Towards Collaborative Requirements Engineering Tool for ERP Product Customization

Boban Celebic, Ruth Breu, Michael Felderer and Florian Häser

Abstract Requirements Engineering (RE) is the foundation for efficient software quality management. It is a cumbersome and complex task, particularly in the context of complex software products such as ERP systems, since it has to deal with numerous and specific challenges and large number of requirements to develop successful product, and therefore requires a systematic and collaborative approach. Tools which support RE in general are numerous nowadays; however, the task of providing a tool that specializes in RE for dynamic, customizable service-centric systems has been addressed seldom. In this sense, the result of our effort to provide such a tool—a support tool for collaborative requirements engineering and software artifacts linking (traceability), with focus on ERP product development and customization—is presented in this short paper. This tool was developed based on results of an analysis of challenges for RE in a highly dynamic ERP environment—these challenges were identified by performing survey of literature and through intensive discussion with our industry partner.

Keywords Collaborative requirements engineering · Enterprise resource planning (ERP) · SME · Product customization · Stakeholder collaboration

B. Celebic (✉) · R. Breu · M. Felderer · F. Häser
Institute of Computer Science, University of Innsbruck, 6020 Innsbruck, Austria
e-mail: boban.celebic@uibk.ac.at
URL: http://www.uibk.ac.at

R. Breu
e-mail: ruth.breu@uibk.ac.at

M. Felderer
e-mail: michael.felderer@uibk.ac.at

F. Häser
e-mail: florian.haeser@uibk.ac.at

F. Piazolo and M. Felderer (eds.), *Novel Methods and Technologies for Enterprise Information Systems*, Lecture Notes in Information Systems and Organisation 8,
DOI: 10.1007/978-3-319-07055-1_9,

1 Introduction

An Enterprise Resource Planning (ERP) system is a business management software that enables enterprises to bind all common data, practices, organizational units and activities across an enterprise into a single unified system, in order to achieve better performance and smooth workflow and to produce, process and access information in a real-time environment. ERP implementation is the customization, configuration and integration of an ERP system into an enterprise (company, organization).

Nowadays, ERP systems, in essence, represent highly complex networks of components and services. Aligning these services to specific needs of customer is crucial for the overall success of the system implementation. Unfortunately, implementing an ERP system into several enterprises can result with a complex history of product releases, industry-specific solutions or configurations specific to individual customers. This makes requirements engineering an even more troublesome process and demanding task than usual; yet, it is still essential for efficient quality management and successful ERP implementation [1].

On the other side, most Requirements engineering and management tools available nowadays, in addition to being decoupled from the development process, do not stimulate actively involvement and interaction among stakeholders. There are numerous such RE tools, both commercial and non-commercial, currently available on the market, but, based on our literature and online market research, very few of those focus on RE in the context of customizable ERP systems. In addition, most of these tools have other drawbacks as well. For example, difficult learning curve is one similar trait of most tools. Some require installation on client side. Of all the tools that we have reviewed, the most similar approach to ours is the WinWin methodology and support tool [2]. WinWin is a collaborative system requirements elaboration and negotiation tool. It integrates the group productivity techniques and some collaborative tools. It has support for stakeholder cooperation, requirements prioritization, issues management; glossary of terms is included as well, which is similar to our own tool. However, the training/learning curve is rather difficult for average stakeholder and requires additional effort, which undermines the collaborative aspect of this tool. Authors had to develop a lightweight version of their tool to ease the difficult learning process: EasyWinWin. Another difference from our tool is that WinWin doesn't have such strong support for collaboration with external systems (issue trackers, tests suites), which also undermines the collaborative aspect of the tool and makes the product customization process more difficult.

As a result of previous observations, our research addressed, distinctively, collaborative aspects of several different success-critical *stakeholders* involved; their roles will be explained in more detail in following sections.

In short, this short paper aims at advancing the field of requirements engineering for dynamic service-centric systems (ERP systems particularly) by sketching a novel approach and support tool which focus on project development

and, more specifically, *product customization* management and linking, traceability and visualization of software artifacts. The framework and its tool, developed on the basis of these ideas, are our main contribution to this field of research.

The rest of the paper is organized as follows. Section 2 gives a brief introduction to specific problems of requirements engineering in the context of customizable ERP systems, with focus on the collaborative aspect of requirements knowledge management process. Section 3 gives a sketch of the tool and solution framework and then reports the feedback from an ERP expert of our industry partner. Finally, we conclude and present future work.

2 Challenges for Requirements Engineering in the Context of Customizable Software (ERP) Products

Unfortunately, the ERP implementations failure rate keeps high [3]. Thus, numerous studies have been conducted in order to find and categorize all the challenges that ERP system implementation projects face. In this section we address the challenges to the requirements engineering in the context of (ERP) product customization and implementation, with additional focus on collaborative aspect of various involved stakeholders—namely: *business analyst* (often called *consultant*), the representative of the ERP vendor who discusses requirements for customized product with the customer and has to identify the collection of requirements which both satisfy the needs of the customer and makes the implementation efficient (thus, business analyst is usually a requirements engineer); *product manager*, who is responsible for development of the software product and plans its releases; *project manager*, who is responsible for the implementation of the customized product; *customer*, who purchases the ERP system implementation; *developer*, who develops the system and sometimes customizes it to specific needs of the customer; and finally *tester*, who develops test cases with aim to check the functionality and proper alignment of the product to specific needs and requirements. These challenges are result of our extensive literature review, and were additionally filtered through intensive discussions with our industry partner. We have identified some of these challenges in an earlier paper [1] and took them into strict consideration while designing our approach and framework.

- Collaborative requirements knowledge management

One of the challenges for RE is to bring various forms and representations of knowledge about requirements into conformance. On one hand, information about requirements can be held in more or less (un)structured textual form, like office documents; such documents are usually result of requirements negotiation between business analyst and customer. On the other hand, project and product managers, as well as developers and testers, need more concise and formal representation of

requirements and knowledge about software product (e.g. in form of models or class diagrams), in order to perform their tasks in satisfying manner.

- Software artifacts traceability and Change management

For achieving successful product customization and implementation, as well as efficient quality management/control, stakeholders need the ability to trace changes in requirements and their manifold interdependencies with other software artifacts (issues and customer requests, tests, risks), as well as their realization (e.g. release, branch-specific product) and state (implemented, under development, deprecated). This tracing ability is also a cornerstone for effective change management, since it helps with tracking changes (e.g. in standard product, requirements, other artifacts) and propagating them further (e.g. change of requirement needs to be reflected in the product implementation) in order to maintain actuality of requirements.

- Quality of requirements

Besides the consistency of product functionality with requirements, mentioned in previous challenge, quality attributes (e.g. completeness, stability, verifiability, comprehensibility) are also crucial for efficient requirements reuse. Only when these quality attributes are met, the requirements quality will be sufficient for successful requirements knowledge management. There are several approaches to model quality assessment which can be used to assess the quality of requirements (e.g. [4] for model-based requirements and [5] for textual requirements).

- Problems related to products and services

New challenges are emerging lately as a result of introduction of services in the cloud, increasing the need for the flexibility of the customization process. Similarly, customer-specific services demand for more flexible composition as well. These services may have many variants (e.g. for different industries) or are even customer-specific, which makes the requirements management an even more complicated task.

3 Tool Implementation

As stated before, we have developed a novel framework for Requirements Engineering in the context of customizable service-centric systems. In this section we sketch our framework and support tool, which is addressing the challenges mentioned in preceding sections. In the following paragraphs, we explain the framework in more detail.

The framework has been conceptualized in lively discussion with experts from our industry partner in this project. Our primary focus was to support collaboration of various stakeholders—our framework, thus, provides a front-end for these stakeholders to create and manage requirements artifacts. We took into account

that a product like an enterprise resource management system today is a complex network of services. These services may have many branch- or industry-oriented versions (variants) or are even customer-specific.

Typical scenarios we decided to consider are:

- Consultants authoring customer-specific requirements, looking for similar requirements having been implemented for other customers.
- Product managers planning releases and variants on the basis of various internal and external requests.

After thorough discussion and literature research our concept has comprised the following aspects:

- A Requirements Knowledge Base describing a business oriented view of the product as a base of interrelated requirements
- Support for traceability among artifacts related with product requirements like issues (e.g. customer requests, bugs), test cases and the product components
- A change-driven lifecycle model where every data element (like a requirement, a product component or an issue) has an associated lifecycle state which may change over time (e.g. a requirement being productive or deprecated)
- An evaluation-controlled process by systematic assessment of data elements (e.g. attaching requirements, test cases and product components with risk categories and risk values)
- To provide each stakeholder with the appropriate view on the Requirements Knowledge Base to support the stakeholders tasks (following the principles of view-based software engineering); this e.g. means that consultants are provided with interfaces where text can be easily edited, product managers get graphical charts to bundle and abstract requirements, whereas developers interact based on textually described models
- To support users of the system by recommendations derived from the central Requirements Knowledge Base (e.g. proposing links between artifacts).

In the following we describe the current status of the tool prototype, followed by an outlook on next steps.

3.1 Meta Model

The (initial) meta model of the framework is illustrated in Fig. 1. The meta model for conceptualizing requirements has been defined to comprise business process definitions, use-cases, and non-functional requirements. Each artifact can be tagged and categorized to support its reusability. Based on the content of requirements artifacts and its tagging, recommendations for the reuse of requirements are provided. Additionally, the requirements artifacts have states and assigned requests. The requests represent requests from customers but also development requests which are linked to product artifacts like components they have

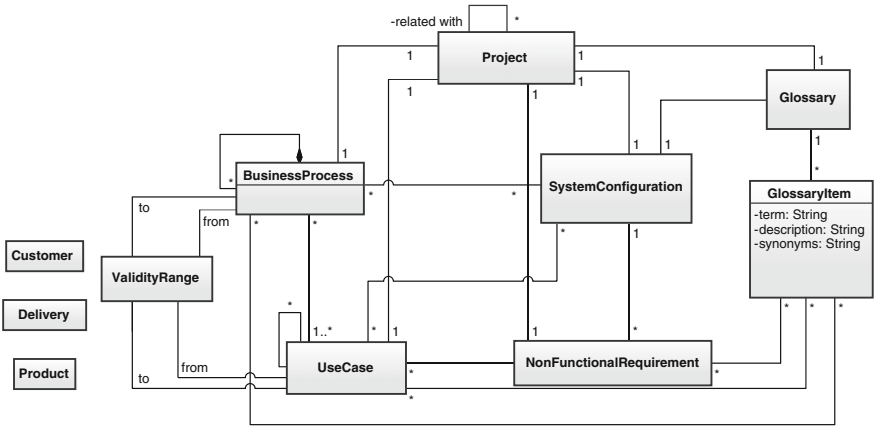

Fig. 1 Tool meta model

implemented or adapted. Support requests are also a valuable interface for Product Managers to plan future releases. If similar requirements occur several times in customizations then their functionality is a candidate to be included into a future release of the core product itself. Due to the internal linkage of the requirements artifacts to model elements representing the logical structure of the software product, change management, collaboration as well as the quality of the requirements can be guaranteed. Concerning the product, the project team has decided that product components are represented at the most abstract level by service hierarchies implementing specific product features. The data aspect is addressed through the concept of glossaries—collections of terms and their descriptions related to each other as homonym or synonym. Requirements may be attached with data imported from other systems. A bundling of requirements is supported by the concept of Owner representing a specific customer. In order to facilitate the collaboration between stakeholders, glossaries are attached to products.

3.2 Tool Functionality

The functionality of the current prototype can be shortly summarized as follows:

- Change-driven refinement process guiding the requirements engineer through the engineering phase, as well as involving various stakeholders into the process of ERP product customization. This allows flexible alignment with the dynamic business, architectural and technical requirements put in front of the specific ERP implementation.
- Support for creation and description of use cases, business processes, requirements and services, in web forms supporting both structured data in the

background (e.g. use cases attached with external files) and informal, unstructured, wiki-kind text editing facilities.
- Enhanced stakeholder collaboration (different types of stakeholders, their improved involvement and faster and more efficient interaction among them).
- Framework is optimized for processing large numbers of requirements.
- Support for all kinds of software artifacts and linking among them (linking among all types of artifacts in a repeatable process)—it provides tight coupling of requirements and other artifacts.
- Suitable/customized visualizations (trees, graphs, tables, diagrams, charts, matrices) of requirements and their prioritization, traceability links, tests results, risks and change impact—in order to support analysis, decision/strategy making and tracking of evolutionary change aspect.
- Support for issues—capability for importing issues from external Issue tracker systems and linking them to requirements.
- Support for test cases—capability for importing tests from external Test suites and linking them to requirements.
- Risk assessment support—with implemented risk assessment model.
- Portability—the tool can easily be adapted to various platforms (e.g. different operating systems, different RDBMS) by doing simple modifications.

3.3 Architecture

In order to enable easy collaboration among stakeholders, we decided to develop a web-based tool prototype; the tool can be accessed by any stakeholder through a web browser. The application allows multi-user access and is protected through role-based access control. Tool architecture is depicted in the following Fig. 2.

3.4 Outlook

The tool development and related research (e.g. comparison with tools available on the market, literature study) showed us the huge potential of the concept. In the subsequent iteration which is under current development we decided to extend our approach in the following way:

- Support of a flexible meta model through a model-driven approach

E.g. use case templates in the web forms may be easily extended by new text fields the web form is generated from the meta model;

- Extension by support for the definition of test cases;
- Strengthening of the view-based approach

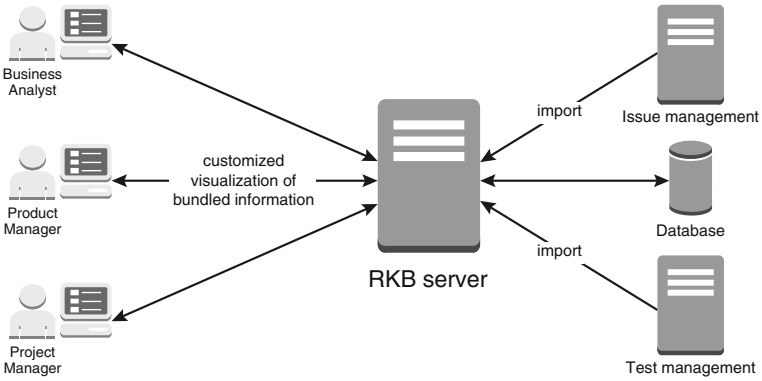

Fig. 2 Tool architecture. Requirements Knowledge Base (RKB) is the application server (application's logic)

Simultaneous work in different views, e.g. the framework supports language-oriented specification of test cases by the test engineer synchronized with text-editor like description of requirements by the business analyst/consultant;

- Flexible view generation with user-friendly interfaces

E.g. generating a matrix view relating selected requirements with selected product components; this requires the development of user-friendly view definition languages;

- Materialization of a generic assessment model

E.g. attaching requirements and product components with risk factors allowing tests to be prioritized; some risk values may be computed in an automated way, e.g. through call of runtime sensors or evaluation of static code metrics.

Opinion of an ERP expert As mentioned earlier, our research was done in cooperation with large European E-service provider which specializes in ERP systems for SMEs. The expert from our industry partner, in charge of the overall cooperation with our team, expressed satisfaction with our results and the framework itself, particularly with the web-based user interface, as well as with the strong portability potential. Some improvements were proposed; these proposals for improvement will certainly be taken into careful consideration during advanced phases of the tool development.

The prototype is available online for demo purpose (http://lr.q-e.at, access details are available on demand).

4 Conclusion and Future Work

This paper sketched challenges and our approach—RE support tool—to collaborative requirements engineering in the environment of customizable service-centric systems and ERP systems. Our framework supports the collaboration of various success-critical stakeholders: requirements engineers, project and product managers, business analysts, customers, etc., in the context of ERP system development and, particularly, product customization. This solution framework and support tool have been developed in collaboration with an ERP system vendor for small and medium-sized enterprises. The implementation phase will be followed by evaluation activities and further improvements of the framework.

Acknowledgements This work is supported by the project 'QE LaB—Living Models for Open Systems' (FFG 822740).

References

1. Breu, R., & Felderer, M. (2012). Challenges to requirements knowledge management of customizable software products. In *Modelling and Quality in Requirements Engineering: Essays Dedicated to Martin Glinz on the Occasion of His 60th Birthday.* Monsenstein und Vannerdat.
2. Gruenbacher, P. (2000). Collaborative requirements negotiation with easywinwin. In *IEEE Proceedings. 11th International Workshop on Database and Expert Systems Applications* (pp. 954–958).
3. Pacheco-Comer, A. A., & González-Castolo, J. C. (2012). An empirical study in selecting enterprise resource planning systems: The relation between some of the variables involve on it. size and investment. *Procedia Technology*, *3*, 292–303.
4. Chimiak-Opoka, J. (2011). Measuring uml models using metrics defined in ocl within the squam framework. In *Model driven engineering languages and systems* (pp. 47–61). Heidelberg: Springer.
5. Gervasi, V., & Nuseibeh, B. (2002). Lightweight validation of natural language requirements. *Software: Practice and Experience*, *32*(2), 113–133.