Requirements Eng (2017) 22:527–542
DOI 10.1007/s00766-016-0256-4

ORIGINAL ARTICLE

Dealing with software process requirements complexity: an information access proposal based on semantic technologies

Ricardo Eito-Brun[1] · **Antonio Amescua**[2]

Received: 20 August 2015 / Accepted: 24 May 2016 / Published online: 4 June 2016

Abstract Organizations developing software for critical sectors like aerospace, automotive, and medical systems need to apply process requirements coming from different sources: industrial standards, customer-provided requirements, and procedures from internal quality management systems. In these situations, software teams need to deal with complex sets of process requirements that govern different aspects of their work. This paper describes the development of a collaborative, web-based solution to improve access to process requirements. The solution makes use of semantic technologies to handle the context of process requirement. Requirements are contextualized by linking them to activities, tasks, and work products. With this tool, software engineers have a single point of access to all the applicable process requirements, avoiding the risk of missing relevant information.

Keywords Semantic technologies · Software process modeling · Process requirements · Semantic wiki · SPEM

1 Introduction

Software development for critical sectors like aerospace, automotive, and medical systems needs to apply hard guidelines to ensure that products provide the necessary guarantees regarding functionality, safety, and performance. To ensure that final products meet end-users' expectations, industrial consortiums and standardization bodies have developed process requirements standards:

> Process requirements do not directly address the end-item system, but rather how the end-item system will be developed and provided … System or system element implementation process requirements, such as mandating a particular design method, are usually captured in project agreement documentation such as contracts, statements of work (SOW), and quality plans. [35]

These requirements add complexity to software development: Not only do development teams need to master domain knowledge, but they must also know the guidelines in order to conduct different activities and produce expected deliverables.

Complexity increases when process requirements come from different sources: (a) internal procedures from the company's quality management system (QMS), (b) industry-specific standards, and (c) customer-provided requirements. Documents containing process requirements can build a complex set of constraints. Process requirements also indicate when evidence must be collected to support the final qualification and certification of the software product. Some examples from the aerospace industry illustrate this complexity:

- Software developed for European Commission's Galileo satellite constellation must follow requirements defined in different sources: (a) Galileo Software Standard (GSWS) [15], that covers processes from requirements' specification to software acceptance and maintenance; (b) Galileo Management Requirements,

✉ Ricardo Eito-Brun
reito@bib.uc3m.es

Antonio Amescua
amescua@inf.uc3m.es

[1] Universidad Carlos III de Madrid, c/Madrid, 128, Getafe, Madrid, Spain

[2] Department of Computer Engineering, Carlos III University, Madrid, Spain

focused on management and project status reporting [18], (c) Galileo FOC Configuration and data management [19] for configuration and document management activities, (d) Galileo FOC GS Assurance and Safety Requirements [16], for quality assurance and safety-related aspects, (e) Galileo FOC-System Verification Requirements Document [17], governing the planning, execution, reporting, and tracking of verification activities, or (f) Requirements for Delivery of Documents [20].

- Software developed for the European Space Operations Center (ESOC) must also meet requirements defined in several sources: (a) tailoring of ECSS-E-ST-40C for space engineering—software [21], (b) tailoring of ECSS-Q-ST-80C for space product assurance—software product assurance [22], (c) tailoring of ECSS Software Engineering Standards for Ground Segments in ESA. Part A–D(**) (BSSC 2005(1) [14], (d) ESOC Generic Ground Systems: Development Requirements Specification [24], or (e) Software Quality and Coding Rules [23].

This can be considered an information overload problem: In addition to the complexity of the technical and functional requirements, staff needs to be aware of a large set of process requirements scattered throughout different documents. This may be error-prone, and there is a risk of missing information and producing deliverables that does not meet all the required constraints. Inadequate access to information has a negative impact on productivity, and errors can be made because of missing information. The omission of activities, tasks, and controls requested by applicable process requirements can also lead to erroneous estimations of effort and compromise the reliability of implementation.

To deal with this problem, organizations must answer these questions:

(a) How can we ensure that involved people are aware of all the applicable requirements?
(b) How can we ensure that work products respect all the constraints?
(c) Are we sure we are doing the tasks as requested?

This paper describes the implementation of a tool that offers a single point of access to corporate guidelines, industry, and customer process requirements. The tool—built on top of the semantic wiki (SMW) content management tool—uses the software process engineering metamodel (SPEM) language to represent processes, activities, tasks, and work products where requirements are allocated. Besides improving access to process requirements, the proposed solution contributes to organizational maturity, as it helps companies adapt their processes to fulfill the needs of different projects [2]. The conceptual basis of this solution is related to the proactive management of work context and contextual information.

2 Objectives of research

The objective of this research is to build a framework for improving access to software process requirements by using, as inputs, problem statements identified through individual interviews with staff working in a software development company.

The main research hypothesis is that a clear contextualization of the software process requirements within tasks, activities, and deliverables allows staff to gain a better understanding of the software development process they need to follow. The systematic tagging and classification of process requirements will improve engineers' capability of working with complex sets of requirements coming from different sources.

The model, and the tool developed for its validation, has been built on top of the ISO/IEC 29110 international standard for software development. Processes defined in this standard have been used to build an exemplary context where process requirements may be allocated. ISO/IEC 29110 could be replaced by other process models to accommodate the needs of different organizations. The model extends the SPEM metamodel and incorporates additional items to manage process requirements and base practices.

The rest of the paper is structured as follows: In Sect. 3, the research methodology is described; Sect. 4 discusses the role of context in software development activities; Sect. 5 describes the construction of the activity context model and the proposed implementation for a demonstrator; Sect. 6 describes the validation of the proposal; and Sect. 7 summarizes the research conclusions.

3 Research methodology

Research has been conducted using the action research methodology, the purpose of which is to influence or modify some aspects of the object or situation under study. Action research combines observation and action to propose and implement improvements with the commitment of the target organization [62, p. 170]. Research was completed in a six-month period in the context of an aerospace company located in Madrid, Spain, with subsidiaries spread in different European countries and North America.

The research process has been structured in these steps:

(a) First, an analysis of the current situation was done, focused on the complexity of applicable software process requirements in two representative projects. The process requirements' complexity is mainly due to the large number of applicable requirements and guidelines in critical sectors (aerospace, automotive, and medical software) and because they are usually spread throughout different documents. Some metrics have been proposed to measure the complexity of process requirements, like design constraints imposed [68]; metrics defined for measuring process complexity might also be applied, like those described in Kluza et al. [39]. The analysis of process requirements was done by direct observation, interviews, and the analysis of applicable standards.
(b) Second, an extension of the SPEM metamodel was defined to incorporate process requirements and guidelines, their properties, and generate an ontological representation that allows the use of these elements to organize requirements' data.
(c) Third, an implementation was done using a popular content management tool—semantic wiki—with existing, real data. The implementation put software process requirements within the context of the different tasks that staff should execute in a typical project.
(d) Finally, the validation of the approach was conducted by means of interviews and focus groups with staff involved in different projects. During the implementation period, the approach was continuously assessed and refined with feedback provided by the staff involved in the research project.

4 Contextual information for software development activities

One key element in the research methodology was the identification of questions related to the information needs of the target users, through direct observation and interviews. Typical questions included: (a) those related to the traceability between information items to assess the impact of changes, and (b) those related to the quality characteristics and expected content of the project deliverables. From these types of questions, it was possible to characterize the *contextual information* software developers must know to ensure the quality of the process.

4.1 Context in software engineering

The meaning of context in software engineering is usually restricted to the context in which the target system will be used. Software engineering practices recognize the need of understanding and modeling the end-user and system operations' context [9], and different techniques have been proposed in the professional literature to achieve this objective [3, 27]. Besides this restricted meaning of "context," software engineering should consider the context in which managers and engineers complete development and maintenance activities. Context is defined as "the set of all events and information, which can be observed or interpreted during knowledge work, except those events and pieces of information that constitute the change" [49]. A simpler, pragmatic definition adopted in this study refers to "the set of rules and constraints that need to be known to do a task or generate a deliverable with success." This definition is closer to the use of "context" in business process modeling standards like IDEF0, where context diagrams include the processes' activities and the rules and guidelines that govern their execution in the form of controls and mechanisms [12].

The big number of process requirements established by standards defined for critical domains (aerospace, automotive, or biomedical) constitutes an information overload problem [26, 77]. Management of contextual data is one of the main approaches for dealing with this challenge. However, which data sets constitute the context that software engineers need to manage? Interviews conducted with software developers working on two different projects identified three types of "contextual information": (a) technical context, (b) organizational context, and (c) activity context.

Technical context refers to the functional and non-functional requirements that must be implemented in the software product. It also includes data about known issues, their status (requested, accepted, answered, implemented, etc.), and the planned evolution of the product: engineering change requests (ECRs), product roadmap, planned features, future versions, etc. Technical context is usually managed by means of traceability data, configuration, and change management tools supporting impact analysis. Technical context gives answers to questions like:

- I need to update this module: Which regression tests do I need to execute?
- Are there any planned changes in the module I need to update?
- What are the source code files that are going to be affected by these changes in design?
- If this source code file is updated, which validated requirements may be affected?

Organizational context refers to the rules that govern the team organization: decision escalation processes, staff and role responsibilities, and customers' and colleagues' expectations. Organizational context is usually managed

with the help of workflow and notification systems that raise alerts when activities are done and distribute tasks accordingly. Continuous integrative environments play an important role in task automation. Organizational context data give answers to questions like:

- Who should know about the changes I am implementing?
- Has V&V staff been notified of the completion of the task, so they can start their activities?
- Who must review and approve the changes we propose to fix this problem?
- Who must execute the regression approach after the implementation of these changes?

Activity context refers to the list of tasks to do during the project life cycle, their purpose and objective, expected input and outputs, and the rules to follow. Activity context is usually managed by means of descriptions of processes, work packages, activities, tasks, and work products. The contextual data gives answers to questions like:

- What are the tasks the team has to complete?
- What is the sequence to follow?
- What are the inputs needed to do these tasks?
- What are the expected outputs?
- What is the expected content of the outputs?

These questions are mainly related to activities, tasks, or work products. In other words, it is feasible to use the activities, tasks, and work products defined in the reference process model used to put process requirements into context. These requirements act as: (a) "drivers" for the planning and completion of activities and work products, and (b) verification criteria to ensure that tasks and work products are done as expected.

The relationship between the types of contextual data and software requirements can be summarized as follows: (a) technical context is clearly related to product functional and non-functional requirements, (b) organizational context is related to requirements that govern the assignments of responsibilities to roles and workflow rules, (c) activity context depends on process requirements. This document deals with the third type of contextual data, *activity context*. This approach may be traced to previous research: For example, the role given to deliverables in activity context may be related to the *artifact-based requirements engineering* approach described in recent studies [52].

4.2 Activity context

The relationship between activity context and software process models is clear: Process descriptions not only provide companies with the list of processes, activities, tasks, inputs, and outputs, they constitute the basis for project planning and can be used as *templates* that are instantiated when a new project starts.

> Software process models give companies guidelines on how to complete projects using well-defined, proven methods, tasks and activities. […] They include descriptions of activities, their inputs and outputs, the flow of data and control, the techniques, methods, tools and roles. [53]

Process requirements should be incorporated into the process models as constraints. Companies implement different process models, which are usually said to be the sources of their competitive advantage and the reason behind their performance. This traditional view is challenged by the fact that most corporate process models are based on standards like ISO/IEC 12207 or ISO/IEC 29110. Usually, companies make an adaptation of these standards by choosing a subset of their processes, tasks, and activities according to well-known tailoring guidelines. As a result, a high degree of commonality may be observed: Corporate models include processes like requirements elicitation, architectural design, interface definition, coding and unit testing, software integration, and the main differences are not found in the processes' definition, but on their capability (i.e., predictability).

Similarities can also be observed in work products. Industrial standards request similar work products (requirements specifications, design documents, test specifications, verification reports, installation, configuration reports, etc.) and content guidelines. As an example, a comparison of the work products requested by ISO/IEC 29110 with those requested by GSWS shows minor differences: GSWS includes three additional work products (SCAR, SAR, and PSJF)—two of them required for real time and critical software—and separate plans for project, configuration, product assurance, and V&V management (instead of a common plan).

The fact that different companies follow process models with a high degree of similarity led us to consider the feasibility of using a standard process model as the basis for building the activity context. ISO/IEC 29110 "Software engineering—Lifecycle profiles for Very Small Entities (VSEs)" was chosen as a reference. This standard provides organizations with a prescriptive process model adapted to the needs of VSEs (very small entities). The basic profile establishes two processes—project management (PM) and software implementation (SI)—with their activities, tasks, inputs, outputs, and roles. This standard offers SMEs guidelines on how to develop software and a reference to conduct improvement plans. The importance of ISO/IEC 29110 has been widely discussed in the professional literature [43, 44, 56, 60, 61]. The possibility of tailoring the processes defined in this standard is possible. In fact, it is

expected that companies adopting ISO/IEC 29110 will make some kind of adaptation, emphasizing those tasks that add more value to their business and incorporating additional practices, tools, and methods [6]. As an example, the process model can be applied to different life cycles (waterfall, incremental, etc.) or adapted to agile practices; in some cases, companies can provide a more detailed definition for some activities or work products (e.g., adding references to the tools, templates, or techniques used to execute the task).

5 Model construction and implementation

Based on the premises described in the previous section, a model was developed as an extension of the SPEM metamodel for software development processes. The model establishes a general framework that allows the encoding and representation of multiple process descriptions and requirement sets. ISO/IEC 29110 was used as a reference to model an exemplary process.

5.1 SPEM and ISO/IEC 29110 as foundations

SPEM, a MOF-based metamodel and conceptual framework published by the Object Management Group (OMG), establishes the concepts and notations to represent, exchange, publish, and enact processes. Its scope includes "the minimal elements needed to define any process and accommodate a large range of development methods and processes of different styles, cultural backgrounds, levels of formalism, life cycle models, and communities." [57].

SPEM makes a distinction between processes and method content. Method content corresponds to *reusable items* that may be used in the definition of processes: *tasks*, *work products*, *roles,* and *categories*. These items are related: Roles participate in tasks that generate or consume work products; categories are used to classify method content items, etc. Method content can be reused, organized, and sequenced in different ways to support alternative scenarios: waterfall, agile, incremental, etc. To leverage reuse capabilities, method content may be customized without changing the definition of the base, reused item. Method content items are combined using activity diagrams or breakdown structures to form processes or higher-level activities. Two types of processes are distinguished: delivery processes and capability patterns. The first one corresponds to end-to-end process templates, and the second one to sub-processes or process fragments that may be later assembled to build delivery processes. SPEM also introduces the concept of phases, iterations, and milestones.

This standard metamodel offers the elements needed to build an ontology with the different items needed to build the activity context. The decision to use SPEM to represent the software development characteristics is due to the following factors:

(a) SPEM is a recognized standard for software process modeling and representation. Like other process modeling languages, SPEM allows the homogeneous description of process languages using a graphical notation.

(b) There are other software process metamodels that could be used as a reference: Open Process Framework (OPF),[1] Software Engineering Metamodel for Development Methodologies (SEMDM) [36], Microsoft Solution Framework (MSF) [73], or the German standard V-Modell XT [72]. The decision to use SPEM was due to the following factors:

  (a) SPEM is an open standard managed by a recognized standardization body, the Object Management Group (OMG) that is also responsible for the widely adopted Unified Modeling Language (UML).

  (b) SPEM includes a complete graphical notation to represent processes and activities at different levels of detail (from diagrams showing the process overview to specific activity diagrams). Graphical notations used in SPEM correspond to well-known, widely used diagramming guidelines, like UML activity diagrams or process diagrams used in the Rational Unified Process (RUP). This means that software engineers can work with diagramming notations that are familiar.

  (c) Other alternatives not only provide the software process metamodel, but a repository of specific activities and work products. SPEM focuses on the metamodel definition, the rules to represent processes made up of any kind of activities, tasks, and work products, and their graphical representation. This gives the flexibility to represent any arbitrary software development process, e.g., those based on ISO/IEC 29110, RUP, OpenUP, or any other one.

  (d) SPEM is supported by a wider number of tools, including the open source Eclipse Process Framework (EPF) and commercial tools like Sparxsystems Enterprise Architect, Osellus IRIS, or Objecteering.

  (e) SPEM is widely used and accepted by the industry [63] and is highly visible in the professional and academic literature [66].

[1] See http://www.opfro.org/. Last visited 25/03/2016.

The possibility of building ontologies using the SPEM metamodel as an input has already been discussed by Aoussat et al. [4] for software process reuse, Liska and Navrat [47] identified the value of SPEM to improve the weaknesses of software process notations described in the SWBOK, and Gazel et al. [28] took SPEM as an input for building a tool to guide CMMI assessments. In this research, the SPEM model was extended to support the ontological representation of the process requirements (see Fig. 1). Individuals or instances for the ontology were taken from the basic profile of the ISO/IEC 29110 standard, building a process representation with a minimum set of elements that can be later expanded and mapped to corporate process models.

Once the activity context (process, activities, tasks, and work products) is established, companies have at their disposal a framework where process requirements may be allocated to activities, tasks, and work products to provide staff with:

(a) A single point of access to the process requirements scattered across several documents.
(b) A detailed, complete view of the set of constraints to consider when doing a task or generating a work product.

The allocation of process requirements extends the process model (activities, tasks, and work products) with the applicable constraints. For example, the standard may request the regular reporting of the project status to the project's stakeholders; this general task may be affected by particular customer requirements imposing constraints on the frequency of the reports, the layout to use, or the data that need to be provided.

5.2 Specification of the demonstrator

To assess the feasibility of the approach in practical terms, a demonstrator was developed with real data provided by the company. The target solution was expected to achieve these objectives:

- Provide a single point of access to all the applicable process requirements.
- Put requirements into context, by linking them to the tasks and work products on which they have an impact.
- Support the definition of corporate processes and its mapping to existing standards.
- Support collaborative work around requirements: annotations, attach evidences, etc.
- Visual representations of the processes.
- Easy access for all the staff involved in the project, avoiding complex learning curves.
- Possibility of creating "ad hoc" queries.
- Incorporation of additional criteria for cataloging process requirements like criticality, systematic reuse.

It is remarked that the proposed solution does not have the aim of managing traceability between work products (e.g., between requirements and design elements, and requirements and test cases), as this type of traceability is part of the technical context of the solution; the purpose of the solution is to deal with the *activity context*, related on how to build the solution from a process perspective.

Figure 2 shows the main use cases defined for the demonstrator.

The planned functionality required the adoption of a content management tool. An efficient solution for managing access to process requirements must support

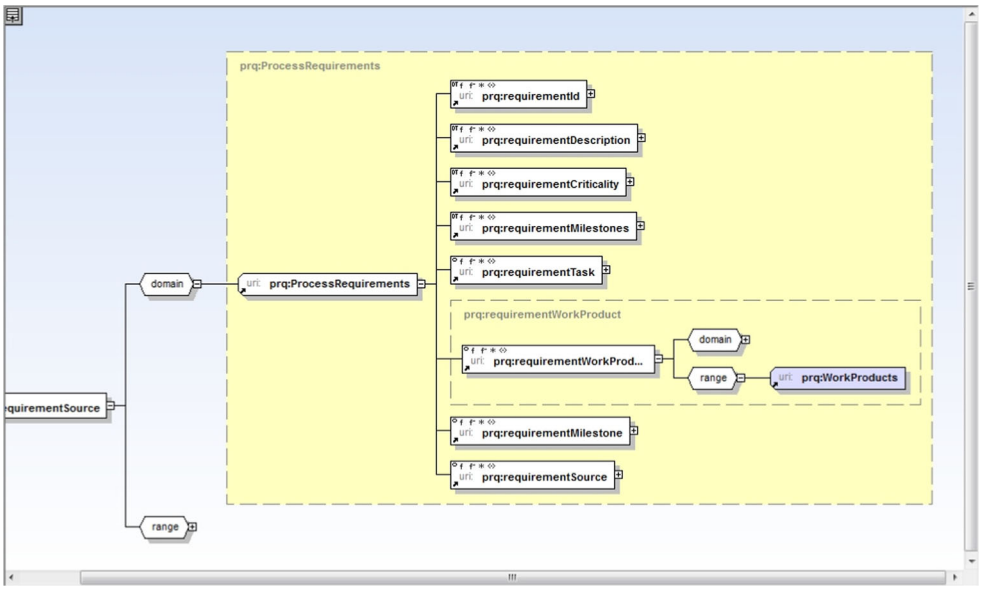


Fig. 1 Fragment of the ontology for encoding requirements data

collaboration and content edition features. Process requirements may be subject to interpretation, and additional data need to be gathered to record how requirements were interpreted in previous projects, or what kind of evidence is needed to demonstrate their achievement. In other words, access to process requirements may be enriched with additional comments, explanations, and sample evidence. To handle this content, it is necessary to set up a collaborative workplace supporting distributed data collection and discussion around requirements.

A trade-off analysis was performed to assess several content management tools (CMS). Among these tools, semantic wiki (SMW) was chosen. SMW is a content management platform that extends the functions of standard wiki sites with the capability of adding properties and metadata to pages and links. These properties may be used to tag content and make its meaning explicit.

The decision to adopt SMW was guided by the tool's capabilities:

- Support to standard ontological knowledge representations with classes, properties, and individuals.
- Ingestion of data encoded in XML, and the capability of exchanging data with process modeling tools.
- Easy incorporation of users' contributions in the form of annotations.
- Wide adoption in the collaboration and knowledge-sharing strategies of many companies, including large-scale collaborative platforms [33].
- Low complexity, adoption, and deployment costs.

SMW features may be summarized as follows:

- *Categories and properties*. Properties work as an extension of MediaWiki Categories to classify and tag data items. Properties define the semantics of the data and the relationships between data items.
  Properties are not predefined, so it is possible to define and use different properties and metadata. There are initiatives to propose a common set of properties for tagging semantic wiki content, like SWiVT; but the capability of defining custom properties and establishing equivalences with properties defined in other schemas offers the possibility of reusing vocabularies and ontologies already defined in the software engineering field [8, 70, 74].
- *Semantic browsing, searching, and data discovery*. Once the wiki content has been tagged with properties, these metadata can be exploited in different ways. For

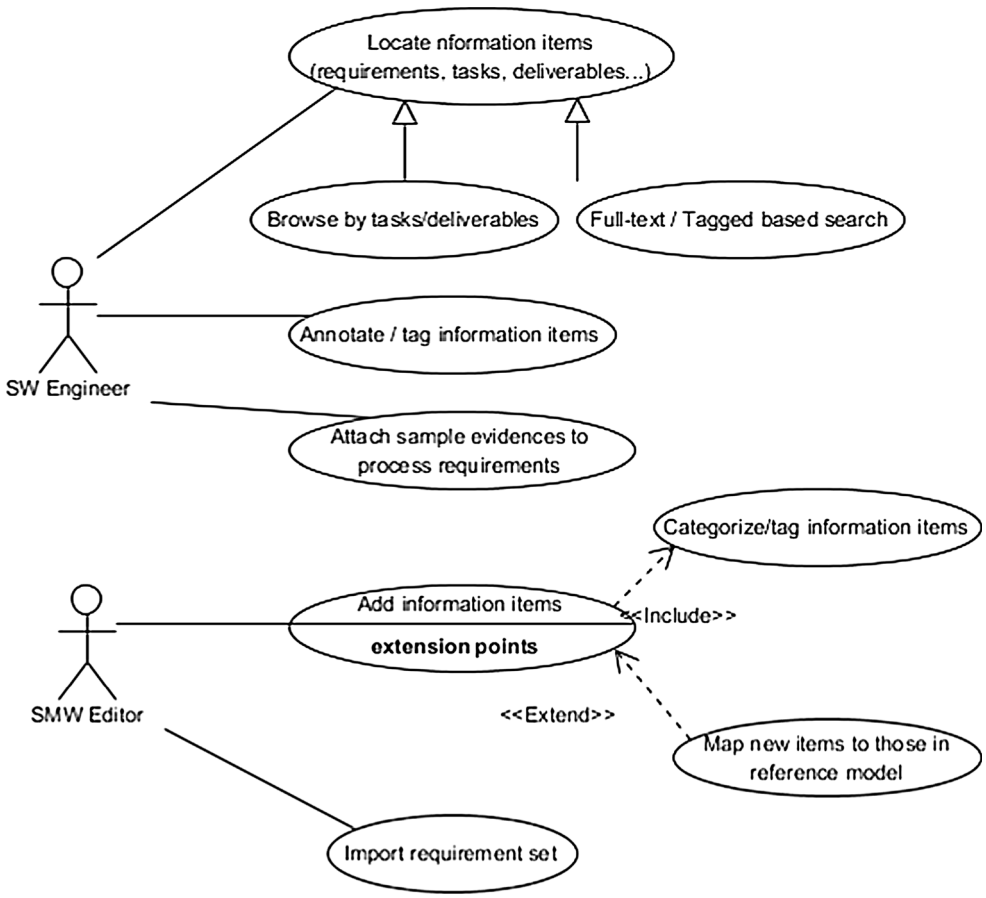


Fig. 2 Use cases supported by the tool

example, "factbox" summarizing property values. By using the properties, users can navigate and locate items matching specific constraints. For example, properties can be used to search: (a) requirements related to specific tasks and work products, (b) work products used as inputs or outputs of tasks, or (c) tasks to be completed by a particular role. A sample search run using properties is shown below:
[[Category:WorkProduct]]
[[usedAsInputFor::TestDesign]]

- *Hierarchical arrangement of classes* (types of items) *and properties*. Using generalization/specialization and "is part of" relationships. Hierarchies are needed to expand and restrict searches.
- *Inline queries and concepts*. These are dynamic queries embedded in wiki pages that filter and display the information on individuals and resources matching a specific set of constraints.

The benefits of SMW have been largely discussed in contexts characterized by a rapid evolution of knowledge [45]. Professional literature reports a variety of projects where SMW has been used to support knowledge-intensive processes in areas like: biomedical engineering [32], medicine [51], earth sciences [29, 50], emergency management [7], product design [34, 58, 76, 78], or innovation management [13, 42]. Conclusions of these reports demonstrate that SMW has become one of the preferred choices to support knowledge management in the extended enterprise [37, 41], collaboration around business process [54], and new forms of cooperation based on social web technologies [64, 67].

Software engineering literature also reports experiences with SMW to support development activities: Happel and Seedorf [31] proposed the use of the Ontobrowse ontology and semantic wiki to align knowledge needs of developers and software architects when describing software architecture. In the context of agile practices, Rech and Bogner [59] included wikis as one of the "human-centered knowledge-sharing platforms" suitable to support knowledge management in software development. The use of SMW to build a library catalog and organize project documentation was identified by Ribaud and Saliou [60, 61] and Baumeister et al. [5]; De Graaf [11] with his ArchiMind system and Tang et al. [71] showed the benefits of using SMW as a collaborative platform for managing software documentation, and Kleiner et al. [38] demonstrated the potential of SMW on incident and configuration management. The feasibility of SMW to support computer science learning and teaching was analyzed by Coccoli et al. [10]. Greaves [30] highlighted the value of the experience acquired in the deployment of SMW-based solutions to deal with the complex requirements of software for collaborative sense-making environments.

In the specific area of requirements engineering, Abeti et al. [1] developed WikiReq to support distributed requirements elicitation and requirements-centered discussions in business process reenginering projects. Liang et al. [46] proposed the use of SMW to support the analysis of requirements by distributed teams in large-scale projects, taking advantage of the SMW reasoning capabilities. In a similar approach, Nordheimer et al. [55] discussed the benefits that small and medium enterprises can obtain with the deployment of SMW to document requirements, architectural design, and keep traceability between them. Other contributions that discussed the use of SMW to keep functional and product requirements are those from Ma et al. [48], who proposed a metamodel to support consistency checks, and Sillaber et al. [69], who designed a platform to deal with the problems derived from the coexistence of different platforms for requirements elicitation. Stateli's ReqWiki platform [65]—an open source solution suitable for small and midsize software companies—incorporated natural language processing to improve the specification of requirements. A practical experience in the aerospace domain is provided by Favaro et al. [25] with its description of the NextGenRE project, sponsored by the European Space Agency (ESA), where SMW was selected for the collection of structured requirements and the implementation of coherence checks.

5.3 Architecture of the target solution

The proposed solution is built on top of SMW, which is based on the PHP programming language. SMW extends MediaWiki software by adding additional tables to the database repository (SQL store), a SPARQL store to support semantic queries, and additional hooks to allow developers to extend the standard functionality and use SMW data from external tools. From a functional perspective, in SMW, a distinction is made between: (a) annotation components (for content editing and manual tagging), (b) dynamic page components (for data browsing and the generation of display formats), (c) querying and searching components, (d) external interfaces for data import/export components and, (e) maintenance tools [40].

The target solution needs to incorporate the following items:

(a) A set of process configuration files (PCFs) to describe the process items (activities, tasks, work products, etc.) used to contextualize the process requirements. The PCFs will have separate data items for each process item. These items shall be tagged with the SPEM metadata defined by the process analyst.

(b) A software component (PCFs import) to process XML-encoded SPEM models, generate the PCFs in a format suitable for SMW, and inject them into the SMW database. This component's responsibility includes the incorporation of inline queries into the PCFs.
(c) A software component (ReqsImport) to generate content pages for process requirements, tag them with their related tasks, work products and other information, and inject them into the SMW database.
(d) A dictionary to support the automatic tagging of process requirements and classify them by activity, task, and work product. The dictionary contains equivalent and related terms for each process element (activity, tasks, and work product). The dictionary is kept in a separate XML file, out of the SMW database.

Components (b) and (c) make use of XSLT stylesheets to convert input data generated from the Eclipse Process Framerwork, MS Word, or IBM DOORS into an HTML format suitable for being imported into SMW. They also use SMW batch import capabilities for loading the data into the SMW database. ReqsImport component makes use of an external web service for term extraction: AlchemyAPI.[2] This parser has shown good performance in the extraction of compound terms, made up of several words [75]. Extracted terms are used as an input to classify requirements by activity and work product by matching them with the terms in the dictionary.

Figure 3 shows the main items in the solution architecture:

5.4 Implementation of the demonstrator

The implementation followed these steps:

- Creation of a visual representation of the ISO/IEC 29110 Basic Profile's processes using the SPEM notation, including the descriptions provided in the standard.
- Task and work products defined in ISO/IEC 29110 were tailored to the corporate needs of the company using the SPEM extension and customization capabilities. The resulting diagrams and related content were uploaded into the SMW site.
- Process requirements from customer and industry standards—as well as individual base practices from the corporate procedures—were stored as independent content units in the SMW site. These requirements were also tagged with the identifiers of tasks and work products. The terms extracted from the requirement descriptions were compared with those in the dictionary to deduce which task and work product should be used to tag the requirement. Automatic tagging showed a correctness close to 76 %. For those requirements where it was not possible to identify one related task or work product, a TBC value was used, to allow their later identification and manual tagging. Additional tags were used to indicate the projects where the standards were applicable and the source document where they are defined.

Figure 4 shows a schema with the main activities in the implementation process.

During this process, SMW features were used:

1. Import capabilities for loading the ontology containing the process model and the process requirements into SMW. The ontology constituted the context where process requirements were later allocated.
   Import capabilities also made it possible to load sets of process requirements. For each requirement, a separate SMW page was created, containing its orthogonal categorization using the ontology's properties: workProduct, developmentTask, criticalityLevel, etc. Preprocessing of the documents containing the process requirements was needed to generate a valid XML suitable for SMW (Fig. 5).
2. SVG representations of the processes were created in order to have a visual map to explore and browse the content of the SMW site (Fig. 6).
3. Content editing capabilities were used to incorporate changes into the process elements descriptions and requirements pages after bulk upload/import.
4. Inline queries were incorporated into the pages corresponding to tasks and work products. These queries retrieve, in a dynamic way, the requirements tagged with the identifiers of those tasks and work products. The fragment below shows an example of an inline query (Fig. 7):

```
{{      #ask:      [[Category:Requirement]]      [[activity::SI_3_4]]
|                          mainlabel=Req.                          Id.
|                                 ?requirementDescription=Description
|                                           ?requirementSource=Source
}}
```

5. Subqueries on categories were used to deal with generalization and specialization of classes, and to

[2] See http://www.alchemyapi.com/. Last visited on 23/03/2016.

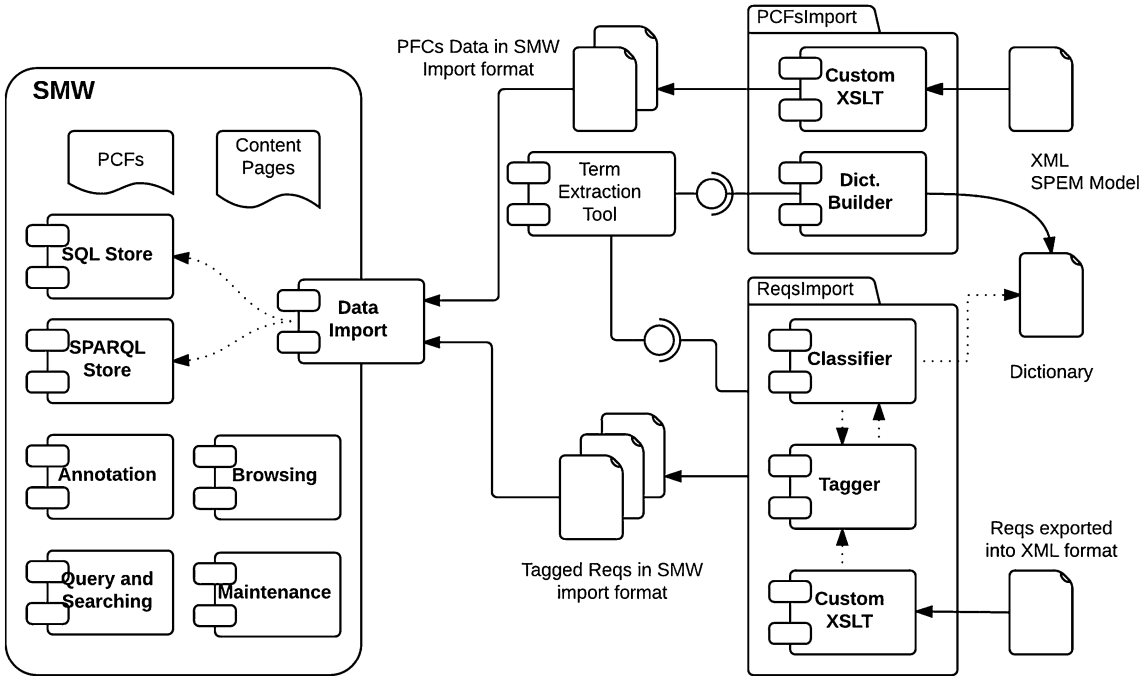
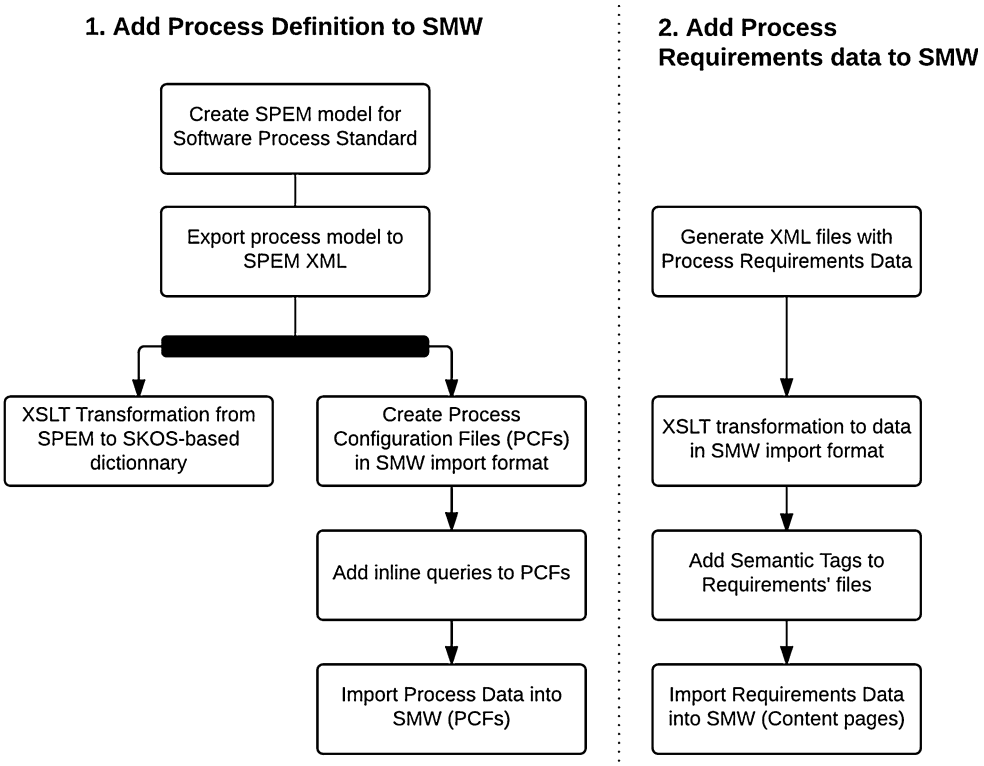


Fig. 3 Proposed architecture

Fig. 4 Implementation process



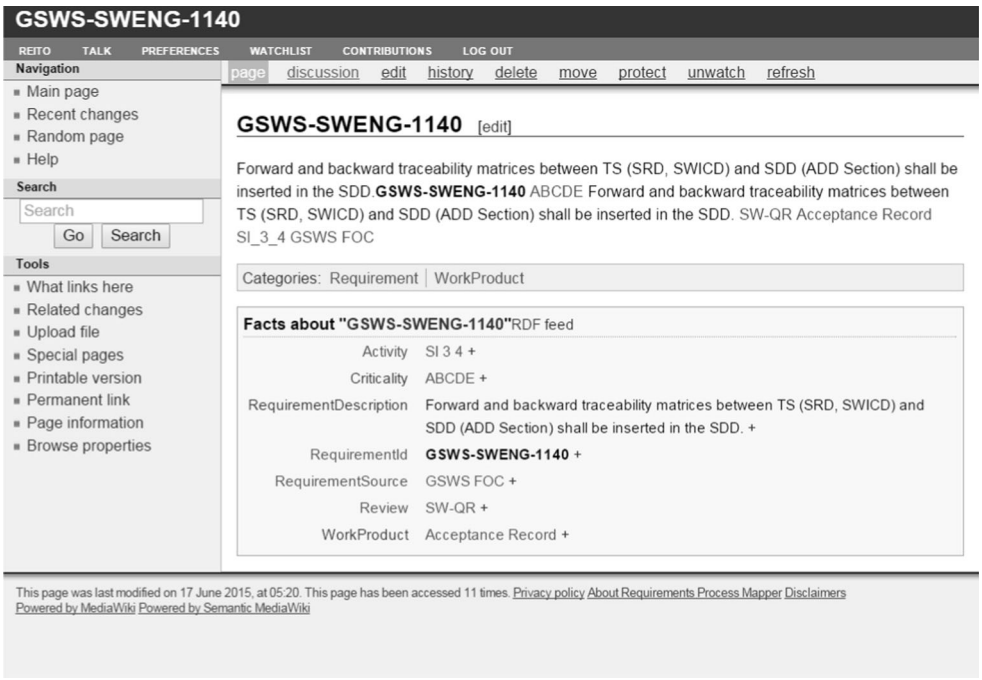


Fig. 5 Process requirement's metadata

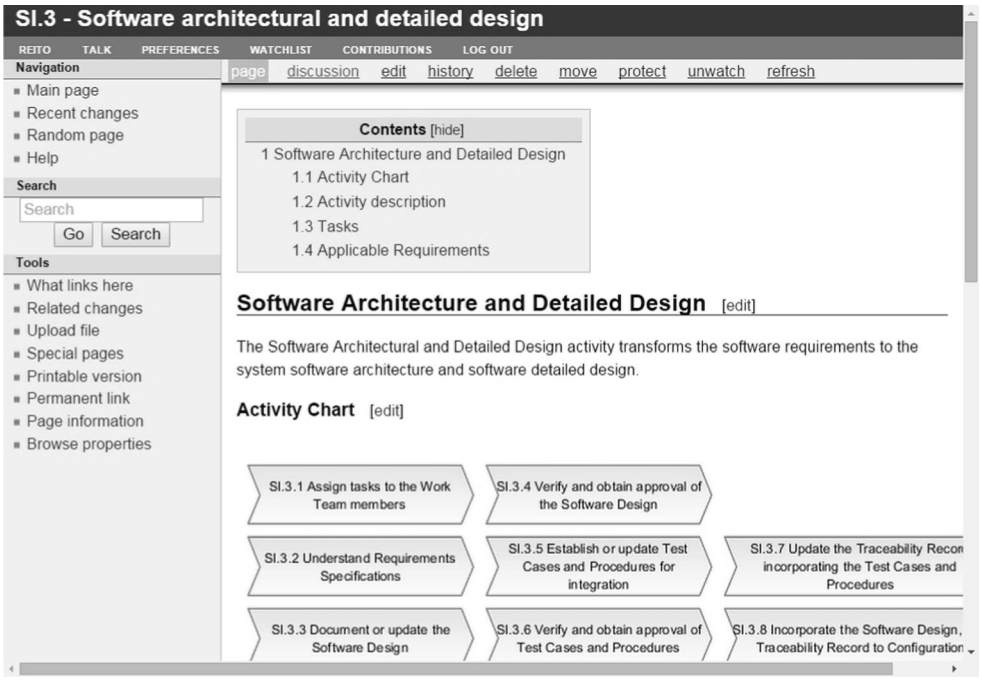


Fig. 6 Graphical representation of activities

group together the specializations for a given concept. As an example, this query extracts the requirements allocated to the different types of test reports:

```
[[Category:WorkProduct]] [[isSpecializatioOf::TestReport]]

[[Category:Requirement]]  [[workProduct::<q>[[Category:WorkProduct]]
[[isSpecializationOf::TestReport]]</q>]]
```

6. Collaborative development of content. As an open, distributed solution for content management, users are able to add annotations to any item in a controlled way.

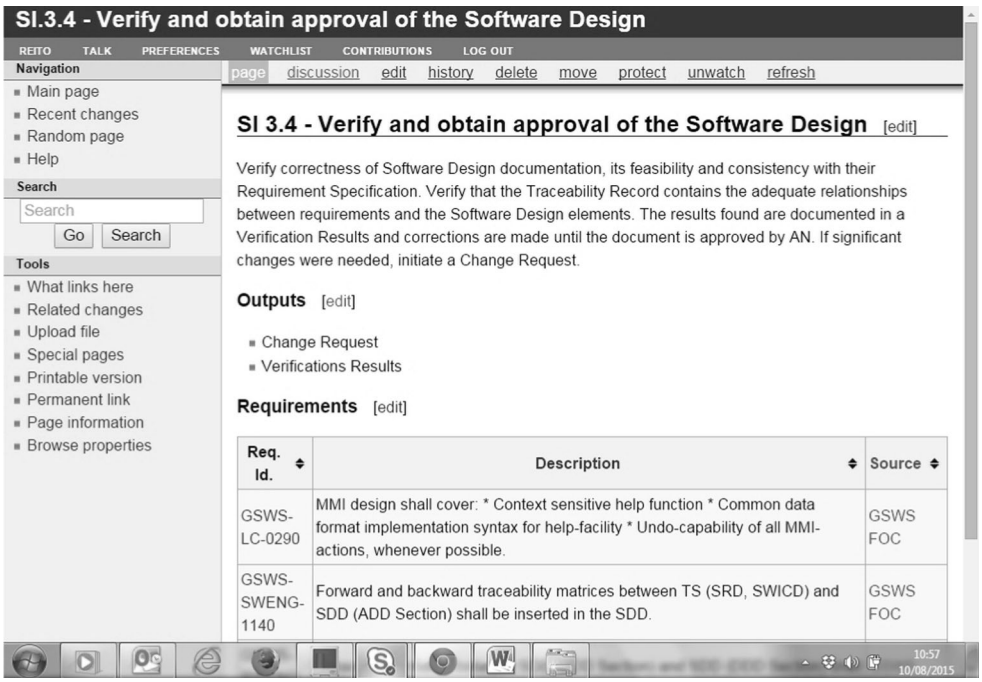


Fig. 7 Inline query showing requirements allocated to task

6 Validation and threats to validity

The proposed solution and its SMW-based implementation was verified and validated in the context of the participant company.

In addition to the verification of the requirements derived from its use cases (see Fig. 2), the tool validation was done through nine individual interviews (after direct observation of the users working with the tool) with staff playing engineering and management roles in different projects. Participants included four senior developers, two verification and validation engineers, two project managers, and one configuration manager. Validation aimed to check the feasibility of using the tool to give answers to common information needs related to process requirements. The staff who were interviewed were challenged to use the SMW site to answer typical questions they face in their daily activities:

- What content is expected for a particular document or deliverable?
- Is there a requirement on how to conduct particular activities (e.g., reporting, unit testing)?
- What is the format and delivery method for distributing the software and related documentation?
- What are the expected diagrams that need to be included in the design documentation?
- Which coding standards need to be enforced?
- What milestones should be delivered from a specific document?
- What is the source code coverage that is critical for a particular software?
- What deliverables should not be produced for a specific technical review when we are following a specific software life cycle (waterfall, incremental…)
- Which inspections and verification must be done when preparing the software acceptance?
- What records must be generated to provide evidence of the execution of a specific task?
- In which reports should I include the status of a specific type of item (software problems, risks, actions…)

To conduct validation testing, two data sets were created: one for the process requirements of the Galileo satellite constellation program, with a total of 3513 requirements, and another one for ESOC requirements, with a total of 2123 process requirements. Participants were not familiar with the process requirements in the sample data sets. They used the search and navigation features offered by SMW to answer those questions, assess the usefulness of the tool, and identify potential improvements. After completion of the exercise, participants were asked to answer a questionnaire with twelve questions related to one of the following assessment criteria:

(a) Information findability (to what extent does it help users find the required information);
(b) Information comprehension (to what extent does the tool help in understanding the data and putting it into context);

(c) Content editing capabilities (to what extent can users contribute with additional information, annotations, and tagging);
(d) Collaboration support (to what extent does the tool leverage team collaboration and knowledge reuse opportunities);
(e) Usability and interactivity of the tool.

Figure 8 shows the results of the average score given by the participants to the criteria:

The answers to the questionnaires were later discussed in a short meeting, and the results were presented to all the participants in a separate session. Conclusions can be summarized as follows:

- The proposed solution was recognized as a valuable tool to reduce the complexity of dealing with the process requirement sets applied in the aerospace industry. The possibility of finding information quickly and the fact that information is shown in the context of the tasks where it is needed was the most valued feature.
- The conceptual arrangement of requirements, classified by process, activities, tasks, and work products defined in the frame of an ontology, was judged of interest in dealing with the diversity of terms and acronyms used in different standards and guidelines.
- Semantic wiki's user-friendliness and its low learning curve were also considered to be beneficial. This includes the easy addition and tagging of content. Four participants remarked the need to establish controls to ensure that requirements are properly tagged.
- Collaboration support was valued as a positive feature, although participants gave this criterion the lowest score. Interviews demonstrated that the tool was perceived as a static repository of process requirements, a place to easily find the rules and constraints that should not be overlooked. But the possibility of using it as a collaboration platform was considered of secondary value.

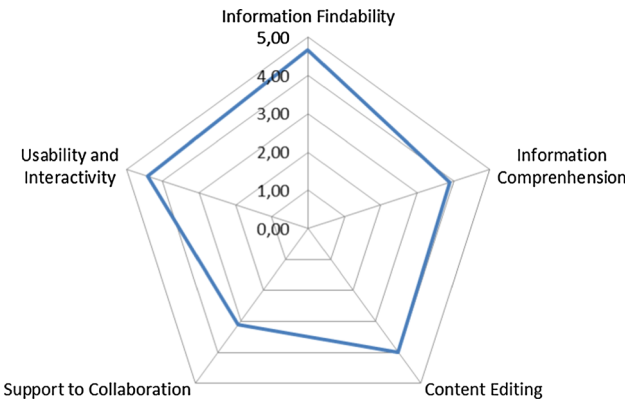


Fig. 8 Results of the assessment

The assessment of the proposed tool can be considered successful, but there are still some validity concerns. The main one is related to the performance of the classifier in charge of automatically tagging the requirements. Although the proposed approach (matching key terms with terms in a dictionary) provided satisfactory results, the automatic tagging of all the requirements was not achieved, and some of them needed to be tagged manually. Different approaches can be explored to overcome this limitation, e.g., using term similarity between requirements to assign the same tags, or include in the tagging process additional information like the sections' and subsections' titles where the requirements are located in the source documents. In any case, even if the tagging process requires some manual intervention, the benefits and time savings that can be achieved justify this additional work.

Another concern is related to the portability of the solution to companies or projects using a different software process model. In the case study, the ISO/IEC 29110 has been used as a reference, but the proposed approach can be easily applied to any other scenario using a different software process model. The flexibility of the SPEM modeling language and its openness to represent any process model ensures this portability.

A final concern was related to the cost of setting up the proposed solution. The trade-off between the implementation costs and the solution benefits (time savings when looking for information and exhaustiveness of data) was discussed with the validation team. Costs were considered justified in the case of long-term contracts and programs that impose common process requirements to different projects: a habitual situation in the aerospace industry.

7 Conclusions

Standards like ISO/IEC 29110 or ISO/IEC 12207 provide companies with process descriptions that may be used to establish tailored processes. But software development companies need to combine their internal process guidelines with additional requirements coming from their customers or industry-specific standards. The activities, tasks, and work products defined in the corporate processes constitute the activity context where process requirements must be understood, applied, and verified. Process requirements are usually spread through different specifications and documents provided by the client as part of the tender conditions or the statement of work. Dealing with requirements documented in different sources implies the risk of disregarding relevant information. This complexity

makes access to information difficult and has a negative impact on productivity.

This paper proposes an approach to aggregate, represent, and give access to complex sets of process requirements. Built on top of the SMW content management platform, staff involved in software development are given a single point of access where they can search and identify the constraints that must drive the execution of the tasks and the generation of work products. Process models are needed to put requirements into context. When working on a specific deliverable or task, engineers can easily get the list of requirements that need to be considered and get a clear understanding of all the applicable constraints. The tool can also be used to ensure an adequate assessment of work products with respect to the expected characteristics. The approach has been validated with the implementation of a demonstrator. Participants recognized that it was a useful solution to avoid missing relevant information and ensure compliance with both internal and external process requirements.

References

1. Abeti LA, Ciancarini PB, Moretti RB (2009) Wiki-based requirements management for business process reengineering. In: International conference of software engineering, pp 14–24
2. Alexis O (2009) Rationale modeling for software process evolution. Softw Process Improv Pract 14(2):85–105
3. Ali R, Dalpiaz F, Giorgini P (2010) A goal-based framework for contextual requirements modeling and analysis. Requir Eng 15(4):439–458
4. Aoussat F, Oussalah M, Ahmed M (2014) A spemontology for software processes reusing. Comput Inform 33(1):35–60
5. Baumeister J, Reutelshoefer J, Puppe F (2011) KnowWE: a semantic Wiki for knowledge engineering. Appl Intell 35(3):323–344
6. Boucher Q et al (2012) Towards configurable ISO/IEC 29110-compliant software development processes for very small entities. In: Winkler D, O'Connor R, Messnarz R (eds) Systems, software and services process improvement, Springer, Berlin, Heidelberg, pp 169–180
7. Caglayan A et al (2012) Semantic technologies for civil information management during comlex emergencies. In: IEEE international conference on technologies for homeland security, HST, pp 523–528
8. Calero C, Ruiz F, Piattini M (eds) (2006) Using ontologies in software engineering and technology. In: Ontologies for software engineering and software technology. Springer, Berlin, pp 49–102
9. Cherry C, Macredie RD (1999) The importance of context in information system design: an assessment of participatory design. Requir Eng 4(2):103–114
10. Coccoli MA, Vercelli GA, Vivanet GB (2013) Semantic Wiki for learning and teaching computer science. J E-Learn Knowl Soc 9(2):173–183
11. De Graaf KA (2011) Annotating software documentation in semantic wikis. In: International conference on information and knowledge management, pp 5–6
12. Eito-Brun R (2004) El contexto de la información. Herramientas y útiles para el proceso de auditoría. El Profesional de la Información 12(4):302–312
13. Elkaffas SM, Wagih AS (2013) Use of semantic wiki as a capturing tool for lessons learned in project management. In: Proceedings of teh 2013 science and information conference, SAI, pp 727–731
14. ESA (2005) Tailoring of ECSS software engineering standards for ground segments in ESA. Part A–D(**) BSSC 2005(1)
15. ESA (2009a) Galileo software standard (GAL-SPE-ESA-SYST-0092). Issue 1.1
16. ESA (2009b) Galileo FOC product assurance and safety requirements (GAL-REQ-ESA-GMS-X/0104). Issue 1.0
17. ESA (2010) Galileo FOC—system verification requirements document (GAL-REQ-ESA-SYST-X-0017). Issue 1.2
18. ESA (2011a) Galileo FOC management requirements (GAL-MGT-ESA-SYST-X/0001). Issue 2.2
19. ESA (2011b) Galileo FOC configuration and data management requirements (GAL-MGT-ESA-SYST-X/0002). Issue 3.1
20. ESA (2013) Requirements for delivery of documents to ESA (GAL-REQ-ESA-SYST-X/0073). Issue 1.2
21. ESA-OPS (2009a) Tailoring of ECSS-E-ST-40C for space engineering—software (QMS-EIMO-GUID-CKL-9500-OPS). Issue 1.0
22. ESA-OPS (2009b) Tailoring of ECSS-Q-ST-80C for space product assurance—software product assurance (QMS-EIMO-GUID-CKL-9501-OPS). Issue 1.0
23. ESOC (2006) Software quality and coding rules (EGOS-QA-XX-TN-9007). Issue 1.4
24. ESOC (2008) ESOC generic ground systems: development requirements specification (EGGS-ESOC-GS-SRS-1001). Issue 1.1.1
25. Favaro JA et al (2012) Next generation requirements engineering. In: 22nd annual international symposium of the International Council on Systems Engineering, INCOSE, vol 1, pp 479–507
26. Freund L, Toms EG, Waterhouse J (2005) Modeling the information behaviour of software engineers using a work—task framework. In: Proceedings of the American Society for Information Science and Technology, vol 42(1). doi:10.1002/meet.14504201181
27. Fuentes-Fernández R, Gómez-Sanz J, Pavón J (2010) Understanding the human context in requirements elicitation. Requir Eng 15(3):267–283
28. Gazel S, Sezer EA, Tarhan A (2012) An ontology based infrastructure to support CMMI-based software process assessment. Gazi Univ J Sci 25(1):155–164
29. Gordon SN et al (2014) Studying the use of forest management decision support systems: an initial synthesis of lessons learned from case studies compiled using a semantic wiki. Scand J For Res 29:44–55
30. Greaves M (2014) Wikis, semantics, and collaboration: symposium on collaboration analysis and reasoning systems, at the 2014 conference on collaboration technologies and systems. In: International conference on collaboration technologies and systems (CTS), pp 469–471. doi:10.1109/CTS.2014.6867607
31. Happel H, Seedorf S (2007) Ontobrowse: a semantic Wiki for sharing knowledge about software architectures. In: 19th international conference on software engineering and knowledge engineering SEKE, pp 506–512
32. He S et al (2009) Collaborative authoring of biomedical terminologies using a semantic wiki. In: AMIA symposium, pp 234–238
33. Herzig DM, Basil E (2010) Semantic MediaWiki in operation: experiences with building a semantic portal. In: Lecture notes in computer science, 6497, pp 114–128

34. Huang YA et al (2015) A semantic-based visualised wiki system (SVWkS) for lesson-learned knowledge reuse situated in product design. Int J Prod Res 53(8):2524–2541
35. IEEE Computer Society (2015) Guide to the systems engineering body of knowledge (SEBoK), version 1.5.1. http://sebokwiki.org/
36. International Organization for Standardization (2014) ISO/IEC 24744:2007—software engineering—metamodel for development methodologies
37. Jung JJ (2013) Semantic wiki-based knowledge management system by interleaving ontology mapping tool. Int J Softw Eng Knowl Eng 23(1):51–63
38. Kleiner FA, Abecker AB, Mauritczat MC (2012) Incident and problem management using a semantic Wiki-enables ITSM platform. In: 4th international conference on agents and artificial intelligence, vol 1, pp 363–372
39. Kluza K, Nalepa GJ, Lisiecki J (2014) Square complexity metrics for business process models. Adv Intell Syst Comput 257:89–107
40. Krötzsch M, Vrandečić D, Völkel M (2006) Semantic Mediawiki. In: Lecture notes in computer science, ISWC, Springer, Berlin, pp 935–942. doi:10.1007/11926078
41. Lahoud IA, Monticolo DB, Hilaire VC (2014) A semantic wiki to share and reuse knowledge into extended enterprise. In: 10th international conference on signal-image technology and internet-based systems, SITIS, pp 702–708
42. Lahoud IA et al (2013) A semantic wiki to support knowledge sharing in innovation activities. In: Lecture notes in electrical engineering, vol 186, pp 217–230
43. Laporte C, Fanmuy G, Ptack K (2012) The development of systems engineering international standards and support tools for very small enterprises. In: 22nd annual international symposium of the international council on systems engineering, INCOSE 2012 and the 8th biennial European systems engineering conference, vol 3, pp 1563–1590
44. Laporte C, O'Connor R, Fanmuy G (2013) International systems and software engineering standards for very small entities. CrossTalk 26(3):28–33
45. Leclercq É, Savonnet M (2012) Système d'information pour la production de connaissances: L'approche wiki sémantique. Ingénierie des Systèmes d'Information 17(3):143–166
46. Liang PA, Avgeriou PA, Clerc VB (2009) Requirements reasoning for distributed requirements analysis using semantic wiki. In: 4th IEEE international conference on global software engineering, ICGSE, pp 388–393
47. Liska M, Navrat P (2011) SPEM ontology as the semantic notation for method and process definition in the context of SWEBOK. Comput Sci Inf Syst 8(2):299–315
48. Ma J et al (2012) Using a semantic wiki to improve the consistency and analyzablit of functional requirements. Commun Comput Inf Sci 319:460–473
49. Maalej W (2011) Context aware software engineering and maintenance: the FastFix approach. http://es.slideshare.net/maalejw/context-aware-software-engineering-and-maintenance-the-fastfix-approach
50. Marques AF et al (2014) Collaborative development of a semantic wiki on forest management decision support. Scand J For Res 29:30–43
51. Meilender TA et al (2012) A semantic wiki for editing and sharing decision guidelines in oncology. Stud Health Technol Inform 180:411–415
52. Méndez Fernández D, Penzenstadler D (2015) Artefact-based requirements engineering: the AMDiRE approach. Requir Eng 20(4):405–434
53. Münch J et al (2012) Software process definition and management. Springer, Berlin
54. Nalepa GJ, Kluza K, Ciaputa U (2012) Proposal of automation of the collaborative modeling and evaluation of business processes using a semantic wiki. In: IEEE international conference on emerging technologies and factory automation. doi:10.1109/ETFA.2012.6489769
55. Nordheimer K, Seedorf S, Thum C (2012) Semantic wiki for tracing process and requirements knowledge in small and medium enterprises. In: Ivan M, Antony T, Rami B, Judith AS (eds) Aligning enterprise, system, and software architectures, IGI Global, Hershey, PA, pp 23–38
56. O'Connor R, Laporte C (2012) Software project management in very small entities with ISO/IEC 29110. Commun Comput Inf Sci 301:330–341
57. OMG (2008) Software & systems process engineering metamodel specification (SPEM) version 2.0. Technical report ptc/07-03-03, Object Management Group
58. Pereira CA, Sousa CA, Soares AL (2013) Supporting conceptualization processes in collaborative networks: a case study on a R&D project. Int J Comput Integr Manuf 26(11):1066–1086
59. Rech JA, Bogner CB (2010) Qualitative analysis of semantically enabled knowledge management systems in agile software engineering. Int J Knowl Manag 6(2):66–85
60. Ribaud V, Saliou P (2010) Process assessment issues of the ISO/IEC 29110 emerging standard. In: ACM international conference proceeding series, pp 24–27
61. Ribaud V, Saliou P (2010) Using a semantic Wiki for documentation management in very small projects. In: Sánchez-Alonso S, Athanasiadis I (eds) Metadata and semantic research. Springer, Berlin, pp 119–130
62. Runeson P et al (2012) Case study research in software engineering: guidelines and examples. Wiley, Hoboken
63. Ruíz-Rube I et al (2013) Uses and applications of software & systems process engineering meta-model process models. A systematic mapping study. J Softw Evol Process 25(9):999–1025
64. Sanna GA et al (2015) A semantic social bookmarking system based on a wiki-like approach. In: Lecture notes in electrical engineering, vol 330, pp 533–538
65. Sateli B, Angius E, Witte R (2013) The ReqWiki approach for collaborative software requirements engineering with integrated text analysis support. In: International computer software and applications conference, pp 405–411
66. Steenweg R, Kuhrmann M, Méndez-Fernández D (2012) Software engineering process metamodels: a literature review. TUM (Forschungsbericht; TUM-I1220), München. https://mediatum.ub.tum.de/?id=1128389. Last visited 25 Mar 2016
67. Schatten M (2013) Knowledge management in semantic social networks. Comput Math Organ Theory 19(4):538–568
68. Sharma A, Kushwaha DS (2010) Complexity measure based on requirement engineering document and its validation. In: Computer and communication technology (ICCCT), 2010 international conference, pp 608–615
69. Sillaber C, Chimiak-Opoka J, Breu R (2012) Supporting social driven requirements processes through knowledge sharing platforms. In: Proceedings of the IASTED international conference on software engineering, pp 60–66
70. Šmite D et al (2014) An empirically based terminology and taxonomy for global software engineering. Empir Softw Eng 19(1):105–153
71. Tang A, Liang P, Van Vliet H (2011) Software architecture documentation: the road ahead. In: 9th working IEEE/IFIP conference on software architecture, WICSA, pp 252–255
72. Termité T, Kuhrmann M (2009) Das v-modell xt 1.3 metamodell. TUM (Forschungsbericht; TUM-I0905), München. https://www4.in.tum.de/publ/papers/tk09.pdf. Last visited 25 Mar 2016
73. Turner M (2006) Microsoft® solutions framework essentials: building successful technology solutions. Microsoft Press, Redmond. ISBN 9780735623538

74. Villela K et al (2005) The use of an enterprise ontology to support knowledge management in software development environments. J Braz Comput Soc 2(11):45–59
75. Wijnhoven F, Brinkhuis M (2015) Internet information triangulation: design theory and prototype evaluation. J Assoc Inf Sci Technol 66(4):684–701. doi:10.1002/asi.23203
76. Zapp MA et al (2012) Collaborative machine tool design environment based on semantic wiki technology. In: Proceedings of the European conference on knowledge management, ECKM, pp 1583–1586
77. Zhaoa Y (2008) High value information in engineering organisations. Int J Inf Manag 28(4):246–258
78. Zhu L, Jayaram U, Kim O (2011) Online semantic knowledge management for product design based on product engineering ontologies. Int J Semant Web Inf Syst 7(4):36–61. doi:10.4018/jswis.2011100102