Journal of Engineering and Applied Sciences 13 (21): 9141-9148, 2018
ISSN: 1816-949X

Towards a Configurable Application Service Platform for Micro e-Services Providers

[1, 2]Azubuike Ezenwoke and [3]Mathew Adigun
[1]Covenant University, Ota, Nigeria
[2]Landmark University, Omu-Aran, Nigeria
[3]Center for Excellence for Mobile e-Services, University of Zululand, Richards Bay, South Africa

Abstract: Application service platforms, also called application Platform-as a Service (aPaaS), comprise a set of software and hardware components (databases, servers, network resources, integrated development environment) that provide mechanisms for the development, deployment and management of end-user application services in multiple business domains. The aim of this study is to propose a design of a configurable aPaaS in a manner that is interactive and cost-effective, while optimal configurations can be obtained from the aPaaS based on the complex multi-dimensional variability techniques used in software product-line engineering. Our proposed approach integrates interactive preference articulation method for multi objective optimization during platform configuration of a product-line-as-a-service platform. The proposed PLaaSP is a viable framework that enables software product-line engineering technique to be consumed on-demand while providing micro e-services providers to adopt software product-line competencies, thereby reducing the cost and steep learning curve of adopting software product-line practice.

Key words: PaaS, application platform as a service, software product line, cloud computing, adopt software, frame work and technique

INTRODUCTION

Application service platforms, also called application Platform-as-a-Service (aPaaS) is a Platform as a service offering, comprise a set of software and hardware components (databases, servers, network resources, integrated development environment) that provide mechanisms for the development, deployment and management of end-user application services in multiple business domains (e.g., business, health, education) (Anonymous, 2017). The aPaaS serves multiple users with different functional and non-functional requirements via. a multi-tenancy arrangement. It can also, integrate many services from diverse third party and external collaborators to actualize the platform's business objective. Some advantages of adopting aPaaS offerings over the traditional approach of the application service development include rapid speed in application development and deployment by overcoming the time delay in coding applications. Examples of existing aPaaS include mendix.com, salesforce.com and xait.com.

The users of the aPaaS, for example, e-Services providers have different views of the platform based on each their requirements and business objectives while the functionalities of the platform, like database, storages, etc. can be described as a product-line of enabling services. Therefore, the aPaaS could be described in terms of features, expressed using feature models. Feature models (Kang, 1990) are the most widely used representations for variabilities and commonalities in Software Product-Lines (SPL) (Benavides *et al*., 2010). The products of a product-line vary in terms of features, therefore an important characteristic of SPL is its support for variability (Filho *et al*., 2012; Svahnberg *et al*., 2005). Variability refers to the ability of an artefact to be configured, customized, extended or changed for use in a specific context. In the aPaaS, users configure the features of the platform, to derive an instance to perform user-specific usiness tasks. Furthermore, these features (and their attributes) of the aPaaS could be very large and an attempt to manually derive meaningful compositions from these models is time-consuming and error-prone (Deelstra *et al*., 2005; Rabiser *et al*., 2009; White *et al*., 2008).

This challenge can be overcome by employing the automated approaches to extract useful information from feature models, during product configuration (Benavides *et al*., 2010; Elfaki *et al*., 2012; Karatas *et al*., 2013). Several studies on automated analysis on feature models have been conducted and reported in the literature (Benavides *et al*., 2010). The search for valid configurations from the feature models ranges from just any configuration with no particular preference to one or more optimal configurations that satisfy specific objective functions based on specific criteria (Trinidad *et al*.,

Corresponding Author: Azubuike Ezenwoke, Covenant University, Ota, Nigeria

2008). Searching for an optimal feature set involves the use of some preference constraints to optimize search results. The main problem addressed in this study optimize the inclusion of features in the configuration of the aPaaS in the face of multiple constraints.

Existing optimal configuration solutions in the literature have used either a priori or a posteriori approaches which do not allow iterative refinement of users preferences in a flexible way that engenders the generation of optimal configurations. Hence, an efficient approach that will facilitate engender the derivation of the optimal feature set that accurately approximates each users requirements and preferences at minimal operational cost and in a flexible way is required. The aim of this study is to propose a design of a configurable aPaaS in a manner that is interactive and cost-effective while optimal configurations can be obtained from the aPaaS based on the complex multi-dimensional variability techniques used in software product-line engineering. Our proposed approach integrates interactive preference articulation method for multi-objective optimization during platform configuration.

Background

Variability management techniques: As with aPaaS, software product-line engineering practice leads to significant reduction in development efforts, costs and time-to-market compared to single developed systems. Modeling variability is an important aspect of a software product-line engineering endeavor because it is used to understand, define and manage the commonalities and the variabilities (Czarnecki *et al*., 2012). Variability modelling research has received a lot of attention in the research community as there are several approaches for modelling and managing variability reported in the literature (Czarnecki *et al*., 2012). These approaches can be categorized into Feature-based and Decision-based Modeling, (Czarnecki *et al*., 2012). A third category consists of other approaches that handle more complex variability information. Rosenmuller *et al*. (2011) introduced a multi-dimensional modelling approach to represent with separate variability models the different variability dimensions of the product line. Hartmann and Trew (2008) proposed the concept of context variability model as a key determinant for variation. The context variability model, together with feature model enables multiple product-lines modelling that supports multiple dimensions in the design space. Extending this concept by adding a branch that captures hardware requirements makes it suitable in for this study. However, a feature-based modelling approach was adopted in this study because of the capability to perform automated reasoning.

Variability modeling languages: Variability Modelling Languages (VML) are either graphical or textual. Textual VMLs are preferred in modelling realistic product families, to graphical notations due to lack of concision, naturalness and expressiveness (Classen *et al*., 2011). Classen et al. identified popular examples of graphical variability notations FODA (Kang, 1990), FeatuRSEB (Griss *et al*., 1998) from (Kang *et al*., 1998) and Generative Programming (Czarnecki *et al*., 2000). Textual variability modelling languages include Velvet (Rosenmuller *et al*., 2011; Clafer Bak *et al*., 2010 and its extension, ClaferMoo Olaechea *et al*., 2012. Velvet is a language for multi-dimensional variability that could be used for both feature modelling and configuration, allowing stakeholders to model different dimensions of the product-line separately and later combined when required. Velvet uses part of the syntax from TVL's and C#. Clafer (CLass, FEature, Reference) is minimalistic modelling language that provides a concise notation for feature models and Meta-Model with formal semantics to specify constraints. Clafer with Alloy, supports automated reasoning. CafeMom extends Clafer with facilities to express multiple optimization objectives.

However, in the aPaaS contex, it is a non-trivial task to define complex dependencies on all the layers of the platform, i.e., the tenant-specific instantiations and underlying provisioning platform. Motivated by the concept proposed by Rosenmuller *et al*. (2011) our approach will provide an explicit means (language) to express such complex variability dependencies. The proposed language would satisfy the criteria defined by Classen, Boucher and Heymans (Classen *et al*., 2011) and at the same time support multiple optimization goals during configuration.

Automated analysis of feature model: Automated analysis of feature models uses computer-aided mechanisms to extract important information from feature models (Benavides *et al*., 2010). The automated approach entails mapping the feature models into a specific formal representation as inputs and solvers are used to perform analysis operations to obtain results. Formal configuration techniques used to provide automated support for analysis operations have been classified into four categories: Propositional Logic (PL), Description Logic (DL), Constraint Programming (CP) and Ad-hoc algorithms (Benavides *et al*., 2007). The PL approaches translate feature models into a propositional formula and satisfiability solvers are used analyze the formula. In DL

approaches, feature models are mapped into description logic and logic-based reasoners are used for the analysis. The CP approach represents feature models as a Constraint Satisfaction Problem (CSP) and CSP solvers use constraint programming to find the solution to the problem.

Multi-objective optimization: Extended feature models are desirable variability representations of aPaaS features because it captures features their attributes and inter-feature cross-tree relationships and the effect of these on product configuration. Even though automated analysis on of basic and cardinality-based feature models have been thoroughly researched and reported, extended feature models have not been widely covered (Benavides *et al*., 2010). Consequently, existing languages and tool support for modelling extended feature models are limited (Classen *et al*., 2011; Olaechea *et al*., 2012). A suitable variability modelling language that can capture the variability complexity towards automated optimal configuration of products to achieve the architectural vision of the aPaaS must be integrated into the platform design from the beginning. The formal semantics of such language must be expressive enough to capture the interactions among the various models and support automated optimized configuration of products. However, many existing languages and tools do not totally support multi-objective optimization (Olaechea *et al*., 2012). Some works attempted to optimize multiple non-functional or quality attributes in deriving best solutions, (Olaechea *et al*., 2012; Siegmund *et al*., 2012; Soltani *et al*., 2012).

Literature review: In extended feature models, variability models are annotated with quality information (non-functional requirements, e.g., memory consumption, cost etc.), the analysis could use qualities as a basis in specifying preferred configuration requirements. Roos-Frantz *et al*. (2012) developed an approach for quality-aware analysis in software product-lines based on the Orthogonal Variability Model (OVM). Quality-centric variability information was translated into a CSP and a prototype tool (FaMa-OVM) was used to perform verification task that meets certain quality conditions. Karatas *et al*. (2013) introduced a way to map extended feature models to constraint logic programming over finite domains. This approach enabled the use of CLP (FD) solvers (a class of CSP solvers) to analysis the models with complex cross-tree inter-attribute relationships. By Soltani *et al*. (2012) used Hierarchical Task Network (HTN), a preference-based artificial intelligence planning technique called to represent feature model's functional and non-functional requirements. In the approach, feature models and stakeholder's preferences (based on non-functional qualities) are mapped into HTN planning problem and SHOP2 is used to derive an optimal plan.

Stakeholders input are vital during the configuration process not just before the configuration process, in order to obtain the most optimal result that satisfies the stakeholder's requirements. Interactive and batch configuration was mentioned by Mendonca *et al*. (2009). However, none of these approaches considers iterative inputs during configuration in order to generate the most optimal results. By Olaechea *et al*. (2012) Siegmund *et al*. (2012) and Soltani *et al*. (2012) attempts to optimize multiple non-functional or quality attributes in deriving optimal solutions were presented, however, none of these proposals, except Soltani *et al*. (2012) considered integrating users preference information in the configuration process. Yet, the research presented in captured user's preference a priori, i.e., before the search for optimal configuration.

The approach presented in this study proposes a preference-based multi-objective optimization algorithm that allows for several intermediate inputs by the user, during the runs of the algorithm, to obtain the most preferred optimal configuration in a computationally efficient manner. The three ways to specify preference information in interactive methods are trade-off information, reference points and classification of objective functions (Miettinen, 2008). We hypothesize that an interactive approach is most suitable for achieving the goals of multi-objective optimization configuration of the aPaaS.

MATERIALS AND METHODS

Configurable product-line as a service platform: In this study the envisioned multi-dimension application service platform is a Product-line as a Service Platform (PLaaSP) (Fig. 1). Based on the underlying principles of PaaS and aPaaS, PLaaSP is conceptualized as a web-based on-demand integrated development environment for the design, development, deployment and management of a product-line of service-based applications. A PLaaSP offering would comprise of databases, middleware, product-line development tools, run-time and execution environments and would host and serve multiple tenants per time with each tenant having a private view of the platform.

The motivation for the pursuit of a PLaaSP initiative is to create a viable platform for micro e-Services firms,

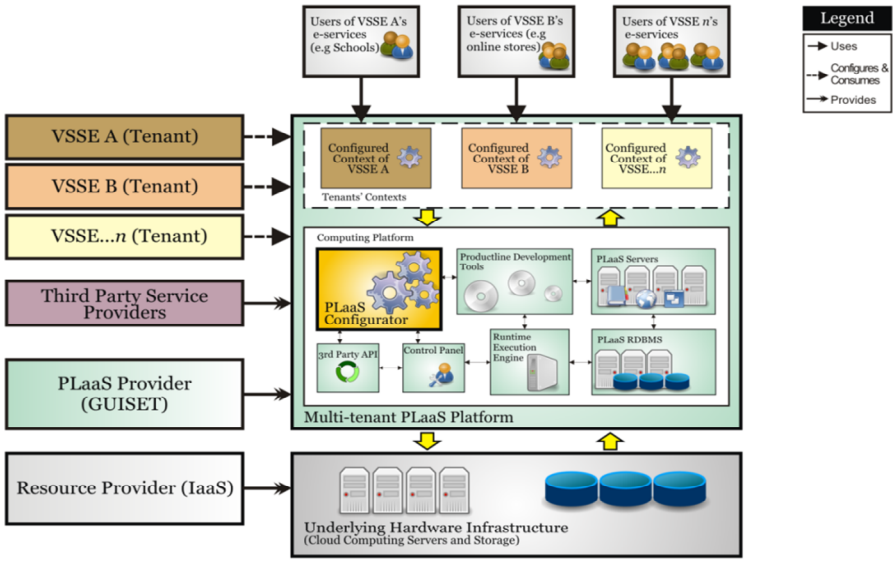

Fig. 1: A view of PLaaSP platform and stakeholders

with fewer than 10 IT employees to readily access on-demand core software product-line services such as variability modelling and custom web application configuration without incurring the huge cost associated with adopting an independent SPL initiative. The use of an interactive preference-based optimization approach for configuration of the PLaaSP will facilitate the generation high-quality PLaaSP configurations that best meet each firm's requirements and preferences in an effective manner.

Plaasp design considerations

Multi-dimensional variability of PLaaSP environment: PLaaSP is conceived as a multi-tenancy arrangement (Krebs *et al*., 2012) that enables the provision of dynamically customizable development environments to satisfy multiple and diverse tenant's requirements in a cost-effective manner. In the literature, the use of variability management techniques from the software product-line domain has been proposed for the realization of multi-tenancy (Mietzner *et al*., 2009; Ruehl and Andelfinger 2011) and hence, software product-line engineering as an engineering approach is suitable to engineer a multi-tenant PLaaSP platform. In a PLaaSP platform, variability could be expressed in various dimensions (Rosenmuller *et al*., 2011; Schroeter *et al*., 2012) by separating the feature set such as the hardware features, external software features, internal (Rosenmuller *et al*., 2011) and software context (Hartmann and Trew, 2008).

Automated analysis of feature models: The multiple variability dimensions of a PLaaSP separated into several feature models, enables on-demand composition and analysis (Rosenmuller *et al*., 2011; Schroeter *et al*., 2012). The language used to model such multi-dimensions must have robust formal semantics that facilitates automated analysis and configuration (Classen *et al*., 2011). Several tools and solvers for automating the analysis of feature models have been proposed in the literature (Benavides *et al*., 2010). Some of these tools use specific solvers based on anyone of CSP-based, propositional logic-based (BDD and SAT) or ad hoc algorithms-based logic representations to automated reasoning on feature models. Benavides *et al*. (2007) believes that performance could be enhanced if multiple solvers are used by integrating them into one tool and hence proposed FAMA (Feature Model Analyser), a framework that integrates CSP, BDD and SAT logic representation to optimize the analysis process. Also Mendonca *et al*. (2009) proposed SPLOT (Rabiser *et al*., 2009) as a web-based system and configuration system that uses an HTML-based template engine to create interactive Ajax-based reasoning based on BDD and SAT solvers. The online usage context and support for interactive configuration of our solution is similar to SPLOT. Even though SPLOT is not used within a service delivery context like the MASP, our approach would be a web-based tool that can be integrated to the PLaaSP platform where prospective IT firm could 'shop' for preferred 'slice' of the platform in an interactive manner.

Multi-objective optimization in PLaaSP environment: Multi-tenant cloud-based environment such as the

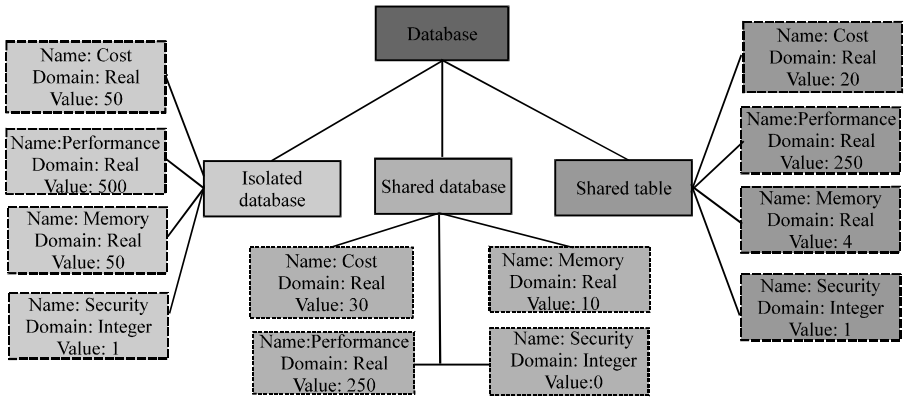

Fig. 2: A view of the database feature from the feature model for an e-Shop product-line (the dotted box denotes the attribute of each of the sub-features)

PLaaSP, requires computationally efficient product configurator (in terms of CPU and memory consumption) for automated configuration of multi-dimensional feature models. Importantly, the platform should comprise a self-optimizing maximizing the business value derived from consuming PLaaSP and the optimal utilization of cloud infrastructure and resources. From the PLaaSP provider's viewpoint an efficient way to multiplex tenants activities and demands to reduce operational cost are desirable (Schroeter *et al*., 2012). The process of selecting the most optimal set of features that satisfy multiple optimization goals during PLaaSP configuration in order to consume PLaaSP is an important design consideration.

The PLaaSP user provides preference information to determine the best available combination of features that approximate the tenant's requirements (functional and non-functional). The PLaaSP platform configurator should comprise multi-objective optimization mechanisms that incorporate the user's preference information in the decision making process. This would be done in such a manner that optimizes the utilization of the cloud infrastructure and resources in satisfying the user requirements and preferences. This means that techniques for configuring the PLaaSP variability should support preference-based multi-objective optimization and should derive optimal (valid and satisfactory) configurations in reasonable time; A valid and satisfactory configuration in the sense that the configuration is correct complete and approximates the tenant's requirements and preferences. The inputs to such optimization operation are the feature models and the objective function while the output is a product configuration that satisfies the criteria defined by the function (Benavides *et al*., 2010). The optimization operation, most suitable on extended feature models is such that a set of features can be selected by maximizing or minimizing the values of given feature attributes (Karatas *et al*., 2013). For example, consider a database feature from the feature model of an eShop product-line (Fig. 2). The decision to select a database facility could be evaluated in terms of its cost, performance, security and memory and the cost is implicitly a function of performance, security and memory. The cheapest database facility to manage the data of an e-Shop instance is shared-tables which costs 20 compared to shared-database or isolated database. A Multi-Objective Optimization Problem (MOP) scenario is described when a user in selecting a database facility for its e-Shop services, desires to maximize performance, security and memory while at the same time minimize cost. Multi-objective optimization is an optimization that involves two or more conflicting objective functions that must be simultaneously optimized in the face of a given set of constraints.

Incorporating user's preference information: The three main methods of incorporating preference information for solving multi-objective optimization problem are a priori, methods (Miettinen, 2008). In a priori methods, the multiple objectives are combined into a single objective function through a process called scalarization. requirements and preference information and the platform's configurator in a search process, attempts to find a combination of feature that approximates as much as possible, the requirements and preference information. The drawback of this approach is that the user may cut off the possibilities of arriving at 'better' feature combinations, due to the constraints imposed by the user's preference, defined a priori. Therefore, the opportunity cost of missing out on a more approximate feature configuration that satisfies user's requirements and preference is very high. In a posteriori methods, a set of Pareto optimal configurations is first generated and the user is expected to select the most preferred. In spite of the user being exposed to an overview of available Pareto optimal feature combinations, the search process of a

posteriori methods could be increasingly complex and computationally expensive. Furthermore, if the search process is terminated too early the feature combinations presented may not be the most optimal.

With interactive methods, the interaction between the user and the platform's configurator increases the possibilities of arriving at more desirable feature configurations and generates better search results. The decision-making process of the user, as well as the optimization activities of the platform's configurator are interlaced together such that the preferences specified by the user at a given instance are what determines the optimal configuration generated by the platform. The partial results of the search are revised again by the users, and the search process progresses.This process continues until the final solution is reached. Furthermore, it is possible that the results of a multi-objective optimization operation may not be desirable at the first iteration. For example, a user of the database service (Fig. 2) as part of a larger composite e-Shop service, may desire to increase his budget by a certain amount in other to get additional value for database performance and security.

Therefore, the iterative refinement of criteria by re-specifying preference information should be allowed until desired feature combinations are obtained. Similarly, a user may desire to relax or tighten its preferences during the configuration search process based on current partial search results. In making such decisions during configuration, the configurator should automatically propagate the effect of such decisions to ensure their consistency with previously made features selections. So far, configuration solutions proposed in the literature, to derive optimal configurations, have mainly used either a priori or a posteriori methods, which lack the kind of flexibility afforded by interactive methods. Hence, the proposed platform supports the derivation of optimal configurations by allowing iterative refinement of preference information during the platform configuration process in a manner that accurately approximate user satisfaction at minimal computational cost.

RESULTS AND DISCUSSION

This study is carried out in the context of the GUISET project (Ezenwoke *et al*., 2013). GUISET is envisioned as both an enabling infrastructure and a suite of on-demand service-based application. As a cloud-computing model, GUISET is aimed at offering affordable e-Enabling and "appliance-like" technology services (e.g., PLaaSP) through the internet to lower the total cost of ownership. The GUISET infrastructure would provide developers with the required tools and environments for consuming PLaaSP on a pay-as-you-go basis. These services are aimed at e-Enabling the activities of under-resourced IT firms. The proposed PLaaSP embodies the following a viable framework that enables software product-line reducing the cost and steep learning curve of adopting software product-line practice, reduces the development efforts and cost of quality software projects and consequently application services by enabling micro e-Services firms to concentrate on their core competence and th engineering technique to be consumed as a service over the internet on a pay-as-you-go payment basis, thereby enabling the development of quality software products, a means to for micro e-Services providers to adopt software product-line competencies in an inexpensive manner, thereby e-Business side of their operations.

CONCLUSION

In this study, we provide an architectural vision of a configurable application platform as a service called the product-line as a service platform. The goal of the platform is to e-Enable micro e-Services firms to adopt software product-line services in a manner that is inexpensive and ability to configure the platform to suit their business objectives. We proposed the use of an automated configuration approach that incorporates interactive preference articulation methods for optimizing multiple and conflicting user's objectives towards the derivation of optimal configurations from the PLaaSP. The derivation of optimal configurations in a time efficient manner optimizes consumption of computational resources (and minimize operational cost) while satisfying the user's requirements in an interactive manner. The providers of the PLaaSP benefits from such techniques to minimize the operational cost for CPU, Memory and bandwidth required to attract and retain potential tenants.

SUGGESTIONS

As a future research, full implementation of the platform would be achieved and experiments would be performed to ascertain the performance of approach proposed the in the same environment.

REFERENCES

Anonymous, 2017. Application platform as a service. Gartner, Stamford, Connecticut, USA. https://www.gartner.com/it-glossary/application-platform-as-a-service-apaas.

Bak, K., K. Czarnecki and A. Wasowski, 2010. Feature and meta-models in clafer: Mixed, Specialized and coupled. Proceedings of the International SLE Conference on Software Language Engineering, September 15-16, 2010, Springer, Berlin, Germany, ISBN:978-3-642-19439-9, pp: 102-122.

Benavides, D., S. Segura and A. Ruiz-Cortes, 2010. Automated analysis of feature models 20 years later: A literature review. Inform. Syst., 35: 615-636.

Benavides, D., S. Segura, P. Trinidad and A. Ruiz-Cortes, 2007. FAMA: Tooling a framework for the automated analysis of feature models. Proceedings of the 1st International Workshop on Variability Modelling of Software Intensive Systems, January 16-18, 2007, Limerick, Ireland.

Classen, A., Q. Boucher and P. Heymans, 2011. A text-based approach to feature modelling: Syntax and semantics of TVL. Sci. Comput. Program., 76: 1130-1143.

Czarnecki, K. and U.W. Eisenecker, 2000. Generative Programming: Methods, Tools and Applications. Addison-Wesley, USA., ISBN-13: 9780201309775, Pages: 832.

Czarnecki, K., P. Grunbacher, R. Rabiser, K. Schmid and A. Wasowski, 2012. Cool features and tough decisions: A comparison of variability modeling approaches. Proceedings of the 6th International Workshop on Variability Modeling of Software-Intensive Systems (VoMoS'12), January 25-27, 2012, ACM, New York, USA., ISBN: 978-1-4503-1058-1, pp: 173-182.

Deelstra, S., M. Sinnema and J. Bosch, 2005. Product derivation in software product families: A case study. J. Syst. Software, 74: 173-194.

Elfaki, A.O., O.A. Abouabdalla, S.L. Fong, G.M. Johar, K.L.T. Aik and R. Bachok, 2012. Review and future directions of the automated validation in software product line engineering. J. Theoret. Applied Inform. Technol., 42: 75-93.

Ezenwoke, A., S. Misra and M. Adigun, 2013. An approach for e-Commerce on-demand service-oriented product line development. Acta Polytech. Hungarica, 10: 69-87.

Filho, F.J.B., O. Barais, B. Baudry and L.J. Noir, 2012. Leveraging variability modeling for multi-dimensional model-driven software product lines. Proceedings of the 2012 3rd International Workshop on Product LinE Approaches in Software Engineering (PLEASE), June 4, 2012, IEEE, Zurich, Switzerland, ISBN: 978-1-4673-1751-1, pp: 5-8.

Griss, M.L., J. Favaro and M. d'Alessandro, 1998. Integrating feature modeling with the RSEB. Proceedings of the 5th International Conference on Software Reuse, June 5, 1998, IEEE, Victoria, BC., Canada, pp: 76-85.

Hartmann, H. and T. Trew, 2008. Using feature diagrams with context variability to model multiple product lines for software supply chains. Proceedings of the 12th International Conference on Software Product Line SPLC'08, September 8-12, 2008, IEEE, Limerick, Ireland, ISBN: 978-0-7695-3303-2, pp: 12-21.

Kang, K.C., 1990. Feature-Oriented Domain Analysis (FODA) Feasibility Study. Software Engineering Institute, Carnegie Mellon University, Pittsburgh.

Kang, K.C., S. Kim, J. Lee, K. Kim and E. Shin *et al*., 1998. FORM: A feature: Oriented reuse method with domain: Specific reference architectures. Ann. Software Eng., 5: 143-168.

Karatas, A.S., H. Oguztuzun and A. Dogru, 2013. From extended feature models to constraint logic programming. Sci. Comput. Programming, 78: 2295-2312.

Krebs, R., C. Momm and S. Kounev, 2012. Architectural concerns in multi-tenant SaaS applications. Proceedings of the 2nd International Conference on Cloud Computing and Services Science (CLOSER 2012) Vol. 12, April 18-21, 2012, At Setubal, Portugal, pp: 426-431.

Mendonca, M., M. Branco and D. Cowan, 2009. SPLOT: Software product lines online tools. Proceedings of the 24th ACM SIGPLAN International Conference on Object Oriented Programming Systems Languages and Applications Companion, October 25-29, 2009, ACM, Orlando, Florida, USA., ISBN:978-1-60558-768-4, pp: 761-762.

Miettinen, K., 2008. Introduction to Multiobjective Optimization: Noninteractive Approaches. In: Multiobjective Optimization, Branke, J., K. Deb, K. Miettinen and R. Slowinski (Eds.). Springer, Berlin, Germany, ISBN:978-3-540-88907-6, pp: 1-26.

Mietzner, R., A. Metzger, F. Leymann and K. Pohl, 2009. Variability modeling to support customization and deployment of multi-tenant-aware software as a service applications. Proceedings of the 2009 ICSE Workshop on Principles of Engineering Service Oriented Systems, May 18-19, 2009, IEEE Computer Society, Washington, DC, USA., ISBN:978-1-4244-3716-0, pp: 18-25.

Olaechea, R., S. Stewart, K. Czarnecki and D. Rayside, 2012. Modelling and multi-objective optimization of quality attributes in variability-rich software. Proceedings of the 4th International Workshop on Nonfunctional System Properties in Domain Specific Modeling Languages, October 1-5, 2012, ACM, New York, USA., ISBN:978-1-4503-1807-5, pp: 2-6.

Rabiser, R., R. Wolfinger and P. Grunbacher, 2009. Three-level customization of software products using a product line approach. Proceedings of the 42nd Hawaii International Conference on System Sciences HICSS'09, January 5-8, 2009, IEEE, Big Island, Hawaii, USA., ISBN:978-0-7695-3450-3, pp: 1-10.

Roos-Frantz, F., D. Benavides, A. Ruiz-Cortes, A. Heuer and K. Lauenroth, 2012. Quality-aware analysis in product line engineering with the orthogonal variability model. Software Qual. J., 20: 519-565.

Rosenmuller, M., N. Siegmund, T. Thum and G. Saake, 2011. Multi-dimensional variability modeling. Proceedings of the 5th International Workshop on Variability Modeling of Software-Intensive Systems, January 27-29, 2011, ACM, New York, USA., ISBN:978-1-4503-0570-9, pp: 11-20.

Ruehl, S.T. and U. Andelfinger, 2011. Applying software product lines to create customizable software-as-a-service applications two kinds of flexibility. Appl. Sci., 2011: 1-4.

Schroeter, J., S. Cech, S. Gotz, C. Wilke and U. ABmann, 2012. Towards modeling a variable architecture for multi-tenant SaaS-applications. Proceedings of the 6th International Workshop on Variability Modeling of Software-Intensive Systems, January 25-27, 2012, ACM, New York, USA., ISBN:978-1-4503-1058-1, pp: 111-120.

Siegmund, N., M. Rosenmuller, M. Kuhlemann, C. Kastner and S. Apel *et al*., 2012. SPL conqueror: Toward optimization of non-functional properties in software product lines. Software Qual. J., 20: 487-517.

Soltani, S., M. Asadi, D. Gasevic, M. Hatala and E. Bagheri, 2012. Automated planning for feature model configuration based on functional and non-functional requirements. Proceedings of the 16th International Conference on Software Product Line Vol. 1, September 2-7, 2012, ACM, New York, USA., ISBN:978-1-4503-1094-9, pp: 56-65.

Svahnberg, M., V.J. Gurp and J. Bosch, 2005. A taxonomy of variability realization techniques. Software Pract. Experience, 35: 705-754.

Trinidad, P., D. Benavides, A. Duran, A. Ruiz-Cortes and M. Toro, 2008. Automated error analysis for the agilization of feature modeling. J. Syst. Software, 81: 883-896.

White, J., D.C. Schmidt, D. Benavides, P. Trinidad and A. Ruiz-Cortes, 2008. Automated diagnosis of product-line configuration errors in feature models. Proceedings of the 12th International Conference on Software Product Line SPLC'08, September 8-12, 2008, IEEE, Limerick, Ireland, ISBN:978-0-7695-3303-2, pp: 225-234.