A Requirements Elicitation Approach for Cloud Based Software Product Line ERPs

Mohamed Ali
Department of Computer Science, Institute of Statistical Studies and Research, Cairo University, Cairo, Egypt
msc.moh.ali@gmail.com

Eman S. Nasr
Independent Researcher
Cairo, Egypt
nasr.eman.s@gmail.com

Mervat H. Gheith
Department of Computer Science, Institute of Statistical Studies and Research, Cairo University, Cairo, Egypt
mervat_gheith@yahoo.com

ABSTRACT

Implementing Enterprise Resource Planning (ERP) systems in business organizations aims to integrate all business units of an organization. Configuring and customizing ERP systems are the main challenges that face the implementation process. ERP systems contain many similar modules and units which can be implemented for most of the ERP systems. Software Product Lines (SPLs) as a trend in software engineering is very promising, as it can offer a lot of facilities and benefits for all types of stakeholders. Building SPLs for ERP systems will affect the implementation process of ERP systems and will increase the flexibility of configuration and customization. Moreover, moving ERPs to the cloud will facilitate the implementation process and will affect the Return On Investment (ROI) due to scalability plans in cloud services. This research introduces an SPLs requirements elicitation approach for cloud ERP systems. This approach combines the principles of SPLs with ERP systems in the cloud environment.

CCS Concepts

• Software creation and management→ Designing software →Requirements analysis • Software development techniques → Software prototyping Computing • Software and its engineering → Software product lines.

Keywords

Software product line; cloud computing; enterprise resource planning; systematic reuse; requirements elicitation; ecosystems; SaaS; ERP customization.

1. INTRODUCTION

Enterprise resource planning (ERP) is a software system that manages the business process of a company or an organization. ERP systems integrate all functional departments inside an organization. They provide also the ability for connecting an organization's departments with external parties who are involved in the organization's processes, such as customers and suppliers. Figure 1 illustrates a simple overview of an ERP system.

AMECSE '16, May 28-29, 2016, Cairo, Egypt.
© 2016 ACM. ISBN 978-1-4503-4293-3/16/05…$15.00.
DOI: http://dx.doi.org/10.1145/2944165.2944171

Industrial and business companies focus on implementing ERP systems to improve their daily activities and to achieve the integration between planning, HR, purchasing, manufacturing, stock and material control, sales orders, etc.

ERP systems must be adapted to support the business requirements. These requirements are changeable from one company to another, and are also subject to change with time in the same company according to many factors, such as changing in business processes or changing in company strategies or plans. Implementing an ERP system will increase the reflected impact on Return On Investment (ROI), in this case ROI depends on the ability to select, configure and maintain an ERP [1, 2].

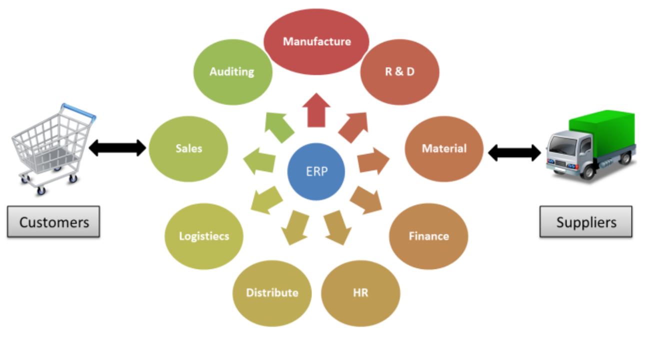

Figure 1. ERP system.

According to the review in [3], many researchers in IS research on ERP studied its implementation process. Based on these researchers the implementation process is considered as an important aspect in ERP researches. ERP implementation is the process that transforms a standard ERP product into an operational system in an organization. ERP implementation concentrates on two main issues: configuration and customization [4, 5, 6, and 7]. Configuration is about assigning values to a number of parameters recorded as data in an ERP [8]. Customization is the processes of extending ERP functionalities by adding new modules or modifying and adding code in an ERP software [9].

Software product Lines (SPLs) development refers to software engineering methods, tools and techniques for creating a collection of similar software systems from a shared set of software assets using a common means of production [10]. Carnegie Mellon Software Engineering Institute defines an SPL as [10]: "a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way." SPLs as a trend in software engineering is very promising as it can offer a lot of facilities and benefits for all types of stakeholders [10]. For example, software vendors can obtain the following benefits: reducing the production cost in the long term, short time to market, ability to provide mass production, more efficient use of human recourses and many other facilities. For software users, it provides well tested software which increases the overall quality of the software, provide highly customizable software and reduce the implementation cost. SPLs could be considered as a long time investment that aims to produce software with low cost in the long term [11]. SPLs consist of two main development phases: domain engineering phase and application engineering phase. Figure 2 illustrates the two phases of SPLs. In the domain engineering phase the assets and the components of an SPL are built to be reused later in the application engineering phase.

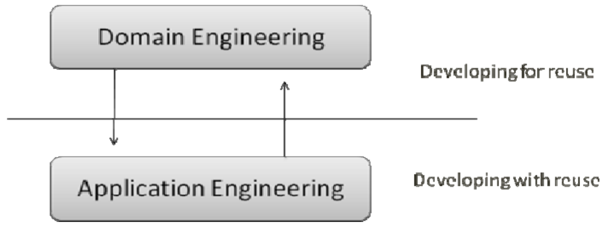

Figure 2. The Two Phases of SPLs.

The nature of ERP systems would directly fit for SPLs, due to the similar process of tailoring and customization among different ERP systems. ERP systems are widely used to support the daily and strategic activities of companies. They also help to integrate and connect all concerned parties inside and outside a company. Implementing ERP systems has gained attention in the last years, and the need for rapid implementation of ERP systems has risen as well. The implementation process of ERP systems among different companies will face similar adaptation and tailoring requests and that would lead to the combining with SPLs.

To the best of our knowledge, there is a lack in the literature in coverage of adopting and tailoring SPLs tools, techniques and approaches for specialized domains. In this research, we aim to introduce a requirements elicitation approach based on SPL for ERP systems which would be suitable for cloud solutions. It will rely on a requirements elicitation system, which is web based, and works to collect and manage requirements for an ERP system. Our approach emphasizes modeling requirements in earlier steps of requirements engineering. The base for modeling requirements are the existence of similar requirements from other developed applications and their modeling. It aims to represent the existing requirements and make them available to the stakeholders. The existing requirements are subject to change and modification according to each system's nature and requirements. The modified requirements would be available in a prototype form, to be validated by the stakeholders. Our requirements elicitation system will provide the ability to generate prototypes of the elicited requirements and then return them back to the stakeholders for validations. The gathered requirements will be stored in a requirements repository in order to be reused in other systems.

The rest of this paper is organized as follows. The research related work is introduced in section two. Section three introduces the challenges and limitations of applying SPLs to ERP systems in a cloud environment. The conceptual model of our approach is presented in section four. Finally, our conclusion and future work are given in section five.

2. RELATED WORK

A systematic literature review was conducted to identify the different ways to apply SPLs to ERP systems [12]. The review addressed the need for approaches, methods, techniques and tools for combining SPLs concepts with ERP systems. Only few researches proposed combining SPLs with ERP systems, e.g. [13, 14, 15, 16, 17, and 18]. Nobauer et al. [13] introduced mapping customization and configuration keys to the corresponding requirements; it also integrates artifacts with a variability model and stores artifacts in a feature model store. The introduced approach has been proposed to work with Microsoft Dynamics. Hamza et al. [14] introduced a practical experience from the application of Product Line Architectures (PLAs) in four companies of the ERP systems domain in Egypt. The research also addressed the challenges and difficulties that could face the implementation of SPLs for ERP systems. Dhungana et al. [15], introduced Invar approach, which integrates and unifies heterogeneous variability models of different actors (vendor, supplier) stored in repositories. The Invar approach concentrates on facilitating the integration of variability models during product configuration. Rabiser et al. [16] introduced a decision-oriented SPL approach to support customization at three levels: derivation by suppliers, configuration by customers and customization by end-users. Wolfinger et al. [17] integrated product line engineering and plug-in techniques to support ERP systems' adaptation. Leitner et al. [18] proposed a variant description model that comprises of all variants resolved and based on the variability defined in the feature model. It also introduced a mapping between the feature model and the family model, which contains ERP configuration options and documentation.

3. CHALLENGES AND LIMITATIONS

Applying SPLs to ERP systems is a very challenging process which faces many difficulties starting from gathering requirements or representing the existing requirements in a consistent form. This consistent form should be mapped onto a features model in order to manage the product line variability and its constraints. Distributed code ownership, documenting of variability, variability tool support and interdisciplinary work are the main challenges addressed in [13].

Owning the source code of the SPLs is a fundamental issue in building an SPL, which allows the software vendors to control and manage the changes of the code to fit different types of requirements for the same asset and to extend the assets functionality. The customization process also requires the existence of the source code. Modeling the variability is a key concept in SPLs [14]. There are many models supporting variability modeling in the context of SPLs; the most popular models are the orthogonal variability model [19] and the features model [20]. There are many tools to support the variability modeling, e,g, DOPLER [21] is a tool to represent variability in product lines. DOPLER is widely used in academic projects and the industry as well. An ERP project is highly connected; domain experts could discover the integration points and the interdependencies among ERP systems. The demand for mapping these interdependencies to SPLs is an important issue.

Differentiation between customization and configuration, system knowledge based on individuals, Unclear mapping between requirements and features, Development environment not suitable for automation and Complex reengineering process are other challenges addressed by [14]. Many software companies used to define features as a part of software which could be exist or not in certain product, in SPLs the feature is totally different. Software companies have problems with differentiates between customization and configuration in the context of SPL.

ERP systems are complex systems and the understanding of the ERP systems should rely on the company not on individual inside the company. Each company should define a real mapping between requirements and features in order to facilitate the reusing of requirements. Providing a systematic reuse strategy is a very expensive effort which requires extensive studying and planning. The existing parts of ERP systems should be re-engineered to the SPLs starting from requirements, this reengineering process is a complex process consumes time and cost.

4. APPROACH OVERVIEW

In this section we are going to discuss the approach in more detail. We divided our approach into two conceptual phases as represented in figure 3, section A represents the first phase while section B represents the second phase. The first phase intends to model requirements into the feature model. The requirements can be classified into two categories, existing requirements and new requirements. The existing requirements refer to the old requirements which collected at either the beginning of building the SPLs or requirements from previous developed applications. The other category of the requirements is the new requirements; the new requirements will be elicited and gathered through requirements elicitation system. The requirements elicitation system is a web based system works to represent the existing requirements from the requirements repository to the stakeholders. The stakeholders will take the decision of selecting among existing requirements or request new requirements. The requirements elicitation system will provide the ability of mapping the requirements onto the features model based one predefined roles.

The requirements elicitation system sends the new requirements to requirements engineers to review it and to maintain and add constraints between requirements.

After reviewing the new requirements, it will be mapped onto the features model; the requirements elicitation system will keeps track of all selected requirements for certain application. Based on these selected requirements and the corresponding features the system would generate requirements prototype.

The requirements prototype will represent the contents of the application under development and the structure of the application. After that the stakeholders will be involved in validating requirements and check for any missing requirements.

In the second phase of our conceptual approach the features configurations will be passed into the code generator. The code generator is a development tool that takes the features configurations as an input and builds the reusable assets based on these configurations. The generated assets will be stored in assets repository; all required assets will form the application under development. The development engineers will maintain the customization of the application assets if required. After the customization processes, these assets will be integrated and tested by test and integration engineers. After the testing and integration processes all required assets will be grouped together to form the final product or application. The final form of the application will be in the form of web based application to be capable to implement over cloud environment.

Our approach aims to be compatible with the implementation concept in ERP with some modification to cope with the SPLs. The implementation consists of two main steps configuration and customization. By configuration we mean selecting among different features to satisfy the stakeholders requirements. The selection is recorded and managed by the requirements elicitation system.

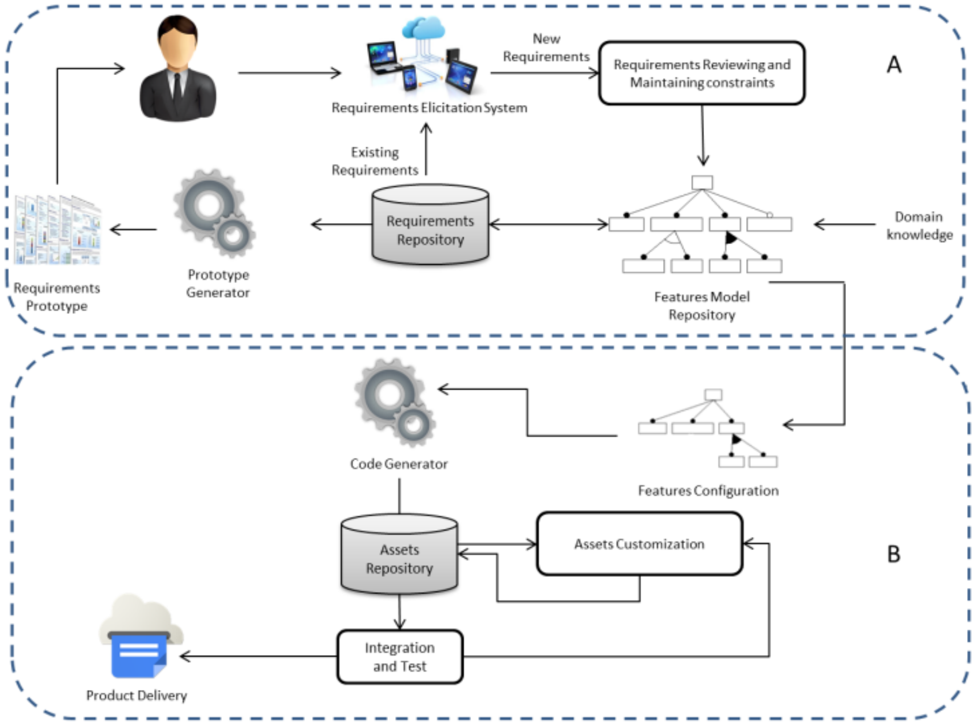

Figure 3. A simplified overview for the approach.

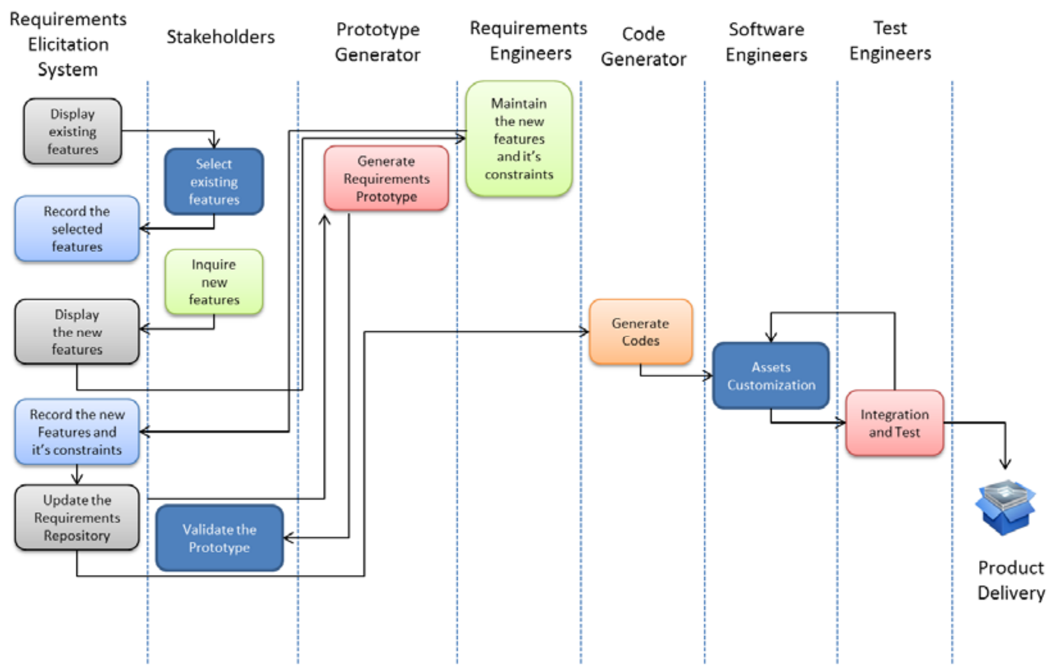

Figure 4. Roles of involved actors

Customization is the process of modifying the generated assets to fit extra requirements or to extend the functionality of the assets.

Our approach relies on two basic mechanisms, the requirement elicitation system and the code generation tool. The two mechanisms automate most of the reusing process of SPL.

As we mentioned before, the final product from our approach will be in the form of web application. After the implementation process and after the running of the application, the application will be connected to the requirements elicitation system. The involved stakeholders can maintain the increasing and the growth of the application functionality by requesting the new requirements from the requirements elicitation system. Stakeholders could be reducing the used assets also by requesting that from the requirements elicitation system. Our approach will work on keeps track the requirements changes after the implementation process starting from enquire new requirements to the delivery of the assets that fit these requirements.

The majority of cloud applications characteristics such as per use, anytime and anywhere accessibility, pay as you go, and scalability could be obtained by applying our approach. The ERP vendors will be responsible for hosting, running and considering the security and reliability issues of the application in addition to applying scalability requests.

Figure 4 illustrates the roles of each actor involved in our approach and the associated activities. The involved actors are stakeholders, requirements elicitation system, requirements engineers, development engineers, and test and integration engineers. Figure 5 illustrates the allocation of our approach into the traditional two phases of the SPLs, the domain engineering phase and the application engineering phase. Domain modeling, code generation and assets repository have been allocated in domain engineering phase.

The domain modeling step contains mapping the requirements into the features model, by requirements here we mean old or exist requirements and the new requirements. Application or product requirements engineering, assets customization and assets integration and testing have been allocated in application engineering phase. The application engineering phase contains the generation of requirements prototype and validating the requirements as well.

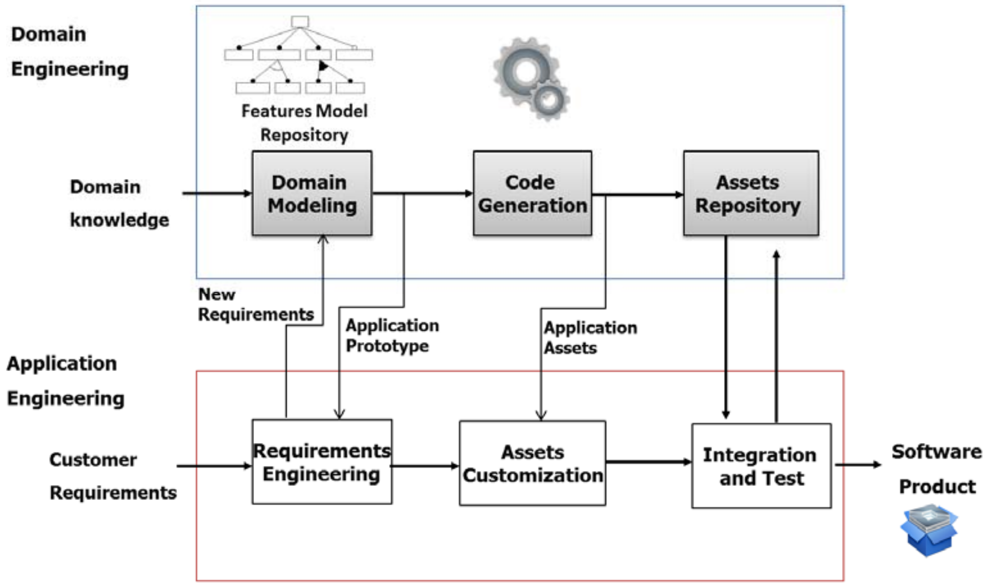

Figure 5. Our approach in traditional two phases of SPL.

Currently we are working to build the requirements elicitation system and integrate it with the code generator. The code generator is a tool used to build some parts of an ERP system, this ERP system has been implemented in the National Research Center (NRC). NRC is an Egyptian research and development center located in Cairo, it was established in 1956 to foster basic and applied scientific research. NRC is the largest institution affiliated with the ministry of scientific research. It has a research staff of more than 4000 scientists.

5. CONCLUSION AND FUTURE WORK

In this paper we introduced an overview of our approach; the approach intends to display the existing requirements of the SPLs to the stakeholders, in order to facilitate the requirements elicitation. The displayed requirements are subject to change and modify based on the stakeholder needs. We introduced the basic idea of the requirements elicitation systems which is responsible of mapping requirements to feature model, manage requirements changes, generating requirements prototype and apply validated requirements. For each application, the approach handles the enquiry of the new requirements and maintains the allocation of the new requirements into the feature model. Our approach relies on a code generator which translates the modeled requirements into reusable assets. The generated assets can be customized and integrated with different assets. The delivered application or the final product will be in the form of web based application to be able to run and operate over the cloud environment and to take the advantages of scalability options and other cloud applications options.

We have combined different techniques and practices from different trends in SPLs and ERP to introduce a requirements elicitation approach. This approach is designed to work suitably in the cloud environment. We consider this combination is the main contribution of our work as we didn't discover any similar research in this area. Most of published researches have considered the applying of SPLs for ERP systems not for cloud ERP.

We are currently working to build the requirements elicitation systems, the system will introduce a clear mapping from elicited requirements into the features model. We plan to use an adapted form of the features model to handle the process of mapping requirements. This form will include an attributes for each feature and we will extend the features model constraints to support the mapping process. The prototype generator is a part of the requirements elicitation systems.

We will also work to integrate the requirement elicitation system with the code generator system. Our future work will introduce comprehensive description of both systems the requirement elicitation system and the code generation system.

6. REFERENCES

[1] Newman, Mike; Zhao, Yu. The process of enterprise resource planning implementation and business process re-engineering: tales from two Chinese small and medium-sized enterprises. *Information Systems Journal*, 18, 4 (2008), 405–426.

[2] Chang, Hsin Hsin. Technical and management perceptions of enterprise information system importance, implementation and benefits. *Information Systems Journal*, 16, 3 (2006), 263-292.

[3] Pekkola, Samuli; Niemi, Erkka; Rossi, Matti; Ruskamo, Miika; Salmimaa, Taru. ERP Research At ECIS And ICIS: A Fashion Wave Calming Down? In *Proceedings of the European Conference on IS (ECIS)* (Utrecht, NL 2013).

[4] Al-Mudimigh, A; Zairi M; Al-Mashari M. ERP software implementation: an integrative framework. *Eur J Inf Syst*, 10, 4 (2001), 216-226.

[5] Shanksa, Graeme. A model of ERP project implementation. *Journal of Information Technology*, 15, 4 (2000), 289-303.

[6] Claremont, Lynne Markus; Tanis, Cornelis; Van Fenema, Paul C. Multisite ERP implementations. *Communications of*

the ACM, 43, 4 (2000), 42–46.

[7] Davis, Ashley. ERP Customization Impacts on Strategic Alignment and Systems Alignment. In *SAIS Proceedings of the Southern Association of IS (SAIS)* (2005), 45.

[8] Rosemann, Michael. Requirements Engineering for Enterprise Systems. In *in Proceedings of the American Conf. on IS (AMCIS)* (2001).

[9] M. , Daneva. Lessons learnt from five years of experiencein ERP requirements engineering. In *Proceedings 11thIEEE Int. Req. Eng. Conference (RE'2003)* (2003), 45-54.

[10] *Software Product Lines*. Carnegie Mellon Software Engineering Institute: http://www.sei.cmu.edu/productlines/.

[11] Clements , Paul C; Mcgregor , John D; Cohen , Sholom G. *The Structured Intuitive Model for Product Line Economics (SIMPLE)*. Software Engineering Institute, Carnegie Mellon University., 2005.

[12] Mazo, Raúl; Assar, Saïd; Salinesi, Camille; Ben Hassen, Noura. Using Software Product Line to improve ERP Engineering: Literature Review and Analysis. *Latin-American Journal of Computing (LAJC)*, 1, 1 (2014).

[13] Nöbauer, Markus; Seyff, Norbert; Dhungana, Deepak; Stoiber, Reinhard. Managing variability of ERP ecosystems: research issues and solution ideas from Microsoft Dynamics AX. In *Proceedings of the 6th Int'l Workshop on Variability Modeling of Software-Intensive Systems (VaMoS '12)* (Leipzig, Germany 2008), 21-26.

[14] Hamza , Haitham S; Martinez, Jabier ; Alonso , Carmen. Introducing Product Line Architectures in the ERP Industry: Challenges and Lessons Learned. In *SPLC Conference* (Jeju Island, South Korea 2010), 263-266.

[15] Dhungana, Deepak; Seichter, Dominik; Botterweck, Goetz; Galindo, José. Configuration of Multi Product Lines by Bridging Heterogeneous Variability Modeling Approaches. In *SPLC conference* (Munich, Germany 2011), 120 – 129.

[16] Rabiser, Rick; Grünbacher, P. Threelevel Customization of Software Products Using a Product Line Approach. In *the 42nd Hawaii Int'l Conf. on System Sciences (HICSS-42)* (Waikoloa, Hawaii, USA 2009), IEEE Computer Society.

[17] Wolfinger, Reinhard; Reiter, Stephan; Dhungana, Deepak; Grunbacher, Paul; Prahofer, Herbert. Supporting Runtime System Adaptation through Product Line Engineering and Plug-in Techniques. In *The 7th IEEE Int. Conf. on Composition-Based Software Systems (ICCBSS)* (Madrid, Spain, 2008), 21–30.

[18] Leitner, Andrea; Kreiner, Christian. Managing ERP Configuration Variants: An Experience Report. In *the 2010 Workshop on Knowledge-Oriented Product Line Engineering, (KOPLE'10)* (2010).

[19] Pohl , Klaus ; Böckle, Günter; Linden, Frank Van Der. *Software Product Line Engineering*. Springer, 2005.

[20] Kang, Kyo C.; Cohen, Sholom G; Hess, James A; Novak, William E; peterson , A. Spencer. *Feature-oriented domain analysis (FODA) feasibility study*. Technical Report CMU/ SEI-90TR-021, SEI, Carnegie Mellon University, 1990.

[21] Dhungana, Deepak ; Grünbacher, Paul; Rabiser, Rick. The DOPLER metatool for decision-oriented variability modeling: A multiple case study. *Automated Software Engineering*, 18, 1 (2011), 77–114.