A Framework for the Choreography of Intelligent e-Services

W L Ntshinga
Tshwane University of Technology
Private Bag X680
Pretoria 0001
+27 82 9260390
NtshingaW@tut.ac.za

S O Ojo
Tshwane University of Technology
Private Bag X680
Pretoria 0001
+27 12 382 9280
OjoSO@tut.ac.za

E K Ngassam
SAP Research Meraka & TUT
Private Bag X025
Lynwood Ridge 0040
+27 12 999 9102
Ernest.Ngassam@sap.com

ABSTRACT
In this paper a framework for the choreography of intelligent electronic services (e-services) that follows the Service-Oriented Architecture principle is presented. The composition abilities of intelligent e-services in the framework enable them to match a user’s request for service consumption. The framework is based on a dynamic model of message query-based with casual knowledge-based expansion. A practical illustration of the application of the framework relies on a domain-specific e-service aggregation following an e-mentoring business scenario. The e-mentoring scenario is chosen in order to take advantage of the innovative features offered by the current World Wide Web, which has evolved into a provider of services.

Categories and Subject Descriptors
D.2.12 [Interoperability]: Distributed objects, interface definition languages.

General Terms
Design, Experimentation, Theory.

Keywords
Choreography, service composition, intelligent e-services.

1. INTRODUCTION
Recent studies from industry and academia have focused on service composition for the characterization of appropriate domain-specific existing services since the problem is about how to identify a set of existing services and use them in a collaborative fashion in order to realize a particular business goal [6]. The ability to take existing services, manually and/or automatically, to form new services is known as service composition [7, 32]. There are two forms of service composition which represent two facets of creating business processes. They are orchestration and choreography [3, 4, 15, 20]. The means of describing orchestration and choreography are pivotal in Service-Oriented Computing (SOC) since each has its trade-offs [10, 11].

Orchestration is the outline of interactions that a service agent, usually referred to as the orchestrator, must follow in order to achieve a goal [26]. A good example of orchestration occurs when a supplier decides to have a single point of interaction for the customer and as such, a customer will interact with the orchestrator in order to agree on a purchase order and pay for the goods instead of interacting directly with the supplier. The orchestrator therefore plays the privileged role in terms of understanding the customer request and identifying appropriate means to realize such request. Orchestration focuses on the internal behavior of a business process, therefore the business process and the services that implement the process are separated. The orchestration engine is responsible for executing the process by calling the respective services and determining the order of execution.

Conversely, choreography is more collaborative and addresses the interactions that implement the collaboration among component services. In choreography no service plays a privileged role as each service describes its part in the interaction [11, 23]. Choreography is focused on the external perspective, interested in process interaction, not execution. As a result it is referred to as an interaction model, which is more deployment-oriented [20]. It follows that orchestration can be viewed as projections on choreographies of the point of view of one single participant. Choreographies and orchestrations that explain the same conversation are connected [10].

In real-world scenarios, corporate entities are frequently not prepared to hand over control of their business processes (which are services themselves) to their integration partners [15]. Therefore, there is a need to define jointly and to agree upon the rules of participation within collaborations. Furthermore, there is a need for designers to create robust, highly cohesive business processes whose services and service operations are strongly and genuinely related to one another [15]. Choreography, as a form of service composition, offers a means by which these rules of participation can thus be defined and agreed to by entities. This enables each entity to implement its segment of the choreography as decided by the common or global view.

The research problem identified is that the methods being employed to promote interoperability amongst multiple enterprises do not adequately equip each enterprise/service with the necessary intelligence and ability to comprehend other enterprises/services. Moreover, these methods often do not define semantics well enough, and in many situations they are ambiguous, inconsistent, incomplete, and very difficult to understand the same conversation are connected [10].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SAICSIT ’12, October 01–03 2012, Pretoria, South Africa.
Copyright 2012 ACM 978-1-4503-1308-7/12/10 $15.00.
extend and reuse. As a result, numerous established and emerging standard bodies are formulating conflicting foundations upon which the industry will build [5].

In the quest to provide evidence and developmental explanation of service composition, we construct a framework that will facilitate the choreography of intelligent e-services. The term e-service is used to define an automated enterprise-service using Information and Communication Technology (ICT) to achieve a business goal [17]. E-services are used to develop business applications. Those services often emanate from different providers and SOA principles for the facilitation of interoperability in some circumstances. The main purpose of e-services is to have a collection of network-resident software services retrievable via standardized protocols in order to be integrated into applications [17].

The study supplements the definition provided for e-services by complementing it with an intelligent capability for the purpose of effective and efficient choreography, hence the term “intelligent e-services” or in short IeSs. An IeS is a secured of effective and efficient choreography, hence the term complementing it with an intelligent capability for the purpose of service composition, we construct a framework that will facilitate the choreography of intelligent e-services, while section 4 expands the related work in service composition with overlapping topics of orchestration and choreography. The section discusses some of the frameworks that are available in service composition and their limitations. Section 3 presents a formal framework for the choreography of intelligent e-services, while section 4 expands on the theoretical explanation of the framework. Section 5 presents an e-mentoring business scenario to prove an application of the proposed framework. Section 6 presents the software architecture approach and the prototype implementation. Section 7 lists the benefits and limitations of the prototype. The conclusion and the way forward both appear in section 8.

2. RELATED WORK

McIlraith [13] define automatic web service execution as a process that occurs when “a computer program or agent automatically executes an identified web service”2. McIlraith and Son [12] proposed an agent technology called ConGolog to logically program web services for subsequent execution. In their approach they make use of a broker (agent) to achieve service composition. Therefore, their approach can be viewed as orchestration, and ConGolog procedures can be seen as workflows as they can encode various control flow elements. Workflows describe the plan of activities (sometimes linear, sometimes in parallel) that need to be performed to solve a problem. Flow charts, petri nets, finite state machines or state charts are typical formalisms used by the workflow community to model business protocols [28].

Ponnekanti and Fox [24] proposed a rule-based expert system to automatically determine whether a desired composite service (i.e. joining existing services from multiple service providers) can be realized using existing services. In their approach the developer specifies the input and output of the composite services in the world model and submits it to SWORD. Using a rule engine SWORD determines whether the composite service can be realized using the existing services. However, SWORD does planning only at composition time and not run time.

Sirin et al., [28] proposed a semi-automatic composition of web services that takes the coordination approach, thus choreography, as their composition presents matching services to the user. Their approach or prototype assists the user in the selection of web services during each step in the composition process, filtering possibilities by using semantic description of the services. This process allows more control over the composition process, however, the composition is semi-automatic and therefore end-users without technical knowledge will probably not use it.

Pahl and Zhu [20] also proposed an ontology framework for web services composition that is based on an operational model, and makes use of a process model for orchestration and an interaction model for choreography. Their framework, similar to what we want to achieve, provides techniques for their description, matching, and composition. They formalize composition and interaction in the ontology framework through inference rules. The disadvantage of inference rules is that service composition, in this case choreography, will remain fixed.

Liu et al., [8] are of the view that little has been done to empower the users to participate directly and even dominate service composition. Therefore, user-oriented approach of [8] allows the users to participate in the service composition in an easy-of-use manner (i.e. available Web Service Description Language (WSDL) files and user-annotated tags are used). WSDL describes how to call up a service. It gives information on the data being exchanged, the sequence of messages for an operation of the service and the description of binding [27]. However, WSDL specifications lack semantics which profoundly affect the correctness and completeness of service composition [33]. Hence, Sivashanmugan et al., [30] explored the possibilities of adding semantics to WSDL and Universal Description, Discovery and Integration (UDDI) to achieve sufficient expressiveness to automate the discovery process and composition of services. UDDI is publicly accessible and allows businesses to register information about web services they offer in order for other businesses to discover them using keywords, wildcard characters, categorization (in taxonomies) and qualifiers [19, 29].

The study does not use agents to achieve the composition of IeSs since agent-based reasoning methods are rule-based rather than experience-based [31]. Thakker et al., [31] propose a Case Based Reasoning (CBR) methodology which is suited to build automatic web service composition framework. The CBR is a

---

2 The terms e-service, web service and service are used interchangeably in this paper.
problem solving approach that captures web service execution experiences as cases (old problems) and uses these cases (stored in a database) for finding a solution for new problem. Therefore, their framework, which is closer to our work, is extendible and reusable. However, their framework assumes that the case library contains suitable cases for every possible problem. Moreover, they use an administrator who can be seen as a broker, thus their service composition is that of orchestration.

The framework, discussed in the next section, uses ontology of business activities to discover domain-specific e-services in order to achieve choreography of intelligent e-services without requiring the direct intervention of the human-user.

3. FRAMEWORK

Figure 1 depicts the framework for the choreography of IeSs. The framework, inspired by the works of [20, 31], is based on a dynamic model of message query-based with casual knowledge-based expansion, and the execution of its model follows the two-party choreography business protocol classification of the framework of Mancioipi [10]. Mancioipi [10] classify the business protocols as two dimensions. The first dimension is the number of participants involved in the conversations (i.e. it can be two-party or multi-party). The second dimension is the perspective adopted to describe the structure of the conversation (i.e. it can be orchestration or choreography). The study limits to two-party conversation because majority of conversations occur between two participants.

In this study the dynamic model is used to express and model the behavior of e-services over time. Therefore, an IeS is capable of acting as a service requestor or a service provider depending on the business problem being addressed. The framework supports technology neutrality (i.e. protocols, description and the discovery mechanism complying with widely accepted standards), loosely coupled e-services, location transparency (e.g. services have location information stored in a repository).

The framework consists of component subsystems. A component describes interfaces through which it can be composed with other components [34]. Each component depicts one or more interfaces through which messages are sent to and received (i.e. bidirectional) from a potential collaboration mate component.

The framework has the service request component that triggers a request based on well-defined specification of the query required by the user as well as the service goal. The service request is sent to the service discovery component that receives the request and matches the service request against domain-specific syntactic rules (specification) and data semantic rules (ontology) (the assumption is that the user would want a new service to satisfy a specific need, thus action to be performed).

The service discovery component, which is goal-based, uses the composer’s specifications (which take the form of high level description of the desired composition), the registry (e.g. UDDI, Cdye, Seekda), the repository and the internet at large to query, find and list the services that are related to the goal of the service request. The approach follows, partly, the style of Sivashanmugan et al., [30] who are of the view that having both the description and query clearly declared, the results will be more relevant than keyword matching based information retrieval. In case the request is either syntactically or semantically incorrect, an exception will be thrown by the framework exception handler component not shown in the framework.

The service discovery component starts by discovering existing services and applications which can contribute to the composition, a business protocol known as two-party orchestration. Different service communities/providers may have a service. The use of service communities speeds up the process of discovering relevant component services [14].

The list of services is then forwarded to the finding candidate e-service component. This component selects the ideal e-services. For effective browsing and searching of the candidate e-services (i.e. the ability to discover domain-specific e-services), an ontology of business activities has to be used at the reasoning level. This is due to the fact that finding the correct e-services would have been easier if registries were categorized based on domains in such a way that each registry could maintain only the e-services pertaining to that particular domain [29].

Once these two stages have been completed, the composition path (i.e. building composite e-services from found paths) is thus reached, forming what is referred to as service composition, represented as the choreography/orchestration engine in Figure 1. The composition paths, which can be several composition paths in the same business protocol, are stored in the local Extensible Markup Language (XML)3 repository and can be used once again by the service discovery component.

Figure 1. Framework for the choreography of intelligent e-services.

The newly composed and coordinated e-services with data semantic case representation format are stored as an experience (i.e. user case) into the repository and can be reused to complete the candidate e-service stage which is very difficult to identify prior to this. The IeSs are therefore formed once the appropriate e-services that match the service request are identified and ordered according to their invocation sequence available in the composition path. The IeSs can then be deployed (i.e. to the end-user since the end-user expects a running IeSs) for execution in order to achieve the desired business goal. The next section describes the framework in detail.

4. THEORETICAL EXPLANATION OF THE FRAMEWORK

Let us provide a theoretical explanation of the framework in order to capture a generic choreographic setting that can be applied in any domain where service composition is of interest. Figure 2 depicts an example of a choreography scene for IeSs triggered by a service request (eSreq) in order to achieve a business goal. This process can be viewed as a service request

3 XML is used because it can run over the internet and it simplifies the exchange of business data between companies as it provides a cross platform support.
component on the framework triggering a specification of a goal that the user wants the service to achieve. This is the initialization of the choreography.

eSreq dispatches the message to the service broker (eSb) as a request of service provisioning. eSreq has to follow the business rules governing the message exchange in the conversation. Therefore a request is sent as a 5-tuple of the form (d, c, x, t, u) following the model depicted by the Web Service Choreography Description Language (WS-CDL) working group in describing the three levels of choreography abstraction, which are, (d) the description of the data types (atomic or complex) used – defining the rules governing message exchange (e.g. the data types of the parameters sent by an e-service must be compatible with the data types of the parameters received by its partner [14]); (c) the conditions under which a given message is sent – is it send message or a receive message?; (x) the description of the physical structure of the information that can be exchanged between the components; (t) the technologies used – that is protocol (protocol make explicit the relationship between messages (methods) supported by the application [34]; and (u) the destination Uniform Resource Locators (URLs) – the web address where the service can be found [2].

The message is then received by the broker (eSb) who has the responsibility to interpret it syntactically and semantically. The syntactic matching engine from the framework contains rules for enabling the acceptance of a specific query by the requestor. If there is no match, an appropriate exception is thrown for further refinement of the query by the requestor. Similarly, the semantic matching of the query by the broker relies on the ontology component of the framework to ascertain that the query being made is part of the various solutions paths available in existing registries. A mismatch in this situation means that the requested query cannot be resolved semantically, which translates to the fact that the problem domain has no cognisance of such a query.

After positively analyzing the request semantically and syntactically, the broker (eSb) is now in the position to find from the various registries possible LeSs that match the criteria of the requestor. eSb is responsible for executing the process by calling the respective services and determining what steps to complete. This demonstrates two-party orchestration business protocol, namely (me, other) or (other, me) [10]. eSb uses semantic description of the service. This is the co-ordination approach, proposed by Sirin et al., [28] (see related work section).

**Figure 2. e-Services in the Choreography Process.**

We assume that for a given problem, three services have been successfully identified (eSb, eSf, eSd) such that their sequential invocation will realize the desired goal of the requestor. eSb, eSf, and eSd could be existing services which can contribute to the composition (i.e. the two-party orchestration demonstrated by “finding candidate e-services” in the framework). We elaborate more on the composition path in section 6. The next section explains a business scenario that is used for the framework.

5. BUSINESS SCENARIO

E-Mentoring is one of the facets of mentoring and it started to gain popularity in the 1990s when teachers realized that electronic mails (e-mails) might be an easier and better way to connect with learners [9]. E-Mentoring is the process where a mentee and a mentor communicate via information and communication technologies in order to solve certain problems [18]. A mentor is able and willing to support, inspire, and transfer knowledge and skills to a mentee. A mentee possesses a certain skills set, attitude and aptitude and has the passion to learn and gain knowledge from a mentor. It therefore means a mentor provides mentoring services to a mentee and the mentee requires the mentoring services.

In order to provide a more uniform mentoring experience and to closely monitor the performance of the mentor, it is desirable to have a single point of interaction with the mentee, hence the orchestrator, referred to in Figure 2 as the Mentoring Service Broker (MSB).

Figure 3 depicts the e-mentoring sequence diagram which provides a sequential map of the message passing between objects over time. The sequence, which is based on SOA elements and interaction, proposes the sharing of information and/or interconnected set of mentoring services, each accessible through standard interfaces and messaging protocols. The figure portrays the two stage process of e-service composition, meaning finding sought after candidate e-services from the repository as a starting point and building composite e-services from found paths. This is staged by the finding candidate’s e-services component in the framework for the choreography of LeSs.

The services that a mentor can provide are published to a MSB. The MSB handles interactions related to mentoring requests, rendering of the mentoring requests and other value-added services. As defined in the introduction this stage is referred to as the orchestration and the MSB can be viewed as an orchestration engine. The MSB has access to a repository for service specifications. The MSB offers searching and matching tools to facilitate efficient finding of mentoring services. The Mentee can request the MSB for the most suitable services during the execution time. All interactions between the services in the composition are channeled through the MSB. The MSB can add, edit, delete, and search for mentors, mentees, and services. There can be a need for an e-service that is not currently defined in the system or known to the MSB. In this case, the MSB discovers the service dynamically, namely, by adding and removing the e-services as the need arises at runtime.

Furthermore, the mentee’s request can be broken down into sub requests. These requests can then be made in sub-requests to access the repository and communicate with the mentor via open Application Programming Interfaces (APIs), and registry. This process allows the formation of new services by composing existing services for the mentee’s request. The MSB then presents a proposal to a Mentee. This proposal to the Mentee is a newly composed e-service and it is further stored in the repository as an experience (i.e. user case). These stored services can be reused within the system to complete other tasks. During this process a choreographer supervises the behavior of services in order for the proposal to be presented. In other words, the choreographer communicates inputs to the appropriate component services and transmits the component outputs either to other components or the mentee or the mentor. The choreographer keeps the history of sequences (i.e. set of inputs and outputs).
6. PROTOTYPE IMPLEMENTATION

The implementation approach, as well as the design approach, discussed in the previous section, followed a top-down approach for demonstrating the choreography of intelligent e-services. This approach provides a conceptually clearer composite of e-services that are easier to verify and maintain. The top-down approach means that a user specifies an abstract description of a desired composite service. A simple example of such an abstract description could read, “I want details of a mentor who has a career in distributed systems and e-commerce who resides in Johannesburg, South Africa”.

The overall implementation is achieved as the composition of peer-to-peer interactions among the collaborating e-services (i.e. choreography). The design approach that was taken is that of loose coupling, which means that the study promotes flexibility of web services and reduces the interdependencies across collaborating e-services.

The framework for the choreography of IeSs was implemented using Microsoft Visual Studio 2010 Professional Integrated Development Environment (IDE) on the client-side and Java Enterprise Edition (J2EE), together with Jersey platform, on the server side. Figure 4 portrays the software architecture for the choreography of IeSs. Simple Object Access Protocol (SOAP) web services, which form part of the ASP.NET component and also form part of the current web, were used to demonstrate the proof of concept of the study. SOAP is an XML-based mechanism for exchanging structured data among network applications [22].

Representational State Transfer (REST) web services were implemented in J2EE and deployed in the Linux server (onto the Apache Tomcat Application Server) which uses Jersey for handling service requests and responses. REST web services were used because they are simple and are suitable for basic ad hoc integration scenarios. Table I conceptually and technologically compares the two integration technologies of web services.

The Linux server uses Jersey for handling service requests and responses. J2EE acts as a middleware and standardizes the interaction with many low-level system services. The Linux server was selected because it is a free operating system and it provides stability, security and flexibility advantages.

In order to prove the feasibility of the proposed framework for the choreography of IeSs, the following steps were followed. Firstly, web services were designed, developed and accessed to demonstrate the two-party orchestration model and the two-party choreography model. Web services were accessed using ASP.NET and Java Enterprise Edition tools. Secondly, a database was designed and created in MySQL.

Table 1. SOAP Web Services vs. RESTful Web Services

<table>
<thead>
<tr>
<th>SOAP Web Services</th>
<th>RESTful Web Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOAP web services are more adaptable and they address advanced quality of services and are ideal for professional</td>
<td>REST is well suited for basic, ad hoc integration scenarios and gives flexibility to optimize the performance of</td>
</tr>
</tbody>
</table>

The third party web services, usually provided by the internet, telecommunications, and third-party providers situated in a central location were used to prove the feasibility of the proposed framework for the choreography of IeSs. The providers have opened their e-services to the public in the form of open APIs, a trend following the Web 2.0 paradigm. This is shown with the DataStore Cloud block in Figure 3. Different vendors provide a number of services, however the difference are mainly about nonfunctional properties [8]. Non-functional
properties are properties that are used for evaluating services, such as service costs, response time, throughput, security, availability, and so forth [15].

One of the vendors used, as part of the implementation, is the UDDI. UDDI was chosen for several reasons. For example, (i) UDDI goes beyond the DISCO by defining how to interact with a full-fledged web services information repository; (ii) UDDI uses standard-based technologies such as Internet Protocol Suite (TCP/IP), HyperText Transfer Protocol (HTTP), Extensible Markup Language (XML) and Simple Object Access Protocol (SOAP) to create a uniform service description format and service discovery protocol; and (iii) UDDI is a logical centralized, physically distributed service with several root nodes that can reproduce data with each other on a regular basis.

Even though UDDI simplifies interoperability, it is less effective as it does not use information in the service descriptions during discovery and searches for a single service rather than a composition of services. Furthermore, UDDI provides weak search facilities since it only allows a keyword based search of business [21]. Finally, the UDDI lacks semantics in the discovery process [30].

On our e-mentoring portal, the user (for the purpose of this paper we will look at one conversation – that of a mentee to a mentor) requests a mentorship service from the MSB. Using Figure 1 as a choreography scenario, this is demonstrated with the service request communicating with service discovery, a business protocol known as two-party choreography (i.e. two parties, a mentor and a mentee). Behind the scenes, the MSB uses a business protocol known as two-party orchestration by using the IP2Geo web service from Cyde to resolve the IP address to network owner name, city, province, and country. In other words, the MSB is the service caller and it calls IP2Geo. Therefore, the MSB is coupled with IP2Geo as it knows what the service is and how it is called as well as what it can accomplish. However, the opposite is true for IP2Geo and the other services (i.e. Mentor List Service (MLS) and Distance Service (DS)).

The IP2Geo web service was used for three main reasons. Firstly, it identifies a geographic location of a user. Secondly, it is used to prove the use of business registry to locate information about functional specification of e-services and use the imported e-services without the knowledge of their implementation (demonstrated in Figure 1 as Specification and Registry blocks respectively) and thirdly, its functions is to find companies or institutions with a given type of e-service or a composition of e-services that address the functional need (demonstrated in Figure 1 as Finding Candidate e-Services block).

The study also tested the same concept defined in the aforementioned paragraph using a REST web service named GEOIP. The GEOIP web service, which required low-level J2EE coding, was deployed on a server and it was called by SOAP web service named MSB (a process that links two services and use the services in a collaborative fashion). In turn, the MSB passes the IP address by calling the GEOIP service to demonstrate the out-in pattern referred to as a “request/response” operation which demonstrates choreography. MLS was deployed on a Linux Server and it searches the database for all the mentors found in a particular country or city. This is demonstrated in Figure 5.

Figure 5. Towards Choreography of iE-Ss.

The MSB calls another REST web service called DS to get the distance between the mentor and the mentee by passing the IP address, country and city. A process that can be viewed as finding candidate e-service and composition path as represented in figure 1. The IP address is used to determine the location of the mentee in a particular city within a particular country; hence the three fields are passed (i.e. IP Address, Country and City). The DS then calculates the distance using the latitude and longitude of the mentor. During these message exchanges, it has to be noted that each message exchange is a new message exchange to each web service. This means that each web service does not keep the state of a conversation in which it is involved. Table 2 lists the examples of e-services, specifications of the e-services, and the functionalities of the e-services.

<table>
<thead>
<tr>
<th>e-Service</th>
<th>Specification</th>
<th>Functionality</th>
<th>Client</th>
</tr>
</thead>
<tbody>
<tr>
<td>eSMSB</td>
<td>{GetCountry, GetCity, GetMentorDetails}</td>
<td>demonstrates how to discover e-services exposed on a given server</td>
<td>eSGS, eSMS, eSDS</td>
</tr>
<tr>
<td>eMLS</td>
<td>{ConnectToDatabase, GetMentorDetails}</td>
<td>searches the database for all the mentors found in a particular country or city</td>
<td>eMSB</td>
</tr>
<tr>
<td>eDS</td>
<td>{LocationMentee, LocationMentor}</td>
<td>to get the distance between the mentor and the mentee</td>
<td>eMSB</td>
</tr>
</tbody>
</table>

Table 2. Example of e-services in the e-mentoring system

Therefore the composition path is as follows:

Search ➔ Find ➔ Present
{ eSMSB, eMLS } ➔ { eSMS, eDS } ➔ { eMSB, eDS }

MySQL open source database was used to demonstrate the concept of storing and retrieving mentor information. MySQL is a database server for data storage and management. A table named e-Mentor formed part of the database and had fields such as identification number, name, surname, qualification, career profile, company, city, country, latitude, work phone and e-mail address. These fields are presented to the mentee as “Mentor Details”. The mentee can then choose from the shortlist the mentor services and the MSB then contacts the mentor by presenting the Mentee’s details. The mentor will then
acknowledge the request by either accepting or declining the request.

7. BENEFITS AND LIMITATION OF THE PROTOTYPE

To our knowledge the proposed e-mentoring prototype has never been done to date. The theory of integrating different technologies was proven when the MSB executed RESTful web services which were developed using the J2EE and Jersey platform. This means that any platform that uses web services can plug into the MSB, meaning that the service providers also have an option to describe and publish their services. Therefore, it was easy for the MSB to make use of services offered by the registries promoting loose coupling.

The concept of storing and retrieving mentor information using MySQL open source database can be replaced by any database management and/or Application Service Providers (ASPs) in proving the integration of different technologies. Any organization or company that has a database of mentors can easily plug into the prototype. The prototype can be used to create opportunities for mentors and mentees to meet and conduct business electronically, hence the term e-mentoring. This means that a mentee will spend less time looking for a mentor as done by means of phone or e-mail or visiting a mentor.

The limitation of the prototype is that it does not allow dynamic change of partners and services as the processes are built on the interface description of a particular service. The prototype also relies heavily on the deployment of standards such as SOAP, REST, WSDL, UDDI, Cynde and Seekda.

8. CONCLUSION AND WAY FORWARD

An understanding of service composition is needed in order to allow larger software systems to be assembled based on services as the basic unit. In this paper we demonstrated this theory by presenting a framework and associated techniques to generate the choreography of intelligent e-services. The framework makes use of e-mentoring business processes in order to demonstrate the choreography of intelligent e-services.

Although the chosen business scenario is e-mentoring, the framework is extensible and can be used in any business domain. The e-mentoring prototype can increase the mentoring efficiency and it can be an appropriate solution for Higher Learning Institutions (HLI), whereby the application can be used to create opportunities for mentors and mentees to meet and conduct business electronically, hence the term e-mentoring.

The limitation of the prototype is that it does not allow dynamic change of partners and services as the processes are built on the interface description of a particular service. The prototype also relies heavily on the deployment of standards such as SOAP, REST, WSDL, UDDI, Cynde and Seekda.

The main contribution of this work is a framework that supports the composition of IEs. The framework complies with widely accepted standards, specifically using a two-party choreography model. Our software architecture is geared to take maximum advantage of the currently available platforms.

There is still much space for further improvement though. We have developed the e-mentoring ontology using Protégé Version 4 by means of Web Ontology Language (OWL) to describe and structure the data of mentoring and choreography ontology. Ontologies will be used in the prototype to provide for the automation of different tasks of service composition (i.e. reuse and maintenance) and the choreography ontology thereof to achieve service composition. This includes the definition of concepts, relations, and instances that are used in representing the e-mentoring domain.

9. ACKNOWLEDGEMENTS

The support of Tshwane University of Technology and SAP Research Mobile Empowerment Africa/Meraka UTD towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at are solely those of the authors and should not necessarily be attributed to the Tshwane University of Technology and SAP Research Mobile Empowerment Africa/Meraka UTD.

10. REFERENCES


