A Requirements Metamodel Framework for Enhancing Product Adoption

Salah K Kabanda, Mathew O. Adigun, and Tarirai Chani

Abstract—The body of knowledge on Product Line Software Engineering is an emerging paradigm alternative to developing software systems from scratch. However, it will only be adopted if it helps enterprises maximize their profits. Whether this investment results in greater profit depends on the particular methodologies and strategy adopted. A number of methodologies have been proposed, with most requiring a requirement metamodel to be constructed as a basis for documenting domain artifacts using three components, namely (i) functional requirement; (ii) non functional requirement component; (iii) and a variation component, which captures variation points. In this paper we argue that a domain is not only comprised of these elements, but in addition is a merger of those components with operating context such as users, policies, and culture which affect product adoption and acceptance rate. The paper proposes a requirement metamodel framework that not only serves as a construct of the business sub process, but also serves as a basis for understanding, interpreting and predicting user's views and adoption rate.

Index Terms— Requirements Metamodel, Product Line Software Engineering (PLSE), Social Requirements.

I. INTRODUCTION

Requirements engineering is the branch of systems engineering concerned with the needs and wishes of software-intensive systems, the goals to be achieved in the software's domain, and assumptions about the domain [1, 2]. The term domain is used to denote a set of systems or functional areas, within systems, that exhibit similar functionality [3] and has led to the development of the term domain engineering which lays the foundation for the emerging Product Line Software Engineering paradigm (PLSE).

Through PLSE, it is believed that depending on the methodology or strategy adopted, substantial savings could be achieved by reusing the common features in programs that are developed as a family [4]. A number of methodologies have been proposed, with most requiring the construction of a Requirement Metamodel (RM) to document all domain artifacts in the Domain Reuse Library for application development. At the heart of each requirement metamodel is the domain requirement model consisting of three components namely (i) Functional Requirement component which describes the behavioral characteristics of the domain and gives what the system should do, i.e. the services provided for the users and for other systems; (ii) Non Functional Requirement component to describe quality characteristics; (iii) and a Variation component, which captures variation points.

Although these are domain characteristics, we argue that a domain is not only comprised of these three elements, but is a merger of those components with operating context such as the users (stakeholders, customers, employees, and other community members), policies, and culture. The users play a key role in the process and are greatly affected by the dynamic business environment. They range from novice to professionals and their adoption rate to new and innovative products is likely to differ due to the education level, culture, legal issues and governing policies.

This paper argues that the incorporation of social context into the requirement metamodel, improves adoption rate and service delivery schemes. The paper identifies societal theories and factors, and proposes a requirement metamodel framework that not only serves as a construct of the business sub process, but also serves as a basis for understanding, interpreting and predicting user's views and adoption rate. The rest of the paper is structured as follows: the next section provides a brief background on software product line engineering and the social trends in software engineering. Section III introduces the requirement metamodel and presents the need to include the social context. The section further provides a framework of the study and basis for the development of the metamodel. While Section IV illustrates a Case study, Section V concludes the paper.

S. K. Kabanda is with the University of Zululand. She is a PhD student in the Department of Computer science (e-mail: s.kabanda@gmail.com).

M.O Adigun is with the University of Zululand, KwaDlangezwa 3886,South Africa (+27 359 026 189; fax: +27 359 026 569; e-mail:madigun@pan.uzulu.ac.za).

T.Chani is with the University of Zululand. She is an MSc student in the Department of Computer science (e-mail: tariraichani@gmail.com).

II. RELATED WORK

A. *Product Line Software Engineering (PLSE)*

Product Line Software Engineering (PLSE) is a set of applications with very similar requirements and few key differences that can be configured to provide reusable assets [4, 5, 7]. Developing software for a PLSE is not an easy task as in a single product development. It requires that the family domain be critically analyzed to identify and define common and variable features that can be used to create individual product instances.

As PLSE gains popularity in software engineering, so are the methodologies evolving. Methodologies ranging from the kernel and view approach to the Feature-Oriented Reuse Method (FORM) have been proposed. Current methodologies require a distinction between the PLSE development process into (i) domain engineering and (ii) application engineering. During domain engineering, domain experts strive to identify and define product family requirements in terms of features; during application engineering, family members are produced by selecting or tailoring features that will constitute the product instance.

Literature has now introduced the use of system features as a key describing property of the domain. The motivation lies in the fact that customers and engineers often speak of product characteristics in terms of features the product has and/delivers [5, 6, 7]. Features enable techniques for developing, parametrizing and configuring reusable assets plus a specific process of commonality and variability analysis to be defined and featured in the requirement domain model.

Although product features are essential inputs for core asset development, they are not sufficient on their own as they are constantly changing with the business environment. This study requires that the business environment should also be incorporated into the domain model [5]. Although it has been suggested that a marketing and product plan perspective (MPP) be included in the domain requirement model; social contextual issues are not fully integrated within. This leaves the requirement model describing the system behavior, specifications, and business operating environment; with little reference to social needs, service delivery mechanism and user adoption rate.

B. *SOCIAL ASPECTS OF IT*

Computer systems are components in larger socio-technical networks that include human beings who use them to enhance their business operations and daily activities. Their success is determined by the community of people who use it. Although there is a broadening consensus regarding the nature of requirements Engineering (RE) as a social and technical process involving a variety of stakeholders engaging in diverse activities throughout; social and cognitive issues are not firmly addressed in designing, building, evaluating and maintaining computer-based systems. As a result, many systems that are built cannot be used as intended, even more systems are abandoned before completion due to cost and time overruns [8, 9].

Knowledge management which is often seen as a problem of capturing, organizing, and retrieving information, evoking notions of data mining, text clustering, databases, and documents; is also currently seeing the value of incorporating knowledge management systems to consider the human and social factors [10, 14] such as social classes, educational status, social status, political group and religion. These factors can create different user groups for profiling and predicting different user behavior.

III. INCOPORATING THE SOCIAL CONTEXT IN TO THE REQUIREMENT METAMODEL

Software development is usually regarded as a job for requirements engineers, domain experts, system analysts and developers. While the domain experts and requirements engineers strive to understand the domain environment to identify requirements, developers are responsible for identifying product features from the domain.

Moon et al [11] suggest that product features for a family of systems be represented using a Requirements Metamodel. A metamodel is a model that describes another model. It consists of appropriate constructs reflecting the declarations of data-definition and data-manipulation languages that represent the constructs or the building blocks of the product. The metamodel also provides a bridge between organizational structural aspects and business subprocess, minimizing the complexity of business process definition and at the same time improving the efficiency and quality of it [11, 13].

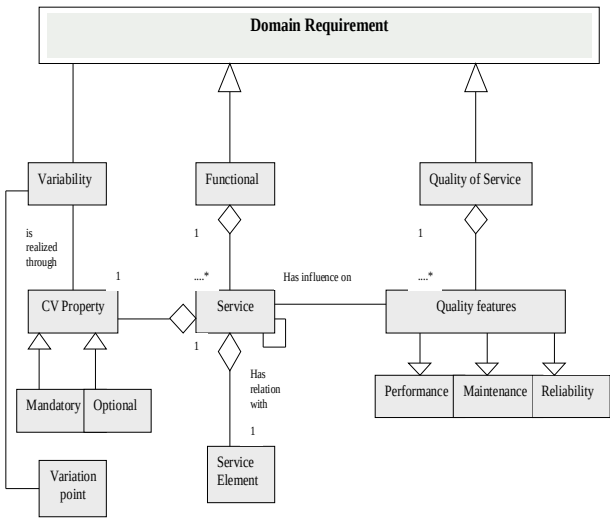

Fig. 1. Requirement Metamodel [11, 13]

Its purpose is to lay down an overall scheme for representing domain requirements. Figure 1 presents a typical metamodel for domain requirements. The core model element of a requirement metamodel is the Domain

Requirement as it represents one general requirement that can be reused as a core asset of developing systems in a product line. This usually leads to the development of a requirement metamodel. The requirement metamodel lays down an overall scheme for representing domain requirements and provide a bridge between organizational structural aspects and business subprocess, minimize the complexity of business process definition and at the same time improve the efficiency and quality [11].

It consists of Functional Requirement and Quality of Service (QoS) requirements with variants to distinguish one product feature from the other. Variations are captured using the Variability element and can either be optional, alternate or mandatory. Functional Requirement consists of services rendered to other parties within or outside the domain. A service defines the functionalities or activities which a product within the domain pursue to fulfill their goals. Services are, therefore, like use cases or scenarios. A service can have multiple sub services and can also be refined to a service element. A service element is the smallest unit that cannot be further decomposed to sub services. A service can be influenced by QoS features, such as performance, maintenance and reliability.

A. Capturing Social Context

Domain requirements are usually identified by requirements engineers or domain experts. The role entails interacting with the customers to identify user requirements, which are then documented to create a requirement specification document. This document usually does not efficiently depict the social context of the domain, but depicts the product requirements, and defines features [5] together with possible domain variations [14]. Although the domain expert's role is to capture these system features, they usually have no idea of user's acceptance rate to each product feature added. This is because the requirement specification document (RSD) normally does not display social requirements [14].

A typical RSD consists of system requirements stating system specification; functional requirements; quality requirements; and in some cases organizational and external requirements to reflect organizational policies, procedures and legislative regulations. Usually the RSD is written in a natural language which the customer understands but is difficult to integrate with the Developer's language. Factors like organizational and social requirements cannot be adequately captured in the developer's language. These are left out during development and caught up with later after the product is at its final stage. This phenomenon constitutes a communication barrier and hinders the development process, causing the right product to be marketed in the wrong market where it is not acceptable. To counteract this problem, the inclusion of a social context factors during features selection phase is crucial. We propose the inclusion of the Social Requirement component in the Requirement metamodel as in figure 2.

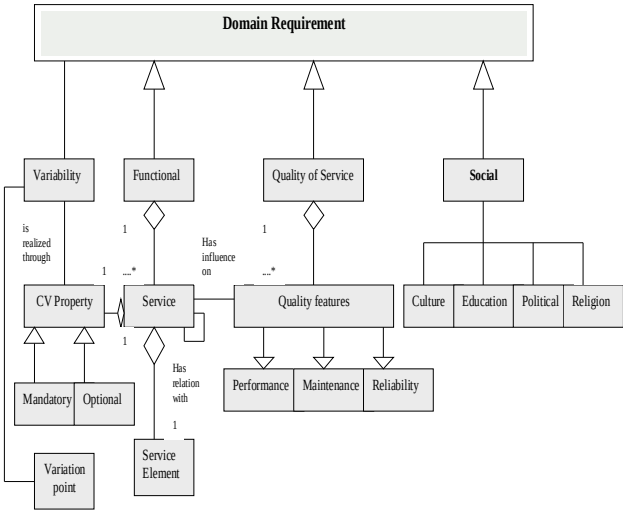

Fig.2. Refined Requirement Metamodel

During domain engineering, domain experts identify and define product family requirements in terms of features. With each identified feature, it should be mapped to specific social requirements. At this stage, the identified features are general to the domain, and thus only a general social behavior of the people is captured. Common social behavior includes cultural theories, legislation rules which bind the whole community as well as social attitude theories towards new innovations. These features are further refined during the application development when family members are produced by selecting or tailoring features that will constitute the product.

Through the pool of identified domain social factors, features are added and selected to suit the users. These social factors include cultural values and ethics, moral issues and assumptions shared by a group or a subgroup during their product development. This helps to continually remind them of the social context of the environment in which the product will be assimilated and helps them identify the cultural shifts which might point to new products that might be wanted by users or to increase demand. Cultural shifts could be as a result of wanting more leisure time, or telecommuting workers. Different classes or groups differ in the type of features they prefer in the product and their adoption rates to the product.

Further more, in developing countries, access for people with low literacy rate and people not fluent in the median language in which the product is presented, can be a stumbling block in the products success. Poor education background of users could result in false perceptions on decisions to buy or not to buy the product. Thus software engineer's literacy and understanding of the society is essential to ignore false perception about the community they intend to serve. Although most think that by training and providing support to the users, it is enough to make them adopt and accept it, this is usually not true in developing countries where technology is just picking up pace and is perceived as foreign. In addition, understanding the legislation matters which each country adopts and is

regulated upon is crucial in enhancing product adoption. Legislation includes safety rules that are concerned with possible loss, damage or harm resulting in the use of the product; security and privacy requirements which are concerned with protection of the data used or created by the product, identity authentication requirements and the policies and regulations contained in using the product. In addition, legislation needs to ensure that the product incorporates specific features for all users, that is, start making products that are accessible to users who have disabilities. For example, people with disabilities have unnecessary difficulties using the Web, and in some cases, cannot effectively use the Web at all [12]. Accessibility also includes product dissemination to all users who reside in both rural and urban areas. Thus software developers need to work hand-in-hand with social sciences and business management disciplines to identify or tailor domain features that are acceptable to constitute product instances during the application development phase. This adversely results in a rich Requirement Metamodel that incorporates factors that affect user adoption rate and buying decision.

B. Requirements Elicitation

Use Cases have been proposed as an effective approach for Product Line Software Engineering elicitation to capture product features (especially functional requirements) for software systems [15, 16]. Table I provides a clear illustration of a use case constitutes. Use cases allow structuring of requirements documents with use goals and provides a means to specify the interaction between an actor and its environment. The term actor is used to describe the person or system that has a goal against the system under discussion. There are two main actors namely the primary and the secondary actor.

TABLE I
USE CASE TEMPLATE

Goal in context	<a longer statement of the goal in context if needed>	
Scope and level	<what system is being considered black box under design>	
Precondition	<what we expect is the state of the world>	
Success end condition	<the state of the world upon successful completion>	
Failed end condition	<the state of the world if goal is abandoned>	
Primary and secondary actors	<a role name or description for the primary actor> <other systems relied upon to accomplish the use case>	
Trigger	<the action upon the system that starts the use case>	
Description	Step	Action
	1	<steps of the scenario from trigger to goal delivery, and any clean up>
	2	<……>
Extension	Step	Branching action
	1a	<condition causing branching> <action or name of sub-use case>
Sub Variations	Step	Branching action
	1	<list of variations>

While a primary actor triggers the system behavior in order to achieve a certain goal, a secondary actor interacts with the system but does not trigger. A use case is completed successfully when its goal is satisfied and is extensively described in the use case "Description". Use Case descriptions also include possible extensions as reflected in Cockburn's Use Case Template in Table I. The "Description" section is expressed in natural language sentences, describing a sequence of what actions of the system while Variations are expressed (in the "Extensions" section) as alternatives to the main flow, linked by their index to the point of the main flow from which they branch as a variation [5].

Variations are described and specified by tags that indicate those parts of the product line requirements needing to be instantiated for a specific product in a product related document. The tags represent three kinds of variability:

(i) Alternative, expressing the possibility to instantiate the requirement by selecting an instance among a predefined set of possible choices, each of them depending on the occurrence of a condition;

(ii) Parametric/Mandatory, from which instantiation is connected to the actual value of a parameter in the requirements for the specific product and;

(iii) Optional of which the instantiation can be done by selecting indifferently among a set of values, which are optional features for a derived product.

IV. NONGOMA SMES AS A CASE STUDY

Nongoma is a deep rural area in North KwaZulu Natal characterized by high illiteracy and poorer conditions as compared to other rural areas as it is situated more than 5km from a tarred road. The case study involved enabling Nongoma's Small and Macro Enterprises reap the benefits of collaborative commerce by developing a localized/ customized e-Commerce solution. This includes building an innovative enabling platform based on grid computing, to promote a new collaborative business model for SMEs. The generic nature of the Grid computing approach allows customization of the functions of the platform according to specific business requirements. South Africa's rural SMEs are characterized by high illiteracy rates and poor economic growth. For the e-Commerce solution to be applicable in Nongoma, we need to have an understanding of its social context in order to have a pool of knowledge sufficient enough to develop a product that not only meets their productivity but also meets their social needs.

A. Requirements Elicitation

The grid infrastructure is to provide a foundation for SMEs to perform their business online with other business partners. The grid infrastructure incorporates a portal which provides services in two main product categories namely, the ordinary portal services and the mobile portal services for mobile users. Each product category is differentiated into three different instances depending on the number or level of features. The instances include: (i) Mini portal features: These are necessary features that are required for a portal to be functional such as Catalogue Management which includes controlling and displaying product range; and an Ordering System responsible for placing and processing customer orders; (ii) Standard

portal features that include mini portal features and additional features that offer differentiated characteristics eSales Management for orders and billing management, reports generation and analysis, marketing and advertisement though banners; and Procurement which includes handling inventory and integrating transaction between buyers and suppliers either by event notification such as SMS or email; and (iii) Deluxe specialized features that include both mini and standard portal features. Compulsory features implemented will include Catalogue Management; eSales Management; Procurement; Customer Relationship Management, ensuring provision of customer self services and direct marketing using PDAs for subscribed customers; and Website Management to handle content layout, web load testing and functional testing. A use case template is then set up to capture both domain and social requirements; analyze and documented them to comprise preliminary domain knowledge.

Table II illustrates the elicitation of cultural factors as social requirements. The Use Case name is Culture and the goal is to successfully map the system specification to meet cultural values, with the aim of enhancing easier user adoption and acceptance of the product.

TABLE II
CULTURE USE CASE TEMPLATE

Goal in context	Understanding whether the product is in alignment with cultural beliefs of the target market	
Scope and level	South Africa, Zulu Kingdom	
Precondition	Cultural theory adherence	
Success end condition	Product adoption	
Failed end condition	Product conflicts with cultural values	
Primary and secondary actors	Zulu speakers, external parties interested in the product	
Trigger	Ethics	
Description	Step	Action
	1	Field study of Zulu beliefs
	2	Understand business models used by Zulu kingdom
Extension	Step	Branching action
	1	Beliefs
	2	Business practice
	3	Product Adoption

The Use case is triggered once the Ethical behavior and moral values of the target market is questioned. The final product is targeted for South Africa where the case study is being conducted but specifically to rural SMEs in the Zulu Kingdom tribe. The Zulu tribe becomes the primary actors who are to either accept and adopt or reject the product. In fulfilling the use case, a number of activities have to be followed including: (i) a field survey of the Zulu Kingdom to understand their culture; (ii) a survey of their existing business models, to identify ways in which they can be adjusted to accommodate the new product. In studying the culture, different dimensions can be taken upon, such as their beliefs, attitudes, awareness, and business practices.

B. Requirements Specification

The requirements specification process requires that an actor selects features from predefined domain knowledge that serves as dynamic content repository corresponding to a portal application on the server. Figure 3 shows an actor specifying the domain types, which in this case study, is the e-Commerce domain.

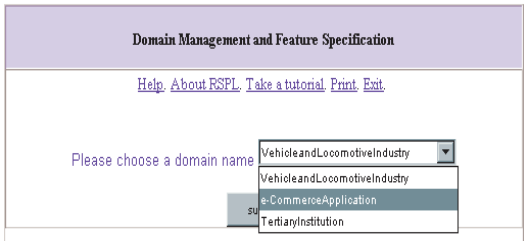

Fig.3. Domain Specification

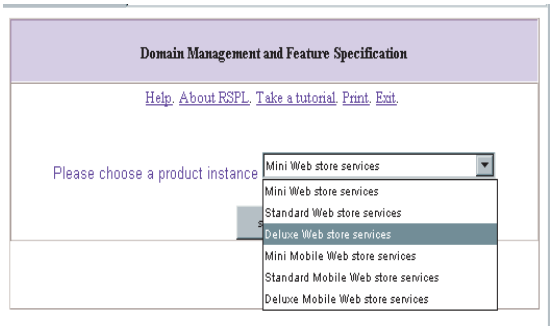

Fig.4. Product instance specification

Fig 5. Elicit Social Requirement

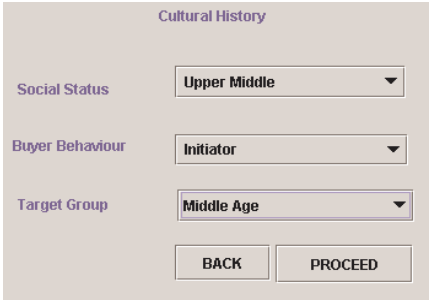

Fig 6. Sub Factors of the Cultural Feature

The different product instances that characterize the domain are displayed as in figure 4 and the actor makes a choice of for example, deluxe features that offers specialized services. After eliciting the domain and product, the domain expert, together with the social sciences and management disciplines, uses the Social Context requirement template to document gathered knowledge in the system - Fig 5. Each requirement can be further refined to its simplest element. For example in Fig 5, once the Culture was selected, the tool automatically uploads pre-defined requirements, Fig 6, to identify the specific social status and buying behavior of that culture with respect to other cultures. This serves to properly document social user

requirements. Software developers can now easily retrieve this data to map product features to what users want and subsequently define and add in product features suitable for that particular culture or subset of a group.

V. CONCLUSION

As software development moves from a single product development to a family of systems, requirements engineers need to rethink new models for tool support. The tool support is an essential part in software development as it (i) decreases time to market by automatically generating requirements or code (ii) decrease errors due to limited human intervention (iii) increases return on investment by decreasing overhead costs. It is therefore essential that the employed tools not only capture functional requirements but also captures social features necessary for enhancing product adoption and easier acceptance among the target market group. However, this can only be achieved if the methodology employed for system development does have a social elicitation capability right at the Requirements gathering stage. This paper recommends the refinement of the requirement metamodel to incorporate social features. In future we intend to use the Technology Acceptance Model to assess the product acceptance rate in the SME community and to the developers.

ACKNOWLEDGMENT

This paper is part of the research work conducted in the center for mobile eServices for Development, University of Zululand, under the sponsorship of Telkom, Huawei and Thrip. Sincere thanks to our sponsorships and partnerships.

REFERENCES

[1] R. Wieringa, and C. Ebert, "RE'03: Practical Requirements Engineering Solutions", IEEE Software Computer Society, 2004.

[2] D. Tomas, and A. Hunt, "Nurturing Requirements", Software, IEEE Volume 21, Issue 2, Page(s):13 – 15. Mar-Apr 2004.

[3] S. Jarzabek, O. W. Chun, and H. Zhang, "Handling Variant Requirements in Domain Modeling" Journal of Systems and Software, 2003.

[4] D.L. Parnas, "On the design and development of program of families. "IEEE Transactions on Software Engineering, vol.2, no. 2, Mar.1976.

[5] K. C. Kang, J. Lee, and P. Donohoe, "Feature-Oriented Product Line Engineering." IEEE Software 19, 4: 58-65. 2002.

[6] W.N. Kassel, and B.A. Malloy, "An Approach to Automate Requirements Elicitation and Specification". Proceedings of the 7th IASTED International Conference Software Engineering and Applications, November 3-5, 2003

[7] K.C. Kang, Kim, S. Lee, J. Kim, K. Shin, E. Huh, "FORM: A feature-Oriented Reuse Method with Domain-Specific Reference Architecture", Annals of Software Engineering, Volume 5, Number 1, pp. 143-168(26). 1998.

[8] M.S. Ackerman, The Intellectual Challenge of CSCW: The Gap Between Social Requirements and Technical Feasibility; Human-Computer Interaction, Vol. 15, No. 2&3, Pages 179-203. 2000

[9] J. Goguen, "Social Aspects of Information Technology", Available: http://www-cse.ucsd.edu/~goguen/projs/soc.html 2005

[10] J.C.Thomas , W.A. Kellogg and T. Erickson, "The knowledge management puzzle: human and social factors in knowledge management", IBM Systems Journal, Available: http://www.findarticles.com/p/articles/mi_m0ISJ/is_4_40/ai_82373859. 2001

[11] M. Moon, and S.C. Heung, "An Approach to Developing Domain Requirements as a Core Asset Based on Commonality and Variability Analysis in a Product Line", IEEE Transactions on Software Engineering, Vol. 31, No.7 July 2005

[12] S.L. Henry. "Social Factors in Developing a Web Accessibility Business Case for Your Organization", WAI: Strategies, guidelines, resources to make the Web accessible to people with disabilities Available: http://www.w3.org/WAI/bcase/soc

[13] Schmid, K. A Comprehensive Product Line Scoping Approach and Its Validation. International Conference on Software Engineering, Proceedings of the 24th International Conference on Software Engineering. pp. 593-602. 2002.

[14] S.K Kabanda and M.O Adigun. Importance of Social E-Factors for Grid based Infrastructure. Southern African Telecommunication Networks and Applications Conference 2006.

[15] Fantechi, A. Gnesi, S. John, I. Lami, G. and Dorr, J. (2002) Elicitation of Use Cases for Product Lines. 9th IEEE Conference and Workshops on Engineering of Computer-Based Systems. Available at http://fmt.isti.cnr.it/WEBPAPER/25-PFE-ucf.pdf Date last accessed March 2005

[16] A Cockburn, Writing Effective Use Cases. The Crystal Collection for Software Professionals. Addison Wesley 2001.

Miss S. K. Kabanda is with the University of Zululand where she works as a lecturer in the department of Computer Science. Miss Kabanda obtained her first degree at the University of Venda in Business Information Systems. Her Honors degree was in Business Information Systems at the University of North West and her Master of Science in computer science at the University of Zululand. She is currently pursuing her Doctorial degree.

Prof. Mathew O. Adigun holds a Computer Science doctoral degree and is head for both the department of computer Science and the Center of mobile eServices at the University of Zululand.

Miss T. Chani is with the University of Zululand where she is currently enrolled for a Master of Science degree in Computer Science.