

An Adaptive User Interface model using a Service Oriented Architecture

E.K. Senga, A.P. Calitz., and J.G. Greyling

Department of Computer Science and Information Systems

Nelson Mandela Metropolitan University, PO Box 77000, Port Elizabeth, 6301

Tel:041-504-2247 Fax:041-504-2831

Email: {Emile.Senga, Andre.Calitz, Jean.Greyling}@nmmu.ac.za

Abstract— The advancement of computing technology has allowed organisations to store and manage an increasing quantity of data and information on a variety of platforms. Integrating systems on these platforms by way of enterprise architecture paradigms such as Service Oriented Architecture (SOA) has become popular. However, the focus of this paradigm has been on functional services and the communication between services. The alignment between user interfaces (UI) and SOA is not widely discussed. As a result, SOA implementations do not take advantage of the potential of SOA in the delivery of the UI to end users. This problem is the main focus of the study. Furthermore, presentation of information is not based on the characteristics of the end users.

In this paper a model is presented which utilises an SOA architecture and web services to generate an adaptive UI (AUI). AUI techniques are employed to adapt the UI of web services depending on the inferred expertise of users and the service profile.

Index Terms—SOA, User Interface, Personalisation, Adaptive User Interface.

I. INTRODUCTION

Integration of disparate information systems using enterprise architectures such as SOA has gained popular support in recent years. This support has increased due to the benefits that SOA advocates, such as loose coupling and interoperability of system components. Analysts predict that 80% of mission critical systems will be based on SOA by 2010 as organisations increase adoption of SOA in order to realise its benefits [2], [16], [17].

Designing systems using SOA increases interoperability between organisational units (e.g. different departments or different organisations) aiming to share information by exposing applications as services. Currently, a plethora of definitions for SOA exist. For this study, however, we define SOA as ‘An architectural style whose goal is to achieve “loose coupling” among interacting and contracted services via communication protocol’ [17]

SOA advocates the design of computer systems using services. The OASIS group defines a service as “a mechanism to enable access to a set of one or more capabilities, where the access is provided using a prescribed

interface and is exercised consistent with constraints and policies as specified by the service description” [14]. Services are the most fundamental components of SOA, each representing a unit of logic which can be atomic (simple calculation logic) or composed of more than one service. Services communicate using standard messaging protocols such as the Simple Object Access Protocol (SOAP).

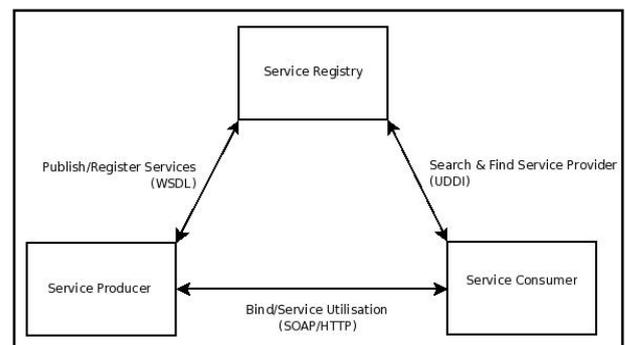


Figure 1: Service Interaction Model

Figure 1 shows the service interaction model. The service registry allows service producers to register and publish services which service consumers are able to access. Service consumers search the registry to find the most suitable service using a protocol such as the Universal Description, Discovery and Integration (UDDI) protocol. Consumers bind to the service provider after which they can begin consuming the service by invoking it via SOAP or HTTP. Service consumers can be human users, composite applications or other services exploiting the capability provided by the service [6], [12]. A service must therefore have the following characteristics [17]:

1. It must be accessible via a clearly defined service interface which describes the service capability, the input parameters and outputs (for example, service A attempting to interact with service B uses service B’s interface to establish a connection). A service itself is the interface for a business function or functions (it can have many operations). In web services the defined interface is described in the Web Service Description Language (WSDL);
2. It must correspond to real-life business activities. A service is the interface for a business function or functions and it can have many operation;

3. It must be agnostic of other services in the environment in which it is deployed to minimise dependencies and coupling.
4. It must use established standards for communications so as to maintain interoperability.

The focus of information exchange has been on data exchange between communicating web services [22]. SOA as a design paradigm focuses on units of computation which are services. The UI is not traditionally seen as a unit of computation, but as a means for computer applications to interact with users. Additionally, SOA evolved from the client-server architecture to deal with problems of data exchange and as such, the principles of loose coupling and re-usability of defined services in SOA have not been carried over to the UI [4].

Certain web services require input directly from users; therefore, a UI is required. Currently, UIs used for such input are not extremely elaborate and typically consist of a form based UI [7]. This gives rise to the opportunity for applications to be created entirely from web services that users can access remotely. For example, applications running on computers in remote areas would not require UI development; the combination of services will create the application and a UI generated at runtime. This concept is well motivated in past works [7], [9], [19]. In this study, web services that are capable of receiving input directly from the user are referred to as user-facing web services.

Before elaborating on this opportunity, it is important to understand what a UI is. The purpose of the user interface is three fold:

1. to allow users to interact with applications;
2. to allow users to retrieve information; and
3. to allow users to author information.

The interface acts as a layer between the user and the underlying system functionality. In SOA, application functionality is distributed; therefore, user interfaces that are not tightly woven into the underlying application are most common. Such UI include [5]:

1. Thin Clients: the presentation layer is hosted on the client side, with system logic and data residing on different servers;
2. Portals: aggregate information from multiple sources and act as an entry point for users to access services;
3. Rich Clients: complex applications with functionality distributed across the web or network and
4. Smart Clients: stand alone applications that hold additional content on the web or network.

The UIs of applications that use the above techniques must still be designed, implemented and deployed before they are usable. Analysis of the application must take place followed by development and testing of the UI. In addition, reusability of such UI's entails adding components to the application or altering application code in order for the UI to function. This lack of integration between the architecture and its user interface is referred to as the missing link in SOA [4]. The steps above can be eliminated if the underlying architecture is exploited using techniques such as embedding UI meta-data into web services (which is subsequently interpreted into a workable UI or using a framework to generate UIs for the web services [7], [9], [19].

Generating UIs during interaction with users implies that UI artefacts can be embedded with data at run time, or placed in the optimal position on screen in order to increase the flow of information between users and computers. Information about the user can be collected and inferences about the user made. Consequently the user interface can be adapted as more information is collected. Figure 2 illustrates this cycle of data collection, inferring characteristics about the user, adapting the user interface, and collecting more data. Adaptation of the user interface improves the flow of information between humans and machines. This is vital in a generated UI as users may require assistance with using a UI that is created on the fly as parameters change and affect the UI). In this study, user expertise is inferred based on specific user metrics and this information is used in the process of adapting the UI.

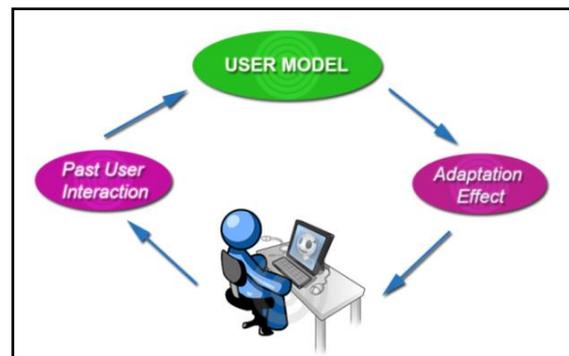


Figure 2: Interaction of an AUI with a Human User

In this paper a model for adaptive web service UI generation using user expertise and service profiling is presented. The sections of the paper are as follows:

1. Related work in web service UI generation, and models for this.
2. The field of Adaptive User Interfaces (AUI) is explored, and the literature regarding its components is examined.
3. User expertise literature is investigated to uncover UI design implications of user expertise.
4. Web service profiling for characterisation of web service is discussed.
5. The proposed model is presented.

II. RELATED WORK

The following sections discuss related research in web service user interface generation, AUI, web service profiling and analyses literature in the field of user expertise.

A. Web Service User Interface Generation

There is limited research in web service user interface generation. The Web Services Graphical User Interface (WSGUI) is a general approach to supplement web services with user interface descriptions stored in a number of UI meta-data files constructed by a designer during the development of the web service. The UI meta-data separates the display of the web services' user interface from its implementation, thus maintaining the separation of concerns between a web service's implementation details and its appearance [9]. Dynvoker extends the WSGUI approach by using XForms to display the user interface. A separate approach that is similar to Dynvoker, maps the WSDL data

types to XForms input types and then generates XForms UIs [19]. Another approach generates multimodal user interfaces by using abstract UI notation to define the UI [21]. The Adaptive User Interface Generation Framework, from which the model in this paper is partially derived, is based on constraints and preferences. It uses device constraints and user preferences in the generation of the web service user interface [7].

The main components of this model include:

1. GUI Display and Layout framework;
 - a. UI Objects
 - b. Object Layout Hierarchy
 - c. Global Layout Preferences
 - d. Basic Layout Constraints
2. Adaptive Interface Generation Framework.
 - a. Specification documents

The most important component in this framework is the Object Layout Hierarchy (OLH) which specifies the relationship between UI objects as a hierarchy of groups. The hierarchy is represented by nested groups and each group belongs to one of the defined types: plain group, ordered group, plain horizontal group, plain vertical group, ordered horizontal group and ordered vertical group. The group types represent the general layout for all UI objects in that group. This allows UI elements to be grouped and laid in meaningful ways.

Conceptually, the frameworks and processes discussed in this section function in much the same way. The process begins with the web service definition to establish the basic requirements of the service in terms of inputs, outputs and data types. Additionally, some approaches make use of external definitions such as semantic mark-up or object hierarchies to refine the UI generation process. These are processed and a user interface is presented to the user. The processing and presentation of the UI is a fundamental aspect required of a solution that adapts the user interface based on user expertise and service profiling. The ability to generate a UI for a web service implies that by introducing a component that manages the parameters used in generating the UI, the interface can effectively be manipulated to suit the user. The proposed model discussed in this paper elaborates on the implications of managing UI generation parameters.

B. Adaptive User Interfaces

AUIs are defined as *Systems which can automatically alter aspects of their functionality and/or interface in order to accommodate the differing needs of individuals or groups of users and the changing needs of users over time* [2].

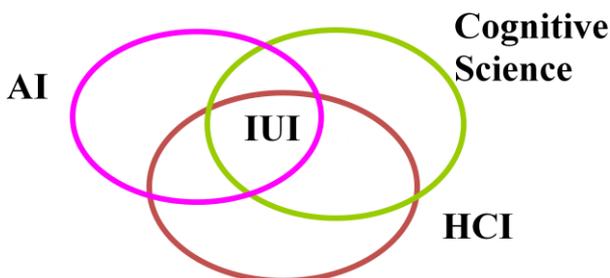


Figure 3: IUI research context [3]

AUIs are a subset of Intelligent User Interfaces (IUI) where inferences made result in adaptation of the user interface.

IUIs are a major, multidisciplinary research area in Human Computer Interaction (HCI) and the aim to improve the efficiency, effectiveness, and naturalness of HCI by representing, reasoning and acting on models of the user, domain, task, discourse, and media [12]. It encompasses fields such as artificial intelligence (AI), HCI and cognitive science (Figure 3). These fields of research merge in determining the optimal methods of increasing the flow of information between humans and computers.

Benefits of IUIs to users include personalisation of the user interface by context, content or form, and assistance with task completion. Intelligent interfaces are able to support different modes of interaction for users and are capable of making inferences about the information requirements of a user in order to complete a task. In this regard, IUIs differ from traditional user interfaces [11]. AUIs can generally be described as consisting of 3 components, namely:

1. Afferential: acquires information that is used by the AUI. This is information on which inferences are made in order to adapt the UI. This information is stored in a model which represents the user, the domain or the system.
2. Inferential: uses acquired information (stored in models) to infer what adaptations should occur.
3. Efferential: applies the desired adaptation to the user interface. There are different levels of efferential adaptation ranging from information level (information is adapted), through to the user interface level (the user interface is adapted to the user) and finally the functional level (systems functionality is adapted to suit the user).

The illustration in Figure 2 shows the interaction cycle between the user and an AUI. The user's actions are stored in a user model (part of the afferential component), from which inferences are made in order to adapt the UI, which in the diagram is represented by "past user interaction". This is a continuous cycle of observing the user, collecting data, making inferences, adapting the user interface and back to the beginning [8].

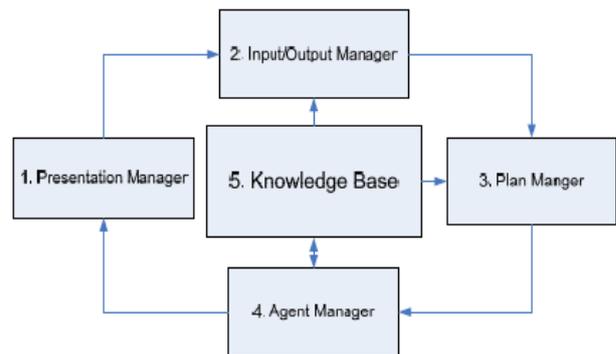


Figure 4: AUI Model

Figure 4 [18] shows the main components of an AUI model. These components are critical for the collection, storage and inference of the model. The presentation manager is the efferential component and deals with presenting intelligence via the UI to the user. The Input and Output Manager are the afferential component which captures user events and directs it to the plan manager. The

plan manager is the inferential component used to determine a user's overall goal or plan. The Agent Manager is an intermediary between the plan manager and the rest of the system and the knowledge base is the repository of knowledge used by the intelligent interface. The user, task, system or domain models are stored in the knowledge base. These components work together to achieve the cycle described in Figure 2.

C. User Expertise

The main function of AUIs is to adapt to the user in order to increase the flow of information between the computer and the user. Research shows that user expertise can be classified based on two criteria: user's knowledge of a system and time spent using that system [8], [15], [23]. These criteria can be decomposed further into 3 main dimensions:

1. Experience with the system;
2. Experience with computers in general; and
3. Experience with the task at hand.

Experience with the system refers to the application software, and to what degree the user has used similar applications. Experience with computers in general refers to the user's computer literacy. Experience with the task at hand refers to users experience with the task the system will be performing.

Novice and expert users have been shown to exhibit different ways of thinking, and these are defined as qualitative differences. Novice users exhibit fragmented conceptual models of the system, and are therefore more concerned with *how* to accomplish tasks as opposed to how *fast* they can accomplish tasks. Expert users, in contrast, exhibit a consolidated model of the systems inner workings and are able to infer new knowledge to achieve their goals. As such they are more efficient at completing tasks and thus require fewer steps [8].

The implications for user interface design are thus as follows: novice users require step by step instruction to allow them to gain knowledge about the inner workings of a system. Information to aid them with their tasks must be provided at every major step in the user interface. Expert user interfaces on the other hand need to be efficient and uncluttered and whenever possible afford the expert shortcuts to aid with efficient task completion [8].

D. Web Service Profiling

An AUI normally alters its appearance to suit inferred characteristics of its users. This inference is based on input parameters that may be from the user, such as performance parameters. However, the user interface can also adapt based on non user parameters, and still benefit the user. User-facing web services have different characteristics such as varying granularity and different parameters of different types. Granularity refers to the quantity of operations exposed by a service in relation to the total number of services. Dense granularity means there are a high number of operations amongst a small number of services, while sparse granularity means there is a large number of services, but each has few operations. UIs that are generated may represent a single service or a combination of services. Establishing the granularity of the web service prior to creating the UI allows us to determine optimal page layout. For example, three dense web services could be displayed on multiple pages because there are a large number of

operations, while three sparse web services only require a single page for interaction with the user.

Additionally, some parameters (not necessarily from the same service) may share a semantic relationship. For example, two unrelated web service operations may have parameters *billing address* and *home address* respectively. Services may see the parameters as unrelated, but to users these are both addresses. Displaying these elements without showing the *address* relationship would confuse the user.

Web services profiling allow us to classify web services to facilitate the generation of user interfaces. The following section explores SOA implementation frameworks (SOAIF) that could be used to realise a prototype for evaluation.

III. SOA IMPLEMENTATION FRAMEWORKS

SOAIFs provide the tools for development and runtime environments for SOAs. Commercial and open source alternatives were evaluated for this study to establish the best option, based on supported features that allow for effective SOA implementation and supported features. Some metrics against which the frameworks were compared include:

1. Support for service development;
2. Support for application integration;
3. Enterprise standards support;
4. Efficient Process composition; and
5. Service Debugging.

It is important that these products also support Enterprise Service Bus (ESB), Business Process Management and Service Oriented Integration. Products considered include Microsoft Connected Services Framework, OpenESB, Oracle SOA Suite and SAP EA. The environment selected for this study was OpenESB. This environment was selected as it is open source and is therefore freely available with a large community for support. OpenESB also adheres to all the requirements for a SOAIF.

IV. PROPOSED MODEL

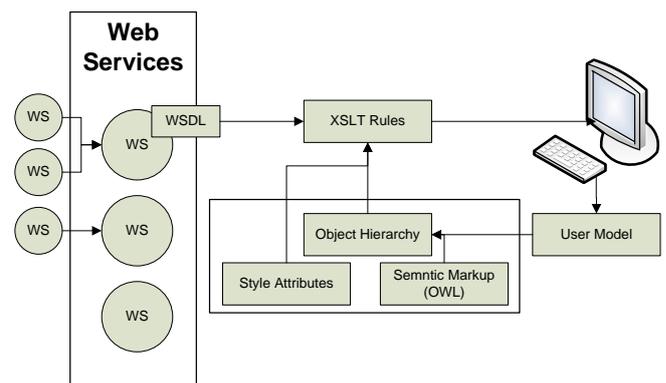


Figure 5: Proposed Model

The proposed model attempts to shift the focus from message exchange between services, to integration of user interfaces for web services which will allow seamless interaction with the web service. Figure 5 illustrates the proposed model of this study. The main components are the

1. web service layer,
2. the extensible style-sheet (XSLT) rules,
3. the mark-up data (object hierarchy, style attributes and semantic mark-up) and
4. the user model.

The web service layer is a collection of web services that are accessible via their WSDL interfaces. The web services are developed as Enterprise Java Bean (EJB) modules and are deployed on a GlassFish v3 Application Server using OpenESB [13].

The model functions as follows; when a web service is invoked, its WSDL is enumerated using the **XSLT rules** to obtain operation information. The operations are parsed to establish any “complexTypes” associated with operation input and output messages (Figure 6).

```
<xsd:complexType name="getProdName">
  <xsd:sequence>
    <xsd:element name="ProdID" type="xs:string" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

Figure 6: Schema definition for complex type "GetProdName"

Complex Types define object types that are passed as messages between services. From the complexTypes, simple types (e.g. type string, int or double) are derived for every input and output parameter associated with an operation. From this information, a basic form can be created for data input using XSLT rules. For example, in Figure 6 the complex type “getProdName” which can consist of a sequence of elements, has a single element “prodID” of type “string”.

The XSLT rules map the data types to input elements. For example, the type string maps to the text input element, boolean to radio buttons and restriction/enumeration to a drop down list [7], [19]. Table 1 shows the mapping rules used in this study. The mapping allows the model to generate UI elements depending on the input or output parameters of an operation.

XML Simple Data- Type		XHTML Control
Built-in Type		Input
Restrictions	String of restricted length	Input / Textarea
	Selectable items	Select
	Range	Range select
Lists	List of selectable items	Select
Union of various items		Composition of controls
Unknown		Input

Table 1: Mapping Rules of XML simple data types to HTML Controls

XSLT rules are also used to create the user interface by transforming the information derived from the WSDL in combination with data from the OLH, semantic mark-up and the style attributes. Operation information derived from the WSDL determines which labels and UI objects to display in the user interface.

The OLH defines the relationship between elements that appear in the UI. The OLH is an XML file and is used during the transformation stage to determine the layout of the UI objects. This component is derived from the user model and the WSDL of a web service. Optionally, it can include **semantic mark-up** data which further defines relations between UI objects. The **style attribute** define the look and feel of the UI Objects such as colours, width and

length. Since the proof of concept is HTML based, the style attributes are defined using Cascading Style Sheets (CSS).

The **User Model** forms part of the AUI, and its function is to store user expertise information. Whenever the user interacts with the system, the user model is updated with user information such as user preferences (for layout), proficiencies, interaction history and classification (novice or expert) [10]. Information such as proficiencies is derived from potential predictive features during task completion. Data such as total time, dwell time, selection time, number of items visited and average dwell time is used in this derivation [8].

In section 2 C we established that the novice users should be presented with a step by step style UI for the web service while the expert is presented with a more compact user interface for proficient task completion. The user model allows the system to determine the classification of the user, i.e. novice or expert. This information is used as a parameter in the user interface generation process. This is achieved by modifying the OLH and by changing the relationships between UI elements. For example, when the user model establishes that the user is a novice, the OLH is edited to separate complex user interface interaction (e.g. a workflow) into steps on multiple screens, consistent with a step by step UI to aid novices with task completion [8]. On the other hand, when a user is seen as an expert, the OLH is edited to compact the user interface and eliminate unnecessary UI items such as inline help. This process is illustrated in a sequence diagram in Figure 7 for clarification. The UI invokes a web service. The web service WSDL is passed to the OLH along with relevant user information from the user model. XSLT rules are then used to generate the UI. Hierarchy

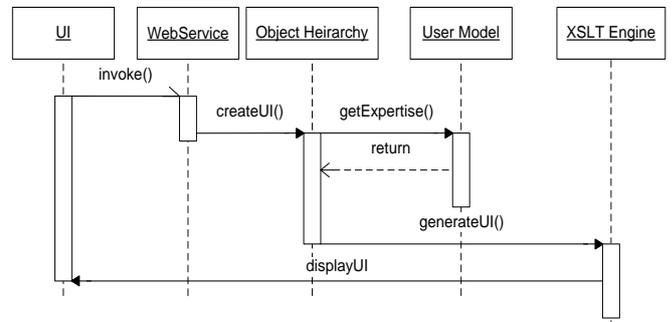


Figure 7: UI creation process

V. TESTING AND EVALUATION

Evaluation of the model requires that the user interface generated be tested for usability, and that user performance be compared to existing UIs of similar applications. The metrics used to establish usability of the generated UI are *effectiveness* and *ease of use*. User performance is measured using task completion, error rate and self-comparison metrics. These metrics are used in comparing the AUI prototype with an existing user interface that makes use of the web services. This establishes if the AUI allows users to complete tasks with better performance compared to the existing UI.

Novice and expert users are evaluated to determine whether the proposed model tracks the level of expertise of users with sufficient accuracy. Known expert users are

tested first, as a pilot study, to determine the range of performance values for experts. Novice user performance will be compared to these threshold values to infer when a novice user's performance can be classified as expert, therefore prompting the AUI to generate the UI suitable for expert users.

VI. CONCLUSION AND FUTURE RESEARCH

Generating user interfaces for web services has clear benefits in reduced development time and development costs for applications that incorporate user-facing web services. The need for extensive change to make services accessible will be reduced. This paper explored the current situation in user expertise, adaptive user interfaces and user interface generation and proposed a model to encompass techniques from each field in order to adapt a user interface for web services based on the inferred level of user expertise and service profile. Different models and methods were analysed and critical components identified. The contribution of the study is an adaptive UI generation model using SOA.

The paper introduced an AUI model, and a proof of concept is currently being implemented. Implementation is followed by user testing and evaluation to determine the usability of the prototype.

ACKNOWLEDGMENT

The author would like to acknowledge the NMMU Telkom/CoE for research funding.

REFERENCES

- [1] Benyon, D. and Murray, D. (1993) Applying User Modeling To Human-Computer interaction design. *Artificial Intelligence Review*, vol. 7, pp. 199-225.
- [2] Chatpar, A. (2008). Increased business agility through BRM systems and SOA. (online) Available at: <http://www.ibm.com/developerworks/architecture/library/ar-brmssoa/> [Accessed: August 2008]
- [3] Connati, (2008) Intelligent User Interfaces. Lecture Notes. Cpsc 422 (online) <http://www.cs.ubc.ca/~conati/422/422-2009World/schedule-422-2009.html> [Accessed August 2008].
- [4] Davies, D. (2006): The Missing Link of SOA. In *Proceedings of International Developer*. pp.71.
- [5] Ellinger, R. S. (2007) Service Oriented Architecture and the User Interface Services: The Challenge of Building User Interface Services Northrop Grumman Information Technology.
- [6] Erl, T. (2005) Service-Oriented Architecture: concepts, technology and design. Prentice hall, Upper Saddle River, NJ, USA.
- [7] He, J., Yen, I., Peng, T., Dong, J. and Bastani, F. (2008) An Adaptive User Interface Generation Framework for Web Services.
- [8] Jason, B., (2008) An Adaptive User Interface Model for Contact Centres. Master's Dissertation. Department of Computer Science and Information Systems. NMMU.
- [9] Kassoff, M., Kato, D. and Mohsin, W. (2003) Creating User Interfaces for Web Services. IEEE Internet Computing. Web Services Track.
- [10] Kules, B. (2000) User Modelling for Adaptive and Adaptable Software Systems (online). Available at: <http://www.otal.umd.edu/UUGuide/wmk/> [Accessed: 19 February 2009].
- [11] LeeSon, A. and Calitz, A.P. (2006) Use of Mobile Technology and Intelligent Interfaces to assist contact Centres and Field technicians.
- [12] Maybury, M.T. (1999) Intelligent user interfaces: an introduction. In *Proceedings of the 4th international Conference on intelligent User interfaces* (Los Angeles, California, United States, January 05 - 08, 1999). IUI '99. ACM Press, New York, NY.
- [13] Netbeans (2009) NetBeans IDE 6.5 Features <http://www.netbeans.org/features/soa/index.html> [Accessed: 15 March 2009].
- [14] OASIS (2006) Reference Model for Software Oriented Architectures. (Online). Available at http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm [Accessed: 15 March 2008].
- [15] Prumper, J., Frese, M., Zapf, D. and Brodbeck, F. C. (1991) Errors in computerized office work: differences between novice and expert users. *SIGCHI Bull.*, vol. 23, pp. 63-66.
- [16] Roch, E. (2007) Gartner on SOA. Available online: <http://it.toolbox.com/blogs/the-soa-blog/gartner-on-soa-16696> [Accessed: 26 April 2009].
- [17] Shen, H.T (2008) INFS 3204/7204. Service-Oriented Architecture. ITEE, UQ. Semester 2, 2008.
- [18] Singh, A. and Wesson, J. (2007) Designing an Intelligent User Interface for Contact Centres. Masters Dissertation. Department of Computer Science and Information Systems. NMMU.
- [19] Song, K., and Lee, K.H. (2007) An Automated Generation of XForms Interfaces for web services. In *Proceedings of 2007 IEEE International Conference on Web Services (ICE2007)*
- [20] Sommerville, I. (2008) *Software Engineering*. 8th Edition. Pearson Education Limited, UK.
- [21] Steele, R. Khankanm K. and Dillon (2005) T.Mobile Web Services Discovery and Invocation through Auto-Generation of Abstract Multimodal Interface. In *Proceedings of ITCC'05*.
- [22] Tsai, W. et al., 2008. Service-Oriented User Interface Modeling and Composition. In *Proceedings of the 2008 IEEE International Conference on e-Business Engineering*. IEEE Computer Society, pp. 21-28. Available at: <http://portal.acm.org/citation.cfm?id=1472207> [Accessed April 20, 2009].
- [23] Wu, J. (2000) Accommodating both Experts and Novices in One Interface (online). Available at: <http://www.otal.umd.edu/UUGuide/jingwu/> [Accessed: 30 August 2008].

Emile Senga received his BCom CS&IS degree at the Nelson Mandela Metropolitan University and his BCom (Hons) CS & IS degree at the Nelson Mandela Metropolitan University. He is currently pursuing a Masters degree in Computer Science and Information Systems at the same institution.