An Approach for e-Commerce On-Demand Service-oriented Product line Development

Azubuike Ezenwoke[1], **Sanjay Misra**[2], **Mathew Adigun**[3]

[1]Department of Computer and Information Sciences, Covenant University, Ota, Nigeria, azu.ezenwoke@covenantuniversity.edu.ng

[2]Department of Computer Engineering Atilim University, Ankara, Turkey, smisra@atilim.edu.tr

[3]Department of Computer Science, University of Zululand, Kwadlangezwa, Kwa-Zulu Natal, Republic of South Africa, madigun@pan.uzulu.ac.za

Abstract: The growth of Small, Medium and Micro Enterprises (SMMEs) is important to the economic development of Africa. This growth can be greatly enhanced by leveraging IT in business activities since e-commerce is a vital tool to allow participation in globalization. Many SMMEs cannot afford to own e-commerce facilities and to reduce cost. An SMME can pay for just the e-commerce facility they use without owning the services or infrastructure. Due to the dynamic nature of the business domain, delivering such on-demand functionalities involves high flexibility in adapting to new client requirements; therefore, a systematic approach to software component reuse must be adopted to reduce cost and the time to market for new products. This work explores the reuse capabilities of a hybridization of Service Oriented Architecture (SOA) and Software Product Line (SPL).

Keywords: E-commerce; Service Oriented Architecture; Software Product line; and Small Medium and Micro Enterprises

1 Introduction

The emergence and growth of the World Wide Web (WWW) has greatly impacted businesses in various industry sectors, changing the way business is done. An aspect of business that has been largely influenced by the WWW is the sales and purchases of goods and services. This is referred to as electronic commerce (e-commerce). In the emerging global economy, e-commerce is an important tool for business development, with the potential to increase the market reach of businesses and to improve business efficiency, and, therefore, it is a veritable catalyst for economic growth [1]. In Africa, there exists some evidence of e-commerce adoption in large organizations, but there seems to be insignificant use of e-commerce tools among Small, Medium and Micro Enterprises (SMMEs).

Some of the reasons for the low adoption in Africa are a lack of Information and Communication Technology (ICT) expertise and an inability to afford the total cost of ownership of e-commerce infrastructures and services [2]. To accelerate economic growth, adequate investment must be made to support the growth of SMMEs and to leverage ICT in business operations. SMMEs should be able to afford ICT to facilitate e-commerce by paying for e-commerce services on-demand (EoD).

On-Demand Computing is a paradigm that facilitates the availability of computing resources to users on request. Utility computing is a type of on-demand computing that enables resource provisioning through payment models, such as subscription or pay-as-you-use. The term “Utility” is derived from real world provision of utilities, such as electricity, water and gas, where consumers do not own major infrastructures but pay for the resources used. In enterprise computing, software or hardware resources are accessible, as services, over a network, at the user’s request. This mode of software delivery would reduce the acquisition and operations cost of ICT infrastructures and services, such as Customer Relationship Management, on-line Payment Processing, Report Generation and Analysis, Order Management Systems, etc.

The business environment is characterized by high frequency of change. Changes could result from mergers and acquisitions, new market opportunities, new customer demands, government policies, and/or technological advancements. To maintain competitiveness, SMMEs must adapt to these changes. For an ICT-enabled enterprise, adapting to these changes means acquiring newer ICT solutions as quickly as possible, which poses a challenge to e-commerce solution developers. Traditional methods of building software from scratch cannot meet the rate of demand; hence, in response, developers must adopt newer approaches to achieve quick, flexible and cost effective application development and deployment. This quest gives rise to the concept of Software Reuse. Reuse is widely used in engineering disciplines where systems are designed by putting together existing components. In software engineering, reuse can take the form of design for reuse or design with reuse. Software reuse is the process of constructing software systems from pre-built software units instead of building software systems from scratch [3]. A crucial research question explored in this paper is: *How can we build e-commerce applications that can be customized to meet users’ specific and ever-changing needs?* From the reusability models presented in [4], we explored the reuse capability of the hybridization of Service Oriented Computing (SOC) and Software Product Line (SPL).

2 Service-oriented Product Line

SPL and SOC enable the systematic reusability of software/service components and offer the runtime flexibility required for cost and time effective, high-quality application development. The fusion of SOC and SPLE can be regarded as service oriented product line (SOPL) [5]. Consequently, an SOPL can be defined as a family of SOA-based applications. Below we present a brief overview of SOC and SPL.

2.1 Service-oriented Computing

Service-Orientation is a paradigm for developing and deploying an application quickly and cost effectively. This paradigm utilizes services as building blocks to enable the flexible composition of applications, and it is realizable through the Service Oriented Architecture (SOA) [6]. SOA is an architectural model for building systems based on the interaction of services. SOA applications are developed on the paradigm of Component Based Development (CBD). The CBD approach enables the development of software applications by assembling existing components. These components can be acquired by leveraging legacy systems, as COTS from a third party vendor, or by developing components with reuse in mind. This facilitates shorter time to market, reduced cost, and increased reuse [7]. In SOA, software components are encapsulated as Services. A Service is a software component that enables access to one or more capabilities with prescribed interfaces [8]. Services can be composed to provide higher and more complex functionalities for distributed applications [9, 10, 11].

To drive business competitiveness, enterprises prefer applications tailored to their business needs. Developers are therefore required to produce variants of software to meet such customization. Therefore, building one-of-a-kind software applications for each variant-request involves an increase in cost and longer time to market. To enable effective reuse of software components, the development process must consider the variations of these similar software products. Realizing these types of variations is non-trivial and usually results in redundant component development. Reusability in this context must therefore be strategic, leveraging existing assets, and this is achievable with the Software Product line (SPL) paradigm [12].

2.2 Software Product Line (SPL)

SPL is a set of software-intensive systems that share a common, managed set of features that satisfy 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 [13]. SPL is used for developing core artifacts that enable systematic reuse, thereby reducing time to market. However, in many SPL approaches, the reusable software components are statically configured at design-time, which makes it difficult to make any changes to the application if the context changes [14].

The key to the SPL paradigm is the modeling and management of variability. Software variability refers to the ability of a software system to be efficiently customized or to be configured for use in different contexts. Variability is achieved through variation points [15]. Variation points are places in the design or implementation that signify some variance in functionality. A variation point is resolved by selecting one or more variances from the associated set of variances possible for that variation point. However, handling variability is non-trivial. In relation to SPL, variability describes the difference between the features of the systems that belong to the product line [16]. A feature is a property or functionality of a system that is relevant to some stakeholder. The commonalities or variability among products in the product line can be represented as features. SPL features are usually captured in domain analysis using feature models. Feature models are often used to model the variabilities in a SPL.

Feature modeling, first proposed by Kang *et al.* in [16] as part of the Feature-Oriented Domain analysis (FODA), is the activity of identifying commonalities and variabilities of the products of the product line in terms of features and organizing them into a feature model. With this model, commonalities are modeled as common features, while variabilities among the product line members are modeled as variable features, from which product specific features can be selected for a particular product instance.

A variable feature can be categorized as:

a) **Alternative feature**- indicates a set of features from which only one must be present in a product.

b) **OR feature**- represents a set of features from which at least one must be present in a product.

c) **Optional feature**- means features that may or may not be present in a product.

The commonality and variability information captured in the feature model serves as a basis for developing product line assets [17].

To achieve greater flexibility, an application should be composed at runtime, and this is achieved with Service-Orientation. The fusion of SOA and SPL, as earlier mentioned, is regarded as SOPL. Consequently, an SOPL can be defined as a family of SOA-based applications. SOA and SPL are relevant to the development of e-commerce because with SOA, e-commerce solutions can be composed of reusable services, and SPL can guide the development of the SOA-based e-commerce product family. The use of Service-Oriented techniques in e-commerce development requires additional activities and artifacts that are not realizable by traditional design and development approaches such as Object-Oriented Analysis and Development (OOAD), CBD, Enterprise Architecture Development (EA), or Business Process Modeling (BPM) [18]. The need for Service-Oriented modeling techniques was identified in [6] where the author stated that current OOAD and CBD approaches do not address the fundamental elements of SOA, which are services, their flows, and the components that realize them [18]. Service Oriented Analysis and Design (SOAD) techniques would require a combination of service design methodology with other software development techniques such as EA, OOAD and BPM [19].

However, an emerging SOAD approach as that proposed in [18] can only be used to model and develop single SOA systems. This approach is made up of three key steps, which include the identification, specification and realization of services, components and flows. To create a suite of applications with each build varying slightly from the other, the service design and modeling approach must be enhanced to develop a family of SO systems. This would facilitate the evolution of variants of the e-commerce application quickly. Our approach for developing a family of Service-Oriented e-commerce solutions is a hybrid of the SOA and SPL techniques. SPL would enable the exploitation of the commonalities of e-commerce applications through the systematic reuse of shared core assets, while managing variations for a particular customer.

3 Review of Related Work

IBM proposed a method for the development of Service-Oriented (SO) systems called Service-Oriented Modeling and Architecture, or SOMA [18]. SOMA is a 3-phase method for developing SO systems. These three phases are the Identification, Specialization, and Realization of services, flows and components, with each phase having various steps. The steps for Identification in the SOMA method include the analysis and decomposition of the domain of the emerging SO system, the analysis of the business goals, and the exploration of legacy systems to identify reusable parts. Our approach adapted the first two service identification approaches of the SOMA method. We augmented the service identification process with the feature-Oriented Analysis and Design (FODA) technique [16], in order to capture the commonalities and variability in the features.

In line with the concept of service flows mentioned in [18], we conceptualized an enterprise’s business process as the flow of services in achieving business goals. In our approach, reusability is achieved by reusing orchestrated workflows of services. Reuse becomes achievable via specifying how the services would participate in the orchestration at runtime, by resolving the variation points in the business process.

Up to now, most work in the area of SPL has focused on software, but in [15], a product line architecture was proposed to design, deploy and maintain a family of web applications. The family members were composed from reusable component with an in-built variability-handling mechanism. The variability is resolved using an interactive wizard, and the system retrieves the available feature by reflection. However, in [20] it was established that a difference exists in the development of service-oriented systems compared to traditional web applications, and thus a slightly modified lightweight Product line approach with specific variability information was suggested. This approach influenced the work in [21], where a product line architecture for web service-based visual composition of web application was presented. They created domain-specific lightweight product line architecture and used a visual-based tool to configure web services and flows in the development of a specific web application. [22] proposed a feature model-based multiple view SOA variability model. The model was described in UML and SoaML and enables the systematic modeling of variability, independent of platform. In the approach proposed in our work, a visual-based tool was created that would be used by the application engineer to resolve the variation points in the variable business process. The feature modeling approach used was based on the FODA technique. The features to be included in the business process are to be specified, and through reflection, the variable business process is able to select the required variants.

In the same direction of work, Ye *et al.* [23] developed an approach for Service-Oriented Product Line Architecture (SOPLA) for business process families. The core asset of the SOPLA was a variable business process which captured the commonalities and variability in related business processes. Customization is achieved by configuring the varying elements of the business process for a particular target customer. However, it was not specified how to identify the service elements that make up the business process. Also, in [5] an approach was proposed for Service-Oriented Product Line. The approach has four activities, which are component identification, service identification, variability analysis and architectural specification. The identification activities use feature models and business process models as inputs but do not consider how to develop the services and components for the service oriented product line. The Approach in this paper used the concept of variable business process in [23]. We defined a SOPL architecture as a business processes containing all the items of the product line.

4 Our Approach

The motivation for this approach was adapted from the SOA reference architecture presented in [5, 18]. We considered a business-driven, top-down development approach. SOA defines an architectural style that provides a set of patterns and guidelines for creating loosely coupled, Business-ready services that can be recomposed in response to new business threats or opportunities [18]. Adopting SOA in developing business applications requires that the business functionalities be encapsulated as services. As stated earlier in this paper, an enterprise seeking to adopt e-commerce solutions may have a slightly different process for achieving its business objectives. SPL provides the paradigm to systematically reuse existing services, flows and components. SPL was also used to develop all the assets that will be shared by e-commerce products in the EoD product line.

To develop the approach presented in this paper, a survey of literature was undertaken, and the insights acquired were used to evolve our approach. A case study of EoD product line development was used to validate the plausibility of the approach. A visualization of the approach taken in this research is shown below.

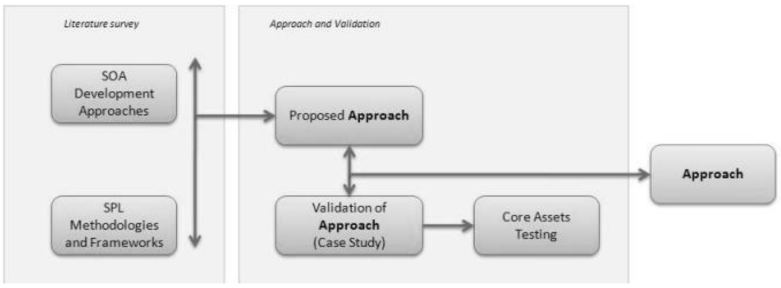
Literature survey
Approach and Validation
SOA Development Approaches
SPL Methodologies and Frameworks
Proposed Approach
Validation of Approach (Case Study)
Core Assets Testing
Approach

Figure 1
Visualization of Research Approach

Firstly, we embarked on a literature survey to analyze the state of the art in SPL and SOA practices and their relevance and application in the e-commerce domain, and we established that the paradigms can be integrated. Adopting the product line approach enabled us to develop the reusable core assets (product-line architecture, variant-rich business process, web services, etc.) required to realize customized EoD from the users' requirement. Also, a study of existing e-commerce solutions and a survey report of a group of SMMEs were carried out to identify the e-commerce requirements and features.

4.1 The Approach Process Architecture

The process architecture, presented in Figure 2, provides insights into the activities involved in our approach. It is an adaptation of the SPL and SOA development life cycles. The SPL process is divided into two: Domain Engineering and Application Engineering [24]. The Domain Engineering phase is where the core assets are developed, and during Application Engineering, products peculiar to the users' requirements are developed from the core assets realized during Domain Engineering.

4.1.1 Domain Engineering (DE)

In this phase, the construction of the core assets to be shared among products of the EoD product line is achieved. The business requirements and the features that the EoD will require to achieve business objectives are explored and this domain knowledge acquired is used to construct the domain artifacts (core assets). These core assets include generic services and flows (business processes) and a reference architecture that shows the interaction of these architectural elements and guides the construction of product line members. The Domain Engineering sub-process consists of the following stages: Domain Scoping, Domain Analysis, Domain Design and Domain Implementation. The activities in these stages are described in detail below.

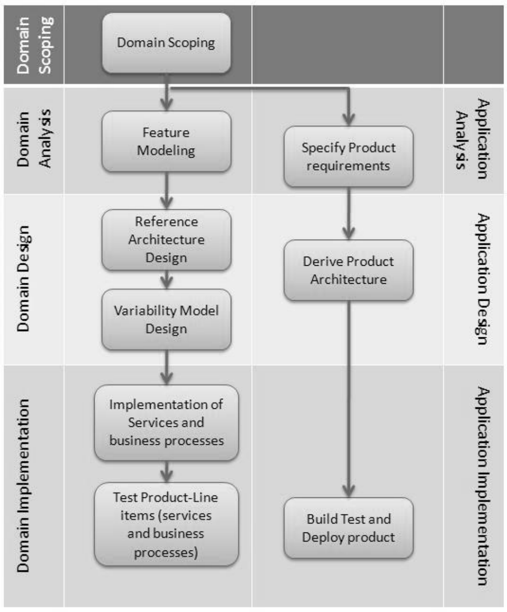
Domain Scoping
Domain Scoping
Domain Analysis
Feature Modeling
Specify Product requirements
Application Analysis
Domain Design
Reference Architecture Design
Variability Model Design
Derive Product Architecture
Application Design
Domain Implementation
Implementation of Services and business processes
Test Product-Line items (services and business processes)
Build Test and Deploy product
Application Implementation

Figure 2
Process Architecture of Our Approach

Domain Scoping

During domain scoping, the boundary of the EoD Product Line is determined. The boundary defines the characteristics of members of the Product Line, and hence the derivable variants of the Product Line. These characteristics are compiled in a product map.

Domain Analysis (Modeling)

This activity includes the collection and organization of information on systems in the e-commerce domain that share similar capabilities and contain slight differences. The information is used to produce a domain model or feature model.

Domain Design

In this sub-process, the reference architecture and Business Processes of the EoD product line are defined. The reference architecture is an abstract structure for all the products in the product line. The variability model for the product line is also specified to resolve variations in the features of products in the product line.

Domain Implementation

During this sub-process, a detailed design and implementation of the reusable service elements and the Business Processes are achieved.

4.1.2 Application Engineering

In the application engineering phase, a new product instance is derived from the core assets based on the feature model defined during DE. This new product takes all similarities from the product line and the variability is managed using a visual tool, we created to resolve variation points in the product line. The application engineering phase also has the following sub-processes, application analysis, application design, and application implementation. During application analysis, the requirements specification for the product instance is developed from reusing the requirements from the domain engineering phase. In application design, the architecture of the product instance is derived from the reference architecture. During application implementation, the product instance is realized by configuring the reusable service elements and business processes that were created during DE.

5 Case Study Implementation

In order to validate the plausibility of our approach, a case study of the development of product line (WebStore) was undertaken. Nongoma is a rural community in Kwa-Zulu Natal, in South Africa. In Nongoma, there exists the Nongoma Tourism, Arts and Crafts center, which consists of local artisans who create and sell arts and crafts artifacts. A survey has been carried out, and below is a summarized report.

5.1 Preliminary Nongoma Questionnaire Analysis

Fifteen group members of Nongoma Arts and craft Cooperative were interviewed on the 7th of June, 2005 (35 Attempted Questionnaires). Below is the summary of results:

- 70% of the members are involved in arts and crafts activities
- 10% involved in Drawing, Music and Video production
- 10% involved in agriculture activities

- 5% involved in welding and building activities
- 5% involved in catering and decorations activities.

Member's Overview

- Groups can consist of group members ranging from 3 to 25
- 75% of members are females.
- 90% of the members do not speak nor understand English.
- Members educational lever ranges from primary to secondary school (standard 2- 10)
- Members are dispersed in various segments.
- 96% of members have neither been exposed to computers nor any IT infrastructure.
- Members have a positive outlook when confronted with the idea of introducing computers into the business.

Envisioned Problems

1. Marketing and advertising
2. Funding
    a. Equipment
    b. Capital
3. Managerial and entrepreneurial skills
4. IT skills

This report formed the basis of the requirements of an e-commerce on-demand solution, called the WebStore. The requirements for the WebStore e-commerce service are:

1. A web presence
2. Online booking for customers
3. An SMS Notification or email Notification for every Order completed by a customer
4. One Monthly Advert of special offers for Business Promotion
5. A basic Accounting and Reporting Service which entitles the SMME to one end-of-month standard report

5.2 Development Process

5.2.1 Domain Scoping

In domain scoping, the boundary of the e-commerce product line was determined. This boundary defines the characteristics of members of the product line, and hence the derivable product variants of the product line. These characteristics are compiled in a product roadmap. The product roadmap captures the domain scope which is used in domain modeling to produce a domain model.

The Application-Requirements Matrix approach presented in [25] was used to capture the commonality and variability among e-commerce products in the product line. The application-requirement matrix gives an overview of the commonality and variability for a given set of software product line application requirements. The product roadmap defines common and variable features at a higher level of abstraction [25]. The column headings of the Application-Requirements Matrix are the e-commerce products and the row headings are the requirements. The requirement of the SOPL from the report of the survey of SMME conducted in Nongoma, Kwa-Zulu Natal, is shown in Table 1.

Table 1
Application-Requirement matrix obtained from Domain Scoping

SN	Requirements	Product A	Product B
1	E-commerce platform Management	Mandatory	Mandatory
2	e-commerce Store Front feature	Mandatory	Mandatory
3	Inventory Management	Optional	Optional
4	Customer Relationship Management	Optional	Optional
5	Order Management	Mandatory	Mandatory

5.2.2 Domain Modelling

All the requirements captured in the product roadmap, with further analysis are used to produce a feature model. Feature Model shows a high-level unambiguous view of the common and varying characteristics of e-commerce products within the product line. Feature models decompose concepts into their mandatory (required) and optional features to produce a set of configurable requirements. These common and variable features captured in the feature model serve as a basis for developing product line assets [17]. The feature model of the Webstore product line is presented in Figure 3.

5.2.3 Domain Design

The feature model created during domain modeling activity is the input of the domain design activity. In this activity, the domain model created was used to produce the Product Line Architecture (PLA).

The Product Line Architecture

The PLA is a structure that incorporates all the features of all the products in the product line, showing the locations where variations occur and how the variation points will be resolved. The Architecture of the product line is presented in Fig. 4.

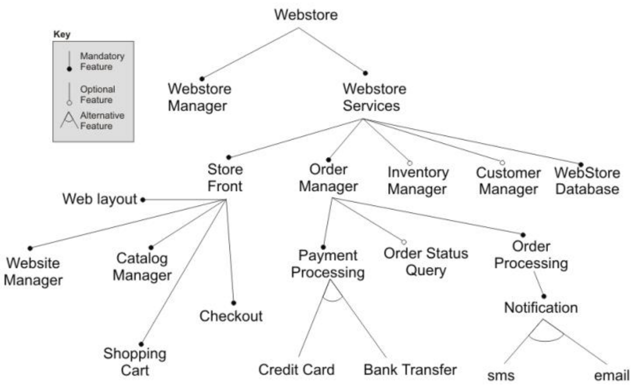

Figure 3
The FODA feature Model of the Webstore Product Line

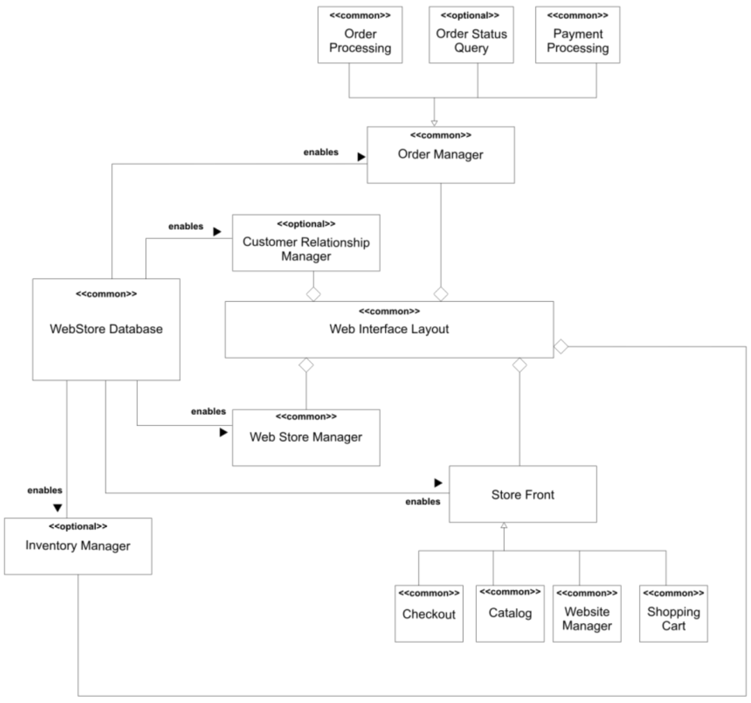
<<common>> Order Processing
<<optional>> Order Status Query
<<common>> Payment Processing
<<common>> Order Manager
enables
<<optional>> Customer Relationship Manager
enables
<<common>> WebStore Database
<<common>> Web Interface Layout
enables
<<common>> Web Store Manager
enables
enables
Store Front
<<optional>> Inventory Manager
<<common>> Checkout
<<common>> Catalog
<<common>> Website Manager
<<common>> Shopping Cart

Figure 4
The Architecture Diagram for the Webstore Product Line

List of the Web Store Business Process

The feature model and the PLA were used to identify the list of business sub-processes (components) that comprise the e-commerce products of the Webstore product line. The identified processes include: purchasing, order manager, inventory manager, and customer relationship manager. The identified processes were further analyzed to identify the various activities required to fulfill each process. Below is list of the processes and the activities/sub-processes they entail:

1) Purchasing
    a) Browse catalog
    b) Add item to shopping
    c) checkout
2) Order Manager
    a) Create new order
    b) Process payment
    c) Notify supplier
    d) Notify customer
    e) Pack and Ship product
3) Inventory Manager
4) Customer Relationship Manager

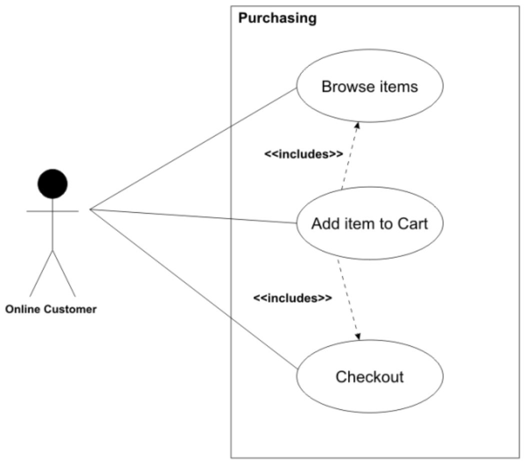

Figure 5

Use case Diagram for Purchasing Process

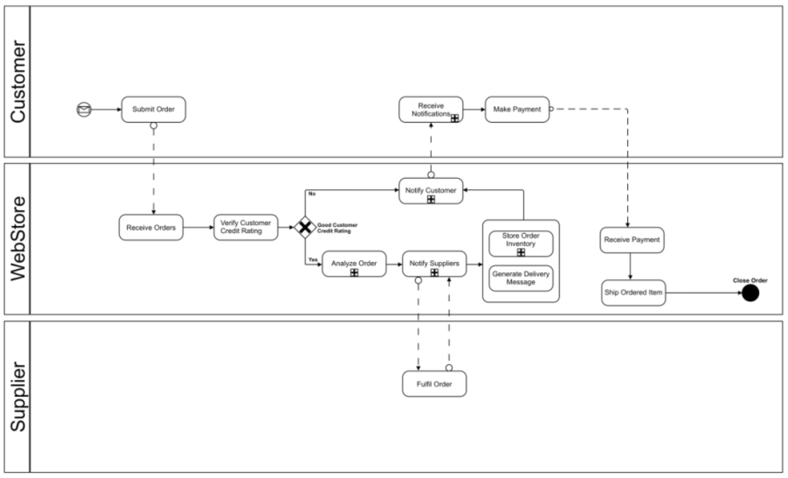

Figure 6
High-level BPMN Model of the Order Manager Process

5.2.4 Implementation Details

The implementations of web services to realize the business process were based on Java Platform Enterprise Edition (JEE) using the Netbeans 6.7.1 Java Integrated Development Environment. The web services were implemented as Enterprise Java Bean wrapped with Web Services Description Language. The EJB accessed the MySQL database for data required to deliver relevant functionality using the MySQL J-Connector. The business processes were implemented using the visual tool provided in Netbeans 6.7.1, where web services are orchestrated to provide a composite functionality using Business Process Execution Language (BPEL).

A composite application was created using the CASA editor in Netbeans 6.7.1. The composite application was deployed on the BPEL engine of the glassfish application server. The composite application was tested using the in-built composite application test case functionality in Netbeans 6.7.1. The storefront component, website management component and the WebStore Manager Component were implemented using Hypertext Preprocessor and Hypertext Markup Langauage and hosted on apache web server. The Storefront component communicated with the business process component using Java EE Servlet. The web layout template was implemented as a cascading style sheet file using Macromedia Flash and Macromedia Dream Weaver tools. The Webstore database was implemented in MySQL, which exploits the JDBC technology to connect to the EJBs.

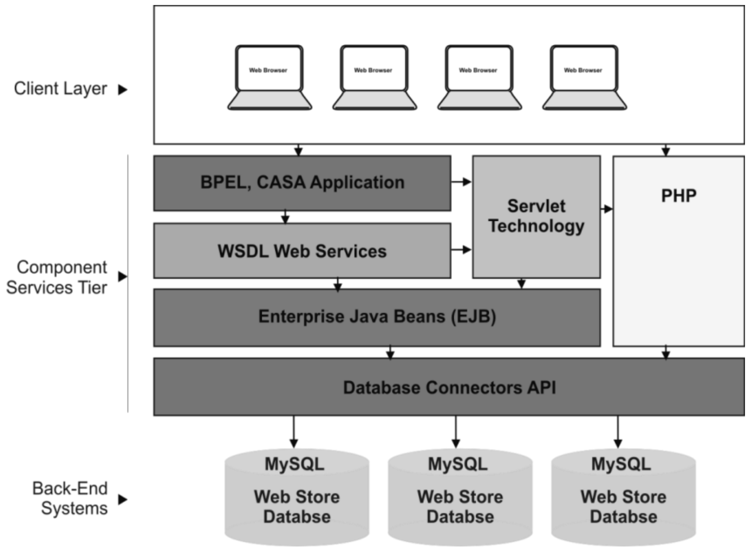

Figure 7
Deployment Technology Stack

Figure 7 shows a view of the run-time deployment architecture of the domain components. It is a 3-tier architecture showing the configuration of the content components as deployed on the Glassfish application server and Apache server in the middle layer. The data layer consisting MySQL database makes up the third layer while request for services are made through the client layer.

5.2.5 Application Engineering

In the Application engineering phase, the creation of specific products in the product line was achieved through the reuse of artifacts created during DE and exploiting the product line variability. In application engineering the parameterizable artifacts were configured with concrete parameters and composed to deliver the required functionalities. In the particular instance of our case study two e-commerce products variants were considered. These are the product-A and product-B.

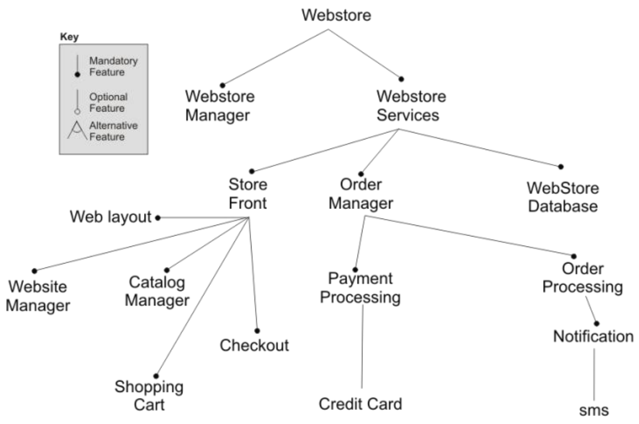

Figure 8
Feature Model of Product Instance-A

Conclusion and Future Work

To participate in the on-going global economy, access to software applications to e-empower SMME operations in developing economies is vital. Providing these technologies will enable local ES development organizations to contribute directly and indirectly to economic development [26], specifically to employment generation, wealth creation and an increase in national GDP. This contribution can be sustained and possibly increased by productive input factors that can drive the quick development and delivery of quality software products. This work demonstrated the application of SOPL to develop solutions for on-demand computing contexts, using the GUISET platform as a case study. This is important to ES developers, who have the burden of evolving solutions in line with changing user requirements. However, in this work, some required services such as credit card processing and credit card verification were known and bound to the application in advance. The use of a service broker to search and bind to required services based on QoS attributes can be explored. Quality of Service (QoS) is a major characteristic of on-demand services, but in this work, QoS-awareness was not considered in the design of the product line components. Therefore, more work would have to be carried out to explore the use of components to offer services with differentiated QoS. Also, the points of variability in the developed product line are minimal and predictable compared to a real life case study product line. It is recommended that a more elaborate case study development would have to be embarked upon to demonstrate SOPL in on-demand contexts. It is desirable that external services be considered in the modelling of product line reference architecture [27], but we assumed in our work that all the product line assets were developed under the control of the developer organization. Further work would have to be done to incorporate third party and external services as part of the variable product line reference model, and their functionalities invoked through the use of APIs. Finally, in the future, specific tools would have to be developed to model and manage the variability in SOPL, and automate products derivations easily and rapidly.

Acknowledgments

A research project, Grid-based Utility Infrastructure for SMME Enabling Technologies (GUISET), is being undertaken at the Centre for Mobile e-Services for Development. The goal of the project is to transform SMME into a global business by providing affordable access to technology on a pay-as-you-go basis. This work was done while the first author was on temporary research fellowship at the Centre at the University of Zululand. The centre is funded by THRIP, Telkom, NRF, Huawei, and Alcatel.

References

[1] Terzi, N.: The Impact of E-Commerce on International Trade and Employment. *Procedia Social and Behavioral Sciences* (2011) pp. 745-753

[2] Consoli, D.: Literature Analysis on Determinant Factors and the Impact of ICT in SMEs. *Procedia - Social and Behavioral Sciences* (2012) 93-97

[3] Krueger, C. W.: Software Reuse. *ACM Computing Surveys* (1992) pp. 131-183

[4] Sommerville, I.: *Software Engineering, 7th Edition.* Addison Wesley, (2004)

[5] Medeiros, F. M., Almeida, E. S., Meira, S. R.: Towards an Approach for Service-oriented Product Line Architectures. Proceedings of the *13th International Software Product Line Conference (SPLC).* San Francisco (2009) pp. 1-7

[6] Papazoglou, M. P., Traverso, P., Dustdar, S., Leymann, F.: Service-oriented Computing: State of the Art and Research Challenges. *Computer, 40* (11) (2007) pp. 38-45

[7] Lau, K.-K., Wang, Z.: Software Component Models. *IEEE Transactions on Software Engineering, 33* (10) (2007) pp. 709-724

[8] http://www.oasis-open.org, Accessed: 28.02.2012

[9] Karunamurthy, R., Khendek, F., Glitho, H. R.: A Novel Architecture for Web Service Composition. *Journal of Network and Computer Applications* (2012) pp. 787-802

[10] Bosin, A., Dessì, N., Pes, B.: Extending the SOA Paradigm to E-Science Environments. *Future Generation Computer Systems* (2011) pp. 20-31

[11] Li, X., Fan, Y., Madnick, S., Sheng, Z. Q.: A Pattern-based Approach to Protocol Mediation for Web Services Composition. *Information and Software Technology* (2010) pp. 304-323

[12] Andersson, H., Herzog, E., Ölvander, J.: (In Press) Experience from Model and Software Reuse in Aircraft Simulator Product Line Engineering. *Information and Software Technology* (2012) http://dx.doi.org/10.1016/j.infsof.2012.06.014

[13] Bass, L., Kazman, R.: *Software Architecture in Practice.* M. A: Addison-Wesley (2003)

[14] Lee, J., Kotonya, G.: Combining Service Orientation with Product Line Engineering. *IEEE Software, 27* (3) (2010) pp. 35-41

[15] Balzerani, L., Di Ruscio, D., Pierantonio, A., De Angelis, G.: A Product line Architecture for Web Applications. *Proceedings of the 2005 ACM symposium on Applied computing*. Santa Fe, New Mexico: ACM New York (2005) pp. 1689-1693

[16] Kang, K. C., Cohen, S. G., Hess, J. A., Novak, W., Peterson, A.: *Feature-Oriented Domain Analysis (FODA) Feasibility Study*. Software Engineering Institute: Carnegie Mellon University (1990)

[17] Lee, J., Muthig, D., Naab, M.: A Feature-oriented Approach for Developing Reusable Product Line Assets of Service-based Systems. *The Journal of Sytems and Software, 83* (7) (2010) pp. 1123-1136

[18] Arsanjani, A., Allam, A.: Service-Oriented Modeling and Architecture for Realization of an SOA. Proceedings of *IEEE International Conference on Services Computing*. IEEE Computer Society Washington, DC, USA (2006) p. 521

[19] Zimmermann, O., Krogdahl, P., Gee, C.: *Elements of Service-Oriented Analysis and Design.* from IBM developerWorks. Available at: http://www.ibm.com/developerworks/library/ws-soad1/ (2004) Accessed:21.07.2010

[20] Capilla, R., Yasemin Topaloglu, N.: Product Lines for Supporting the Composition and Evolution of Service-oriented Applications. *Proceedings of the 1st International Workshop on Principles of Software Evolution*. IEEE Computer Society Washington (2005) pp. 53-56

[21] Karam, M., Dascalu, S., Safa, H., Santina, R., Koteich, Z.: A Product Line Architecture for Web Service-based Visual Composition of Web Applications. *Journal of Systems and Software*, 81(6) (2008) pp. 855-867

[22] Abu-Matar, M., Gomaa, H.: Variability Modeling for Service-oriented Product Line Architectures. Proceedings of the *15th Software Product Line Conference (SPLC)*. Munich (2011) pp. 110-119

[23] Ye, E., Moon, M., Kim, Y., Yeom, K.: An Approach to Designing Service-Oriented Product line Architecture for Business Process Families. *The 9th International conference on Advanced Communication Technology*. Gangwon-Do (2007) pp. 999-1002

[24] O'Leary, P., Almeida, E. S., Richardson, I.: The Pro-PD Process Model for Product Derivation within Software Product Lines. *Information and Software Technology* (2012) pp. 1014-1028

[25] Pohl, K., Böckle, G., Linden, F. V.: *Software Product Line Engineering: Foundations, Principles, and Techniques.* Berlin Heidelberg: Springer-Verlag (2005)

[26] Jaakkola, H.: Towards a Globalized Software Industry. *Acta Polytechnica Hungarica*, 6(5) (2009) pp. 69-84

[27] Khoshnevis, S.: An Approach to Variability Management in Service-oriented Product Lines. *Proceedings of 34th International Conference on Software Engineering (ICSE)* (2012) pp. 1483-1486