description sentences (step 3.a), and values that need to be inserted in a strict way to allow proper tagging between wiki pages (step 4.a). With that specification of wiki pages (requirements), is possible to proceed with the steps 3.b) and 4.b) of the methodology. In those steps, for each article section is created a data property or object property to represent the elicited knowledge in the ontology. The object properties allow the connection of the classes under the wiki root class and those under the classifiers taxonomy previously defined (step 4.b)). Data properties will represent knowledge that is not under that taxonomy (step 3.b)).

The process of assigning articles to categories, in the proposed methodology (steps 5 and 6) is be used to instantiate the ontology. This is done by creating an instance under the class with the article's category name (step 5). Then, based on HTML analysis of articles' content, the knowledge of its sections can be represented in the data and object properties of the previously created instance (step 6).

6.3. Requirements Validation

The requirements validation is done through the compliance analysis in relation to the **acceptance criteria** and **functional tests** gathered. However, the authors also considered that requirements validation could be complemented by the reasoning allowed from requirements specification. Thus, requirements validation is supported by the **Reasoning and Decision Making Module**. This module provides to the community structured and contextual information about requirements. One example is: *Which are the most relevant architectural components to be added to the platform concerning the requirements priority?* The capability of the ontology to answer these questions allows the coordination of efforts to the consortium, namely: Prioritize the requirements that are indicated as a priority; Prioritize the implementation of architectural components that embrace a larger number of user requirements. The reasoning on the formalized requirements can also be useful after architectural components' implementation. Let's consider that some management activity is need regarding one or more components. Then, the reasoning and decision-making module allows to verify which are the requirements that are compromised and notify the corresponding authors.

7. IMPLEMENTATION OF THE REQUIREMENTS MANAGEMENT FRAMEWORK IN INDUSTRY

The Sensing Liquid Enterprise[1] concept has been introduced by the Future Internet Enterprise Systems (FInES) Research Cluster with the support of the European Commission. The FInES community acknowledge the fact that businesses are facing unprecedented challenges, given the current economic crisis, but also more systemic changes related to the shortness of resources, environmental changes, and ever changing societal needs. Therefore, our enterprises need innovative ideas to adapt to these changes and remain competitive, or sometimes, even simply survive in the digital era. The Sensing Enterprise concept is an attempt to reconcile traditional (non 'pure' Internet) organisations with the tremendous possibilities offered by the cyber worlds (from the clouds to the dust) (FInES Cluster, 2010; Moisescu and Sacala, 2016; Santucci et al., 2012).

The <u>OSMO</u>sis applications for the <u>S</u>ensing <u>E</u>nterprise - OSMOSE[2] project has the main objective of developing a reference architecture, a middleware and some prototypal applications for the Sensing-Liquid Enterprise, by interconnecting Real, Digital, and Virtual Worlds in the same way as a semi-permeable membrane permits the flow of liquid particles through itself (Agostinho et al., 2015). The worlds represent a way of organizing the structure of an entire manufacturing enterprise, and the business applications in three types of data management environments: **Real World** - related to data that comes directly from devices that is handled by physical components; **Digital World** - related to data management available in data and knowledge bases or Internet (big data); and **Virtual World** - related to specific management of data with the support of artificial intelligence related programs for specific simulations.

The approach presented in this paper follows the necessity of implementing a requirements management tool able to handle the sensing liquid enterprise transition with the Requirements Engineering Methodology presented in Fig. 3.

7.1. Wiki Front-End

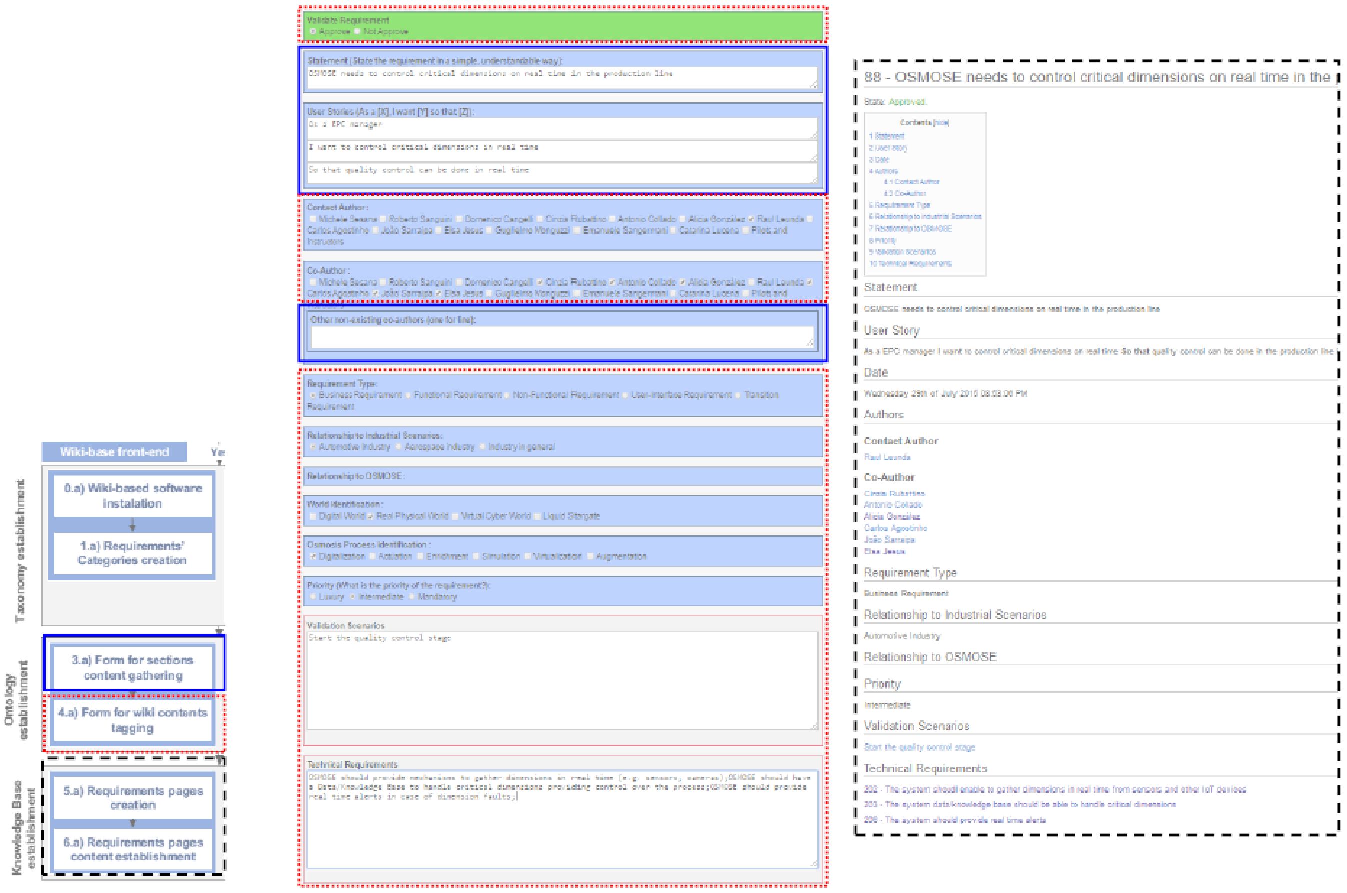
Requirements management tools can be considered as generic. That means they need to be configured to support specific requirements engineering and system development processes. That configuration can be supported by the creation of document templates, schemes of attribute and relation types, and document views. That kind of solution, if applied to wikis, could improve requirements characterization and requirements derivation. Thus, to facilitate the insertion of requirements in the wiki front-end, several extensions were developed following the left part of the methodology of Fig. 5 to originate a form able to facilitate requirements creation and edition (see Fig. 6). The resulting form (requirements article template) is composed by checkboxes and text areas able to suggest the most suitable values to facilitate the tagging between pages and sections (step 4.a)). To handle description sentences, simple text areas are used (step 3.a)).

The representation of the developed form can be observed in the left of Fig. 6, where is possible to verify that the usage of the form resulted in a well-structured article page to represent the requirement, which enables the synchronization with the ontology (steps 5.a) and 6.a)). The adapted wiki front-end, besides the elicitation of requirements through brainwriting, allows its analysis.

First, requirements can be <u>characterized</u> by its: 1) Osmosis world: Real World, Digital World, Virtual World; 2) Type: Business; Functional; Non-Functional; User-Interface; and Transition Requirements; 3) Priority; and 4) Industrial Scenario. The priority is related with the relevance of the requirement in the scope of the project. In the OSMOSE project, two industrial scenarios are considered: aviation and automotive industry.

Since requirements are handled in the wiki-front end, community users can <u>refine</u> the requirements any time, and follow its life cycle while they become even more clarified, focused, consistent, unambiguous and complete. At some point, the requirements are refined enough to be <u>approved</u> by the end-users. The approval of the requirements is made using the checkbox on the top of Fig. 6. Then, the technical team can start the *derivation* of technical requirements. Those are presented in the wiki front-end in a different category

[1] http://finespedia.epu.ntua.gr/Sensing_Enterprise.html
[2] http://cordis.europa.eu/project/rcn/189013_en.html


Wiki-base front-end
Taxonomy establishment
0.a) Wiki-based software installation
1.a) Requirements' Categories creation
Ontology establishment
3.a) Form for sections content gathering
4.a) Form for wiki contents tagging
Knowledge Base establishment
5.a) Requirements pages creation
6.a) Requirements pages content establishment

Validate Requirement
[✓ Approve] Not Approve

Statement (State the requirement in a simple, understandable way):
OSMOSE needs to control critical dimensions on real time in the production line

User Stories (As a [X], I want [Y] so that [Z]):
As a EPC manager
I want to control critical dimensions in real time
So that quality control can be done in real time

Contact Author:
Michele Sesana, Roberto Sanguini, Domenico Cangelli, Circia Rubattino, Antonio Collado, Alicia Gonzalez, [✓ Raul Leunda], Carlos Agostinho, João Sarraipa, Elsa Jesus, Guglielmo Monguzzi, Emanuele Sangermani, Catarina Lucena, Pilots and Instructors

Co-Author:
Michele Sesana, Roberto Sanguini, Domenico Cangelli, [✓ Circia Rubattino], [✓ Antonio Collado], [✓ Alicia Gonzalez], Raul Leunda, [✓ Carlos Agostinho], [✓ João Sarraipa], [✓ Elsa Jesus], Guglielmo Monguzzi, Emanuele Sangermani, Catarina Lucena, Pilots and Instructors

Other non-existing co-authors (one for line):

Requirement Type:
[✓ Business Requirement] Functional Requirement, Non-Functional Requirement, User-Interface Requirement, Transition Requirement

Relationship to Industrial Scenarios:
[✓ Automotive Industry] Aerospace Industry, Industry in general

Relationship to OSMOSE:

World Identification:
Digital World, [✓ Real Physical World], Virtual Cyber World, Liquid Stargate

Osmosis Process Identification:
[✓ Digitalization] Actuation, Enrichment, Simulation, Virtualization, Augmentation

Priority (What is the priority of the requirement?):
Luxury, [✓ Intermediate], Mandatory

Validation Scenarios
Start the quality control stage

Technical Requirements
OSMOSE should provide mechanisms to gather dimensions in real time (e.g. sensors, cameras);OSMOSE should have a Data/Knowledge Base to handle critical dimensions providing control over the process;OSMOSE should provide real time alerts in case of dimension faults;

88 - OSMOSE needs to control critical dimensions on real time in the

State: Approved.

Contents [hide]
1 Statement
2 User Story
3 Date
4 Authors
4.1 Contact Author
4.2 Co-Author
5 Requirement Type
6 Relationship to Industrial Scenarios
7 Relationship to OSMOSE
8 Priority
9 Validation Scenarios
10 Technical Requirements

Statement
OSMOSE needs to control critical dimensions on real time in the production line

User Story
As a EPC manager I want to control critical dimensions on real time So that quality control can be done in the production line

Date
Wednesday 29th of July 2015 08:53:06 PM

Authors

Contact Author
Raul Leunda

Co-Author
Circia Rubattino
Antonio Collado
Alicia Gonzalez
Carlos Agostinho
João Sarraipa
Elsa Jesus

Requirement Type
Business Requirement

Relationship to Industrial Scenarios
Automotive Industry

Relationship to OSMOSE

Priority
Intermediate

Validation Scenarios
Start the quality control stage

Technical Requirements
202 - The system shoud enable to gather dimensions in real time from sensors and other IoT devices
203 - The system data/knowledge base should be able to handle critical dimensions
206 - The system should provide real time alerts

Fig. 6. Creation and edition of User Requirements form.

(with a specific creation and edition template). The tagging between user requirements and technical requirements is supported by an extension able to suggest technical requirements while typing. Finally, the technical requirements can be approved by the technical team in the same way the user requirements were approved by the industrial team. The developed wiki front-end can be consulted through the OSMOSE project page[3] or using the direct link[4].

BDD Implementation in requirements management

BDD features were incorporated in the proposed tool to allow users to insert requirements: 1) user stories; 2) acceptance criteria; 3) and functional tests. in a structured way and following the "rules" described in section 2. As an example, for user stories gathering from industrial partners, User Stories form Fig. 6 was developed. It made sure that all user requirements were inserted using a proper format enabling the technical team to define technical requirements from them.

Requirements Folksonomy

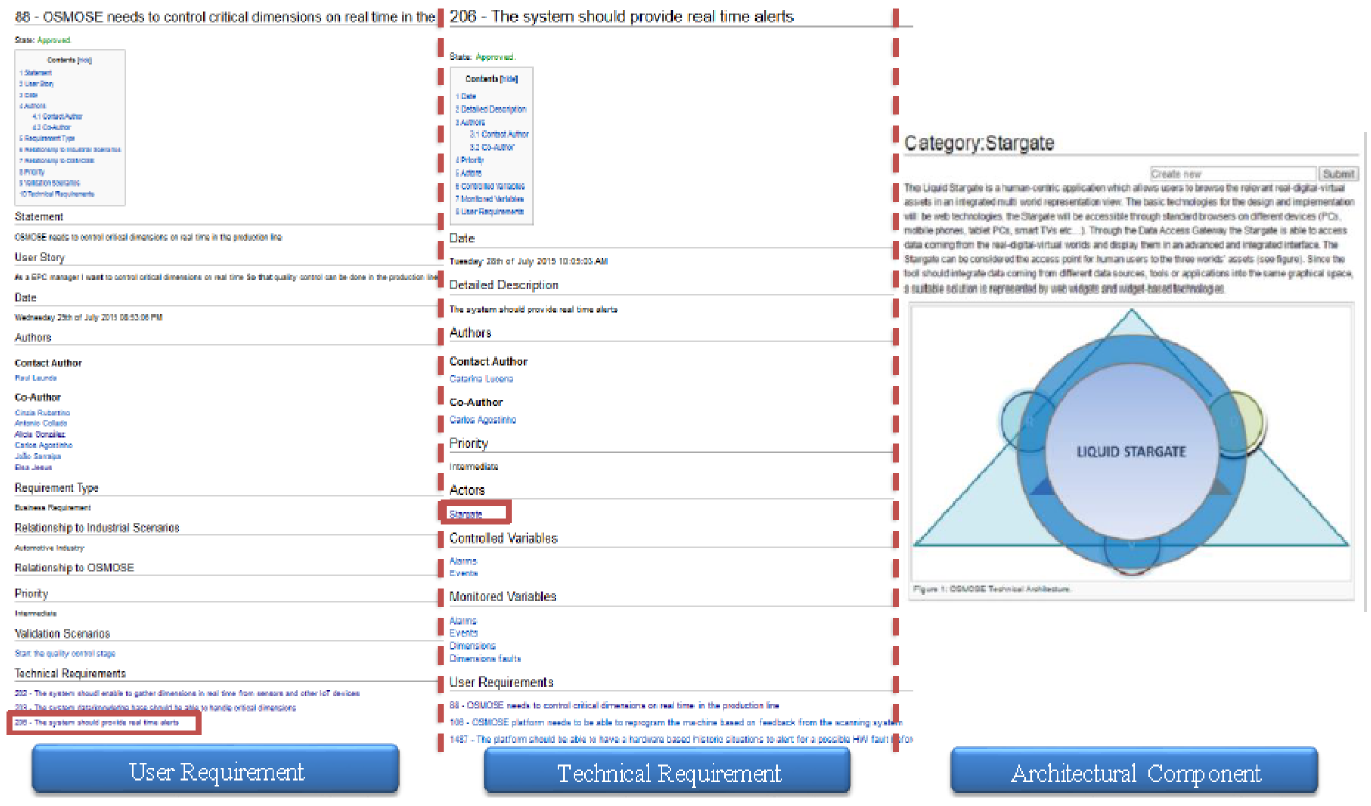
The proposed solution goes towards the folksonomy concept where the OSMOSE community collaboratively contributes for the requirements categorization process. During that process, the developed tool allows the establishment of tags between the wiki contents, namely between requirements and architectural components (developed solutions). The synchronization module is, then, able to translate that tagging into ontological relation allowing further management (e.g. traceability and reasoning).

The wiki main page enables to navigate in the created folksonomy (Fig. 7), starting by consulting the several user requirements. Then, from each user requirement, several links (tags) can be used to access other information like technical requirements and authors.

7.2. Ontology establishment for requirements specification

The OSMOSE requirements knowledge base is a component which purpose is to capture OSMOSE requirements and its relation with other project elements (e.g. authors, architectural components). As explained along the paper, it also serves as a facilitator for requirements management, allowing different views of the information gathered from the wiki. Having this kind of knowledge specified would facilitate the search of specific information.

[3] http://www.osmose-project.eu/
[4] http://gris-dev.uninova.pt/osmose



88 - OSMOSE needs to control critical dimensions on real time in the

State: Approved.

Contents [hide]
1 Statement
2 User Story
3 Date
4 Authors
4.1 Contact Author
4.2 Co-Author
5 Requirement Type
6 Relationship to Industrial Scenarios
7 Relationship to OSMOSE
8 Priority
9 Validation Scenarios
10 Technical Requirements

Statement

OSMOSE needs to control critical dimensions on real time in the production line

User Story

As a EPC manager I want to control critical dimensions on real time So that quality control can be done in the production line

Date

Wednesday 29th of July 2015 08:53:06 PM

Authors

Contact Author

Raul Lucena

Co-Author

Cinzia Rubartino
Antonio Collado
Alicia González
Carlos Agostinho
João Sarraipa
Elsa Jesus

Requirement Type

Business Requirement

Relationship to Industrial Scenarios

Automotive Industry

Relationship to OSMOSE

Priority

Intermediate

Validation Scenarios

Start the quality control stage

Technical Requirements

202 - The system should enable to gather dimensions in real time from sensors and other IoT devices
203 - The system data/modeling base should be able to handle critical dimensions
206 - The system should provide real time alerts

206 - The system should provide real time alerts

State: Approved.

Contents [hide]
1 Date
2 Detailed Description
3 Authors
3.1 Contact Author
3.2 Co-Author
4 Priority
5 Actors
6 Controlled Variables
7 Monitored Variables
8 User Requirements

Date

Tuesday 28th of July 2015 10:05:03 AM

Detailed Description

The system should provide real time alerts

Authors

Contact Author

Catarina Lucena

Co-Author

Carlos Agostinho

Priority

Intermediate

Actors

Stargate

Controlled Variables

Alarms
Events

Monitored Variables

Alarms
Events
Dimensions
Dimensions faults

User Requirements

88 - OSMOSE needs to control critical dimensions on real time in the production line
106 - OSMOSE platform needs to be able to reprogram the machine based on feedback from the scanning system
1487 - The platform should be able to have a hardware based historic situations to alert for a possible HW fault befor

Category:Stargate

Create new | Submit

The Liquid Stargate is a human-centric application which allows users to browse the relevant real-digital-virtual assets in an integrated multi world representation view. The basic technologies for the design and implementation will be web technologies, the Stargate will be accessible through standard browsers on different devices (PCs, mobile phones, tablet PCs, smart TVs etc…). Through the Data Access Gateway the Stargate is able to access data coming from the real-digital-virtual worlds and display them in an advanced and integrated interface. The Stargate can be considered the access point for human users to the three worlds' assets (see figure). Since the tool should integrate data coming from different data sources, tools or applications into the same graphical space, a suitable solution is represented by web widgets and widget-based technologies.

Figure 1: OSMOSE Technical Architecture.

Fig. 7. Requirements pages tagging (folksonomy).

Following the step 0.b) of methodology presented in Fig. 5, the class *Knowledge Representation* was created. Then, the main classes of the taxonomy for requirements representation were identified:

– **Authors** - These are characterized in two groups: end-users and technical team;
– **User Requirements** - Definitions of what the system should do;
– **Technical Requirements** - Characterization of How the system should be implemented;
– **BDD Features** - Composed by acceptance criteria and functional tests.
– **Variables** - That will be controlled or monitored. These are identified in the Technical Requirements;
– **Actors** - Participants of technical requirements (e.g. simulator and simulator programmer);

The OSMOSE knowledge base structure is organized such a way that enables to represent conceptually the requirements classification features and the instances of the Wiki requirements, relating them both while keeping them physically separated. To achieve this, a classifiers taxonomy was build following the step 2 of the methodology for wiki-based front-end contents formalization.

In this step, the necessary information for requirements management is used to create the classifiers taxonomy. Based on it, the three main classes of the classifiers taxonomy are:

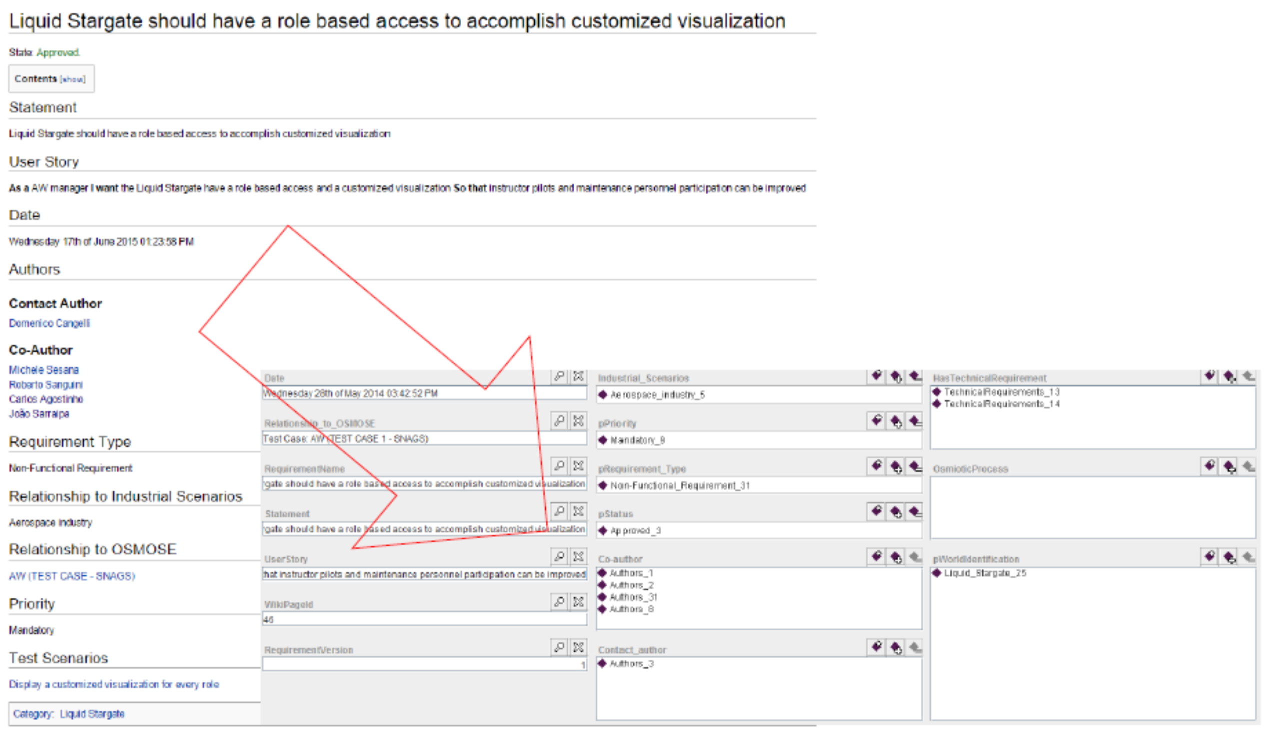
– **OSMOSE Technical Architecture** - enables to define direct links between technical requirements and architectural components;
– **OSMOSE Features** - used to classify requirements using concepts from the OSMOSE project (e.g. Industrial Domain, world; Osmosis Process);
– **Requirements Features** - handles generic requirements features (e.g. priority, Status, Requirement Type).

In the steps 3.b and 4.b of the methodology the data properties to handle free valued properties, and object properties to enable the representation of relations with the classifiers taxonomy are created (left part of Fig. 6). Two of the created data properties that are worth to highlight are the *Wiki Page*, which consists in a unique value that identifies an article in the wiki; and the *Requirement Version* that can range from the version 0 (requirement creation) to the version N (last version of the requirement). Thanks to the adoption of the Wiki page id, a unique value is associated to each requirement. Thus, future developments are clearly indexed to the original requirement contributing to its traceability.

Like is represented in Fig. 8, a direct correspondence between the front-end and the ontology is achieved, and consequently, all the requirements content is successfully migrated.

7.3. Synchronization Module

The synchronization module runs periodically and starts by connecting to the wiki front-end database to verify if any changes occurred since its last run (see Algorithm 1). JDBC (Java Database Connectivity) is used to querying the



Liquid Stargate should have a role based access to accomplish customized visualization

State: Approved

Contents [show]

Statement

Liquid Stargate should have a role based access to accomplish customized visualization

User Story

As a AW manager I want the Liquid Stargate have a role based access and a customized visualization So that instructor pilots and maintenance personnel participation can be improved

Date

Wednesday 17th of June 2015 01:23:58 PM

Authors

Contact Author

Domenico Cangelli

Co-Author

Michele Sesana
Roberto Sanguini
Carlos Agostinho
João Sarraipa

Requirement Type

Non-Functional Requirement

Relationship to Industrial Scenarios

Aerospace industry

Relationship to OSMOSE

AW (TEST CASE - SNAGS)

Priority

Mandatory

Test Scenarios

Display a customized visualization for every role

Category: Liquid Stargate

Date: Wednesday 28th of May 2014 03:42:52 PM

Relationship_to_OSMOSE: Test Case: AW (TEST CASE 1 - SNAGS)

RequirementName: gate should have a role based access to accomplish customized visualization

Statement: gate should have a role based access to accomplish customized visualization

UserStory: hat instructor pilots and maintenance personnel participation can be improved

WikiPageId: 46

RequirementVersion: 1

Industrial_Scenarios: ◆ Aerospace_industry_5

pPriority: ◆ Mandatory_9

pRequirement_Type: ◆ Non-Functional_Requirement_31

pStatus: ◆ Approved_3

Co-author: ◆ Authors_1 ◆ Authors_2 ◆ Authors_31 ◆ Authors_8

Contact_author: ◆ Authors_3

HasTechnicalRequirement: ◆ TechnicalRequirements_13 ◆ TechnicalRequirements_14

OsmoticProcess

pWorldIdentification: ◆ Liquid_Stargate_25

Fig. 8. Direct correspondence between front-end and back-end establishment.

Algorithm 1: Synchronization

```
Establish database connection
Get ‘recentchanges’ → $E_N$
**Foreach** e in $E_N$:
    **If** e(type) in {‘edit’, ‘insert’} **then**:
        Get page current HTML text
        Split HTML code in sections → $S_N$
        **Foreach** s in $S_N$:
            Map s(name) to ontology property
            O(property) = s(value)
        **End**
    **Case** e(type) is ‘edit’ **then**:
        Get e(wikiPage) instance → i
        O(version) = i(version) ++
    **Case** e(type) is ‘insert’ **then**:
        O(version) = i
    Create instance O in ontology
**End**
```

front-end database. By querying the wikimedia table ‘recentchanges’, the authors have access to the set of changed pages, and its type: edition, creation, or removal. If the change is an edition or a creation, through the link to the table text (links to new & old page text) it is possible to have access to the current content of the front-end page. After the collection of the recent changes the HTML of each article or category’s page is processed in order to create/ populate the necessary instances, data properties and object properties in the knowledge base.

7.4. Reasoning and Decision-Making

The formalization of the requirements in a knowledge was developed accordingly with Fig. 4 and supports management features like reasoning over requirements and their historical evolution. These management activities can be sub-categorized, such as:

- Change Management and Traceability;
- Prioritize implementations;
- Common requirements Analysis.

Change Management and Traceability

Requirements Traceability is considered as one of the most important characteristics for requirements management. Some important information can be retrieved from requirements evolution, or in some cases may be necessary to roll back to a past model representation (requirement representation). Since the ontology handles the different versions of a specific requirement taking (directly) into account its properties, it is possible to trace not only between specific dates, but also verify where a specific change occurs (e.g. what where the changes that a requirement suffers to become “Approved”?).

Prioritize Implementations

The defined ontology handles the knowledge about categorized user requirements (e.g. priority and status), their derived technical requirements and consequent architectural components. Therefore, some reasoning to prioritize implementations can be made:

- Which are the approved user requirements that are considered a priority?
    - *Selection of all user requirements whose priority has the value "mandatory" and status has the value "approved"*
- Which are the most relevant architectural components to be added to the platform concerning the user requirement priority?

- *For these requirements follow the path: User Requirements → Technical Requirements → Architectural components.*

– Which are the user requirements that will take advantage of adding one or more architectural components to the architecture?
  - *It is done following the path: Architectural component → Technical Requirements → User Requirements*

The capability of the ontology to answer these questions allows some coordination efforts of the partners, maximizing efficiency.

Common requirements analysis

The major part of the process for common requirements identification is automatic. However, it starts with the manual step of selecting, among the ontological properties available, the characteristics considered as relevant for common requirements election. This process uses the full ramification: user requirement → technical requirements → architectural components. Hence, several properties where elected both from the user and technical requirements:

– **User Requirements**
  - Requirement Type
  - Technical Requirements links
  - OSMOSE World
  - Osmosis Process

– **Technical Requirements**
  - Actors
  - Controlled Variables
  - Monitored Variables
  - Architectural Components links

As part of the manual initial step, it is also defined the minimum shared characteristics (threshold) from which the requirements are considered common or not.

Algorithm 2: Common requirements

```
Define threshold → $Th$
Get set of relevant requirements characteristics → $C_N$
Get all requirements with max version → $R_N$
Foreach r in $R_N$ :
    Get list of relevant properties $OP_N$ defined by
        $OP_N = [p \textbf{ for } p \textbf{ in } r(properties) \textbf{ if } p(name) \textbf{ in } C_N]$
    Foreach p in $OP_N$:
        Foreach re in ( $R_N - r$ ):
            If re(p) == r(p) Then re(score) ++
        End
    End
    If re(score) >= $Th$ Then r(common).add(re)
End
```

At this stage the automatic process can be executed. Therefore, for each requirement, the values of the elected properties are extracted and compared with the values of the full list of available requirements. If a property match, a score is associated to the requirement compared (1 for each characteristic). This is repeated for the set of characteristics selected at beginning, increasing the score each time there is a match. Afterwards, the group of requirements considered common with the one under analysis if the set that share a score above the threshold. The process is repeated for all requirements and all the distinct groups are identified.

Using this algorithm, when a new requirement enters in the OSMOSE requirements management tool, the system can identify the most similar requirements and recommend implementations based on the ones used to develop already existing requirements.

7.5. Execution and Validation of Results

This section reports the compilation and consolidation of the technical requirements validation results. It is done by comparing the actual outcomes of a program execution with its expected behaviour. The complexity of the comparison depends on the complexity of the data to be observed. At the end of the analysis step, a test verdict is assigned. There are three major kinds of test verdicts:

– **Passed** – If the program produces the expected outcome and the purpose of the test case is satisfied;
– **Partially passed** – If the program produces only part of the expected outcome;
– **Failed** – If the program does not produce the expected outcome.

The Technical requirements were implemented using the OSMOSE Technical Architecture and their validation was done through behaviours comparison with the functional tests defined by the technical team. As represented in **Table 2** using a technical requirement associated to the context manager module of the architecture, each technical requirement can yield one or more functional test (validation scenario). Each one of them describe one expected behaviour of the system (technical requirement). A technical requirement is validated in it is compliant with all those behaviours.

Table 2. Technical Requirements Tests and Validation of Results (illustration).

Technical Requirement	**The system should have an events Knowledge Base**
Scenario	***1. Event persistence in the Context Manager***
Acceptance Criteria	**Given** an event happening in a world
	When the event is of relevance for the event history
	Context Manager
Scenario	***2. Event persistence in the Context Manager***
Acceptance Criteria	**Given** relevant events in the past
	When Stargate users open the event history
	Then the Context Manager returns all relevant events

During the technical evaluation 41 requirements have been examined to be included in a prototype: 32% were not addressed since they did not fit the required functionalities while the remaining 68% were implemented and consequently validated. Following the process just explained for technical requirements validation it has been inferred that 96% of the evaluated requirements were passed while 4% (one requirement) only partially passed.

8. CONCLUSIONS

On the proposed Requirements Engineering methodology, behaviour driven features where used to facilitate communication between end-users and technical teams. It was concretized in a wiki based tool for requirements management. Using it, users could collaboratively participate in the requirements management process through an intuitive front-end. Consequently, the lack of domain experts input in requirements elicitation was reduced, which was considered one of the main causes of project failure. The developed tool consists in a wiki which edition section was transformed into a form to facilitate inputs insertion. The developed forms allowed to keep requirements specification, namely their structure and style during its life cycle, addressing the modifiability of requirements.

The proposed BDD based requirements management methodology was successfully implemented in the OSMOSE project when implementing the new sensing liquid enterprise concept. Using the proposed methodology and consequent framework, 92 User Requirements where elicited. Fig. 9. provides a summary of the number of requirements elicited accordingly with their type, OSMOSE World and Osmosis Process. From these requirements 49 where considered as mandatory for the project development.

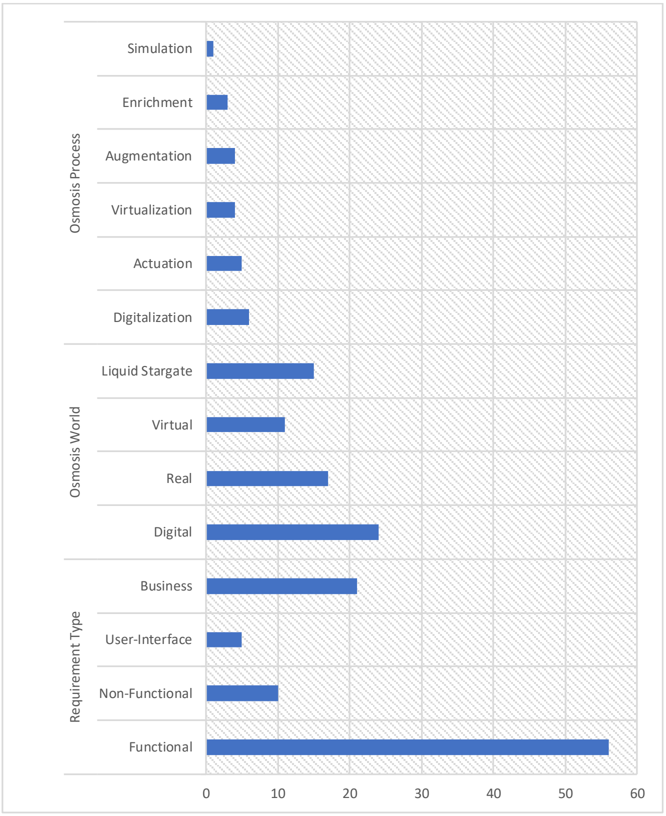


Fig. 9. OSMOSE Outcomes in relation to User Requirements Elicitation.

After User Requirements Elicitation and Refinement, 72 Technical Requirements where derived and associated to OSMOSE Architectural Components (Agostinho et al., 2016). The implemented methodology also supported the project coordination and validation through acceptance criteria and functional tests definition. Section 7.5 describes in detail the coverage of tests execution in relation to the defined requirements.

As future work the authors want to implement the proposed methodology in the cyber physical systems area. A CPS (Cyber Physical System) is a system in which computing, communication and physical processes are so strongly connected that it is not possible to identify whether behavioural attributes are the results of computing, communication, control, physical laws or all of them working together (Dumitrache, 2010, 2011). It means that requirements elicitation of a system like this can be a complex tasks and for sure requires the involvement of several parties with distinct backgrounds (Dumitrache et a., 2013).

Therefore, the authors consider that the proposed approach could be a plus when applied to CPS, since it allows the collaboration of teams with different backgrounds. But to deal with such complex systems, some improvements are required, namely, the capability of handle more formal models when describing both Technical Requirements and Architectural Components.

The authors also want to complement the proposed methodology with links to implementations release. Thus, every time that a piece of code is changed, it should be directly assigned to a new requirements and functional test, allowing companies to keep trace of their implementations.

ACKNOWLEDGEMENTS

This work is funded by FCT – Fundação para a Ciência e a Tecnologia within its Research Unit CTS - Centro de Tecnologia e Sistemas, with reference UID/EEA/00066/2013.

REFERENCES

Agostinho, C., Jesus, E., Sarraipa, J., & Lucena, C. (2016). *Final User Requirement & PoC Specification*.

Agostinho, C., Sesana, M., Jardim-Goncalves, R., & Gusmeroli, S. (2015). Model-Driven Service Engineering Towards the Manufacturing Liquid-Sensing Enterprise. In *Proceedings of the 3rd International Conference on Model-Driven Engineering and Software Development (ModelsWard 2015)*. Angers, France.

Barson, R. J., Foster, G., Struck, T., Ratchev, S., Pawar, K., Weber, F., & Wunram, M. (2000). Inter-and intra-organisational barriers to sharing knowledge in the extended supply-chain. In *Proceedings of the eBusiness and eWork* (pp. 18–20).

Beal, A. (2014). The benefits of using a wiki to manage requirements.

Bozarth, C., & Edwards, S. (1997). The impact of market requirements focus and manufacturing characteristics focus on plant performance. *Journal of Operations Management*, *15*(3), 161–180.

Clark, J. O. (2009). System of Systems Engineering and Family of Systems Engineering from a standards, V-

Model, and Dual-V Model perspective. In *Systems Conference, 2009 3rd Annual IEEE* (pp. 381–387).

Coutinho, C., Cretan, A., & Jardim-Goncalves, R. (2013). Sustainable interoperability on space mission feasibility studies. *Computers in Industry*, *64*(8), 925–937.

Decker, B., Ras, E., Rech, J., Jaubert, P., & Rieth, M. (2007). Wiki-Based Stakeholder Participation in Requirements Engineering. *IEEE Software*, *24*(2), 28–35. http://doi.org/10.1109/MS.2007.60

Dumitrache, I. (2010). The next generation of Cyber-Physical Systems. *Journal of Control Engineering and Applied Informatics*, *12*(2), 3–4.

Dumitrache, I. (2011). Cyber-physical systems-new challenges for science and technology. *Journal of Control Engineering and Applied Informatics*, *13*(3), 3–4.

Dumitrache, I., Caramihai, S. I., & Stanescu, A. (2013). From Mass Production to Intelligent Cyber-Enterprise. In *2013 19th International Conference on Control Systems and Computer Science* (pp. 399–404).

FInES Cluster. (2010). FINES Research Roadmap. *Retrieved, August*, *24*, 2013.

Finkelstein, A., & Emmerich, W. (2000). The future of requirements management tools. *Information Systems in Public Administration and Law*.

Grilo, A., & Jardim-Goncalves, R. (2013). Cloud-Marketplaces: Distributed e-procurement for the {AEC} sector. *Advanced Engineering Informatics*, *27*(2), 160–172.

Hausman, W., & Montgomery, D. (1997). Market Driven Manufacturing. *Journal of Market-Focused Management*, *2*(1), 27–47.

Heslin, P. a. (2009). Better than brainstorming? Potential contextual boundary conditions to brainwriting for idea generation in organizations. *Journal of Occupational and Organizational Psychology*, *82*, 129–145.

Hull, E., Jackson, K., & Dick, J. (2010). *Requirements Engineering* (3rd ed.). New York, NY, USA: Springer-Verlag New York, Inc.

INCOSE. (2002). Tools Survey: Requirements Management (RM) Tools. Retrieved from http://www.incose.org/productspubs/products/setools/tooltax.html

Keogh, E. (2010). BDD: A Lean Toolkit. In *Proceedings of Lean Software & Systems Conference*.

Lazr, I., Motogna, S., & Pírv, B. (2010). Behaviour-Driven Development of Foundational UML Components. *Electron. Notes Theor. Comput. Sci.*, *264*(1), 91–105.

Marques-Lucena, C., Agostinho, C., Marcelino-Jesus, E., Sarraipa, J., & Jardim-Gonçalves, R. (2015). Collaborative Management of Requirements using Semantic Wiki Modules. In *4th International Workshop on Cyber Physical Systems*.

Moisescu, M. A., & Sacala, I. S. (2016). Towards the development of interoperable sensing systems for the future enterprise. *Journal of Intelligent Manufacturing*, *27*(1), 33–54.

Nuseibeh, B., & Easterbrook, S. (2000). Requirements Engineering: A Roadmap. In *ISCE '00 Proceedings of the Conference on the Future of Software Engineering*.

Rogstrand, V., & Kjellberg, T. (2009). The representation of manufacturing requirements in model-driven parts manufacturing. In *International Journal of Computer Integrated Manufacturing* (Vol. 22, pp. 1065–1072).

Santucci, G., Martinez, C., & Vlad, D. (2012). The Sensing Enterprise. In *In FInES Workshop at FIA 2012*.

Software Engineering. (2010). Requirements Engineering = Elicitation + Analysis and Negotiation + Specification or Documentation + Validation ! Retrieved from http://se-thoughtograph.blogspot.pt/2010/01/requirement-engineering-key-aspect.html

Solis, C., & Wang, X. (2011). A study of the characteristics of behaviour driven development. In *2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications* (pp. 383–387).

Stegaru, G., Danila, C., Sacala, I. S., Moisescu, M., & Stanescu, A. M. (2015). E-Services Quality Assessment Framework for Collaborative Networks. *Enterp. Inf. Syst.*, *9*(5-6), 583–606.

Tavares, H. L., Rezende, G. G., dos Santos, V. M., Manhães, R. S., & de Carvalho, R. A. (2010). A tool stack for implementing Behaviour-Driven Development in Python Language. *CoRR*, *abs/1007.1*.

Yang, D., Wu, D., Koolmanojwong, S., Brown, A. W., & Boehm, B. W. (2008). WikiWinWin: A Wiki Based System for Collaborative Requirements Negotiation. In *Hawaii International Conference on System Sciences, Proceedings of the 41st Annual* (p. 24).