Semantic Web Data Description and Discovery

Michael Ryan Bannon
Department. of Electrical & Computer Engineering
University of Waterloo
Waterloo, ON, N2L 3G1,
Canada
mrbannon@swen.uwaterloo.ca,

Kostas Kontogiannis
Department. of Electronics & Computer Engineering
Technical University of Crete
Chania, 73100
Greece
kkontog@softnet.tuc.gr

Abstract

Currently we are experiencing the emergence of the fourth generation of the World Wide Web which is geared towards service and data provision using semantic and ontological information. Specifically, the objective is for data available on the web to be described, retrieved, and used using semantic and contextual information. This paper presents a framework that allows such a polymorphic service provision through the introduction of user personas, semantic data descriptions that extend the WSDL and the UDDI protocols. In this way, web data content is associated at run-time with different services and presentation manifests, according to the context and the environment it is invoked and used in. The framework and its associated architecture have been implemented in a prototype system that utilizes Web Services technology.

1. Introduction

The World Wide Web has evolved from a collection of interlinked static pages to a dynamic environment whereby services and data can be registered, discovered, selected and invoked in a seamless fashion. The latest evolutionary pattern that is taking shape in the area of Web technologies is the use of semantically enriched information that allows for customizable service and data provision in Web enabled environments. Over the past couple of years we are experiencing the emergence of new publish/subscribe protocols for Web Services. These include the Web Services Description Language (WSDL) [17], [19], the Universal Description Discovery, and Invocation (UDDI) [18], and the Business Process Enactment Language for Web Services (BPEL4WS), to name a few. However, very little work has been reported on the use of ontological information that would assist, first to associate contextual information with the offered services, and second to dynamically bind data with services according to the environment they are invoked and used in.

The most current related work in semantic data modeling has been conducted within the framework of DAML+OIL, DAML-S [25] and the Web Ontology Language (OWL) [4]. Each of these approaches uses RDF [3], a subject-verb-object relation language, at its core.

Our research addresses the Semantics level of Information System integration. As Web Services provide the potential to revolutionize the world of service integration, the promise of wrapping legacy systems with standardized, source-independent, data-centric interfaces makes the IT community to invest significant research efforts towards this objective. This data centric service-based platform is however, still fairly immature and a strategy for successful context aware Web Data and Service deployment has yet to be fully realized. As a step towards the definition of a framework for the transparent deployment and integration of software applications, this paper presents a reference architecture that supports robust enterprise level system integration using web services technology. In particular, we propose a framework by which data can be considered as a form of an extended Web Service. Specifically, we present an extension to the WSDL to specify contextual and run time properties of Web data. We refer to this extension as the Web Data Description Language (WDDL), which allows for the definition of a finite set of semantic data schemas.

This language also compliments a semantic data description repository, that we refer to as the Universal Data Description Discovery and Integration (U3DI) server, based on the Universal Description Discovery and Integration (UDDI) initiative. The purpose of the U3DI server is to store semantic and functional information for a Data-As-Service offering the distribution of semantic data as a service. The above concepts are supported by a metamodel that denotes the necessary entities and relations to realize such a context and persona-aware data and service provision. The rest of the paper is organized as follows. Section 2 presents related work. Section 3 presents the overall system architecture, while Section 4 discusses the WDDL, and U3DI models. Section 5 illustrates experimental results obtained from a prototype implementation of the system. Finally Section 6 concludes the paper and presents some future work.

2. Related Work

Much of the work presented in this paper is centered on the concept of the Semantic Web [1]. The Semantic Web is an evolution of the World Wide Web that is given additional markup using ontological technologies for the purpose of allowing web data and services to be "understood" by machines. The rest of this section presents technologies and research that relates with the concept of the Semantic Web and personalized web spaces

The Simple HTML Ontology Extensions language, or SHOE, is one of the earliest technologies developed for the semantic web, developed by the Parallel Understanding System Group in the Department of Computer Science at the University of Maryland [2]. SHOE draws from an ongoing line of research based on knowledge representation and ontologies to allow semantic markup within existing HTML documents. The Resource Description Framework, or RDF [3], is an XML language that allows an author to describe metadata in a relational manner. RDF offers a subject/predicate/object formalism for defining a relationship by relating a subject to a value, or object, via a provided predicate. RDF is the basis for work done in DAML+OIL and OWL. Similarly, the DARPA Annotated Markup Language, or DAML, is an extension of RDF that allows RDF relationships to be associated with semantic ontologies. The Web Ontology Language, or OWL, is yet another realization of a language for implementing the semantic web [4].

In the area of Software Engineering, Laddad describes the concept of aspect-oriented programming (AOP) in [5]. Aspect-oriented programming allows orthogonal objects - each representing a mutually independent goal -to integrate with each other in order to satisfy a larger, more complete goal. AOP is considered relevant to this work in reference to the use of roles and role models. Role models define usage patterns among particular objects; roles are those patterns that are singluar instances of the given model. The use of AOP in role models has been investigated in [6]. Moran and Dourish describe the idea of context-aware computing in [7]. Essentially, context-awareness is an abstract notion of awareness with respect to physical and social situations. This overlaps with the idea of contexts in this paper.

Hong proposes an infrastructure and language for representing contexts in [8]. In it, he offers the *logical context datamodel* as a type of semantic net with *aggregates* that relate multiple entities together as contexts. This relates with the concept of a Persona presented later in this paper.

Theodorakis et al. present a language for specifying a context within an information base [10]. Their work, based on [9] and [11], describes a language framework that denotes a context as a conceptual entity that may contain other contexts or simple objects (i.e. those objects that are not contexts). Complementary to the above an important aspect of recent web technology is the ability for systems to be automatically reactive. That is, given some scenario, a system must have the capacity to take actions based on the conditions of that scenario relative to a predetermined set of rules.

Paton and Diaz offer a survey of active database system research [12]. Event-condition-action (ECA) rules are an important addition to the idea of polymorphic data within an active database. A large amount of work has been done regarding event notification. Bailey et al. offer an ECA language based on XML in [13]. They further go on to analyze the behaviour of such a language, stating that not enough work has been done to satisfy behavioural analysis.

Kiyomitsu et al. propose ECA rules for web personalization in [14]. Specifically, they use XML-based ECA rules to personalize web page content, taking into consideration the logical and physical location of the user, user history relative to the page, and known behaviour of the user. This adds another dimension to reactive web: not only is the system reactive to the data, but also to the profile of the individual accessing the data.

Bonifati et al. offer an argument for the use of XML-based technology to "push" active rules to existing XML repositories [15].

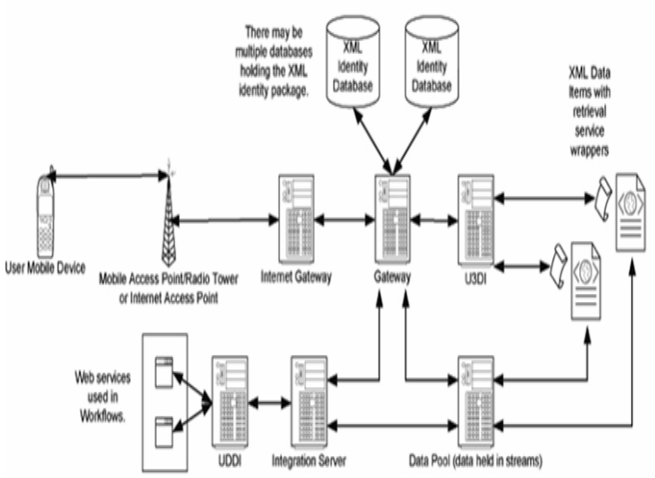

Figure 1. High level system architecture

Finally, there is a wealth of emerging technologies that can be used to support the implementation and deployment of the related work presented above. These include Jena, a toolkit developed by the HP Labs Semantic Web Programme [26], SOAP, UDDI, WSDL, and BPEL4WS [16], [24], [25]. In the following section we present the proposed system architecture and its supporting meta-model.

3. System Architecture

The architecture of the system is comprised of several components. These include the User Mobile Device (UMD), the Gateway, the XML Database (XML-DB), the Universal Data Description Discovery and Integration (U3DI) Repository, the Data Pool, the Data-As-Services (DAS) module, the Integration Server, and the Universal Description Discovery and Integration (UDDI) Service Registry. An overview of the architecture with these components can be viewed in Figure 1.

3.1 Major System Components

User Mobile Device (UMD)
The UMD represents any type of device that can interact with the Gateway and also represents an individual user or a client process in the system. In fact, the UMD does not necessarily have to be mobile as its name suggests. The UMD denotes any client device or process whose purpose is to allow interaction with the system.

Gateway
The Gateway acts as the entry point into the system. The responsibility of the Gateway is to manage the UMD's session on the system. The Gateway is also responsible for hosting Persona composition duties, interacting with the XML-DB, routing HTTP requests between the UMD and the Integration Server and instructing the system to enact Personas. To accomplish its responsibilities, the Gateway has direct connectivity with the Data Pool, Integration Server, XML-DB, U3DI, and UMD.

XML-DB
The purpose of the XML-DB is to store Personas and Persona components in their XML form. A Persona is a collection of properties that denote the context in which a service or data are presented to the client. The XML-DB receives requests from the Gateway and responds with XML encoded descriptions of the corresponding Personas and Persona components that were requested. The XML-DB is also responsible for compiling full Personas on request for use by the Gateway.

U3DI
The U3DI repository's purpose is to encode information related to Data Items and their associated data sources. The U3DI repository receives requests for Data Items from the Gateway and searches for the most appropriate Data Item given the requirements from the Gateway's request. The requirements used for localizing the appropriate Data Item consist of ontological references, descriptions of the data's format, and parameters regarding the retrieval of the data. These factors along with the proposed Web Data Description Language (WDDL) and the U3DI specification are presented in detail in Section 4.3. As its name suggests, the U3DI repository is based on the UDDI specification. However, it may be extended to allow for DNS search capabilities so that the appropriate data source may be found on another U3DI server as described by [20]. Similarly WDDL is proposed as an extension of the WSDL and will be discussed in more detail in Section 4.2.

DataPool
The Data Pool is meant to store and organize all data streams that have been published to the U3DI repository and are meant for use by the Integration Server. It is the responsibility of the Data Pool to notify the Integration Server, in the form of ECA events, of any arrival or change in a data stream. The Data Pool also implements algorithms to support garbage collection of Data Items that are no longer needed.

Integration Server
The Integration Server is responsible for integrating Data Items (from the Data Pool) and the Roles that interpret those Data Items depending on the user's Persona. When a Persona is enacted, the Gateway forwards the Data Item

requests to the U3DI repository and the Roles (encompassing workflows) to the Integration Server. On

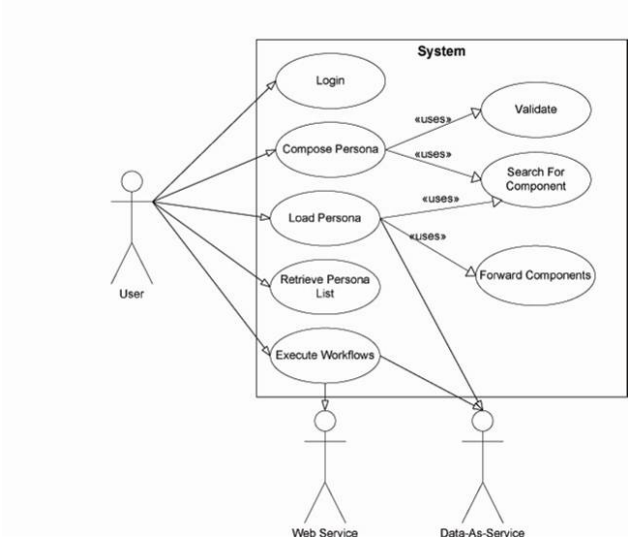

Figure 2. High level Use Cases

receiving the Workflows, the Integration Server will await those events specified by the loaded Workflows. It is also the responsibility of the Integration Server to interact with the web services in question and act as a middle-agent between those services and the UMD for requests/response events between the two. In addition, the Integration Server is similar to the Gateway in that it must check the integrity of the Workflows it has been sent; the integrity of a Workflow means that there must exist knowledge of runtime web services on an available UDDI repository that match the ServiceClasses specified in the Workflows.

UDDI Repository

The UDDI repository is acting as a binder that holds handlers of the available Web services. Much like the U3DI repository, the UDDI repository uses DNS-like algorithms to find desired web services if they are not located at the default UDDI repository [20], [23].

3.2. System Operation and Use Cases

This section conveys the logical system functionality through UML Use Cases. The high-level view of the actors and Use Cases is illustrated in Figure 2. The actors in the Use Cases are the User, Web Service, and Data-As-Service (DAS). Each provides functionality outside the domain of the system however, the DAS is discussed in the following sections. Furthermore, the message sequencing chart for the high level component interaction is illustrated in Figure 3.

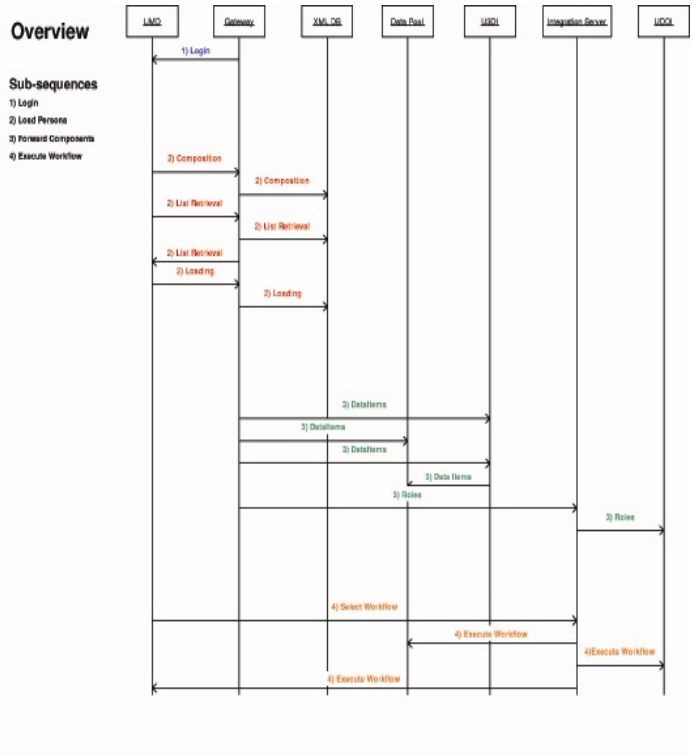

Figure 3. Message Sequencing Overview

4. System Models

This section presents the metamodel that describes the concept of a Persona for context aware computing, and the WDDL and U3DI specification models. A Persona is an abstract construct that allows a user to associate in a polymorphic manner online data with different services or "points of view". The WDDL model extends the WSDL specification in the Web Data domain, while the U3DI extends the UDDI specification to act as a broker for Web Data content as well.

4.1. Meta Model for Context Aware Data

The user's perception of the web depends on the Contexts that are associated with a Persona. A Context is the abstract definition of what data is to be visible to a user and how that data is to behave. Technically, a Context is a container, however, the nature of a Context that defines data and its behaviour is implicit by containing Data Items and Roles. In a nutshell, a Persona is part of an individual's profile that defines a point of view relative to the web. A Persona denotes (a) what data is visible to the user, (b) how that data is semantically interpreted, and (c)

what actions are to occur given the particular values of the data.

Persona
Aside from containing information regarding a user's profile on the system, the Persona describes that user's data preferences and how he/she wishes to realize them. As described earlier, a Persona defines a "point of view" relative to the web. This is accomplished by defining a set of Contexts within a Persona.

Context
A Context is an entity that relates semantically described web data to behaviours that exist as web services. Both data and associated behaviours are described using compatible semantic constructs such that a behaviour defines what data it may operate with, by using the same semantic descriptors used for the data. Consequently, that data may behave differently under Context A than in Context B. A semantically described piece of data is called a DataItem and the entity that relates behaviour to DataItems is called a Behaviour. Behaviours are implicitly contained in Contexts through Roles.

DataItem
A DataItem is an abstract semantic description of data that the user wishes to view or use. DataItems are described behaviourally as instances of specific by DataClasses. Specifications of DataClasses are denoted using the U3DI model that will be discussed in the next section.. DataItems have attributes that obtain single, atomic data values. These attributes are originally defined in the corresponding DataClasses and are related to a DataOntology. However, a DataItem obtains ist Attributes relative to the DataClass that is an instance of.

Role
A Role entity represents a Behaviour set to be associated with DataItems. If a DataItem represents data in a Context, a Role represents how that data is to be used. that is how a Persona should generally act and what actions to take given the data presented to it.
A Behaviour is an entity that describes action data associated with it is to take depending on its value. Behaviours are implemented as Workflows and described by ServiceClasses, obtained from a ServiceOntology, that semantically define what actions can be taken. Behaviours make up the actions that constitute a Role.

Workflows
An ECA (Event Condition Action) entity represents an ECA based workflow. Each ECA entity is composed of multiple rules that describe the logic of the ECA entity and is associated with a runtime that represents the grounding of that ECA workflow. Each workflow that is used in a Role must be associated with those DataItems that it uses.

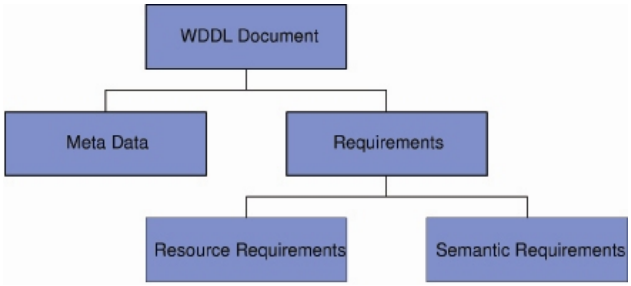

Figure 4. Overview of WDDL data requirement contents.

This ensures that those Workflows brought into an Context will be compatible with the DataItems in the same Context.
A runtime entity is the service grounding for an ECA entity. In effect, it represents the actual workflow that is executable on the web. Every runtime has an associated WSDL file that describes how that workflow can be enacted.

4.2. WDDL Specification

The concept of DataItems as presented above is realized by the Web Data Description Language (WDDL). The purpose of the WDDL, which has been based on an amalgamation of concepts in semantic data and database theory, is to allow a user to quantify the requirements of a DataItem in a semantic and structural fashion based on existing OWL ontologies. When a particular DAS has been found that satisfies the requirements of the provided WDDL document and that DAS has been queried with the WDDL document, the DAS returns an instance of the data that conforms to the requirements of the WDDL. The goal of the WDDL is to enable a user to expressively describe the data he/she requires from a DAS by using semantic constructs and descriptive data requirements. As stated earlier, semantic description languages have been proposed in the form of such languages as DAML and OWL, just to name a few. WDDL stems from the concept that data can be described semantically and includes the notion of requirements on data values, the schematic structure of the data, and conditional statements relative to the data. In addition to using semantic networks to accomplish these goals, the concept of AND/OR trees is used to implement conditional requirements of OWL *classes*. Additional work relative to data requirements can be found in relational databases and soft-goal graphs [21]. Figure 4 illustrates the general contents of a WDDL semantic data requirement document. A WDDL

document is made up of two pieces: a set of metadata and the actual requirements, which is broken up into resource requirements and semantic requirements. The meta-data is used to provide information such as a description of the file, who authored the file, and so forth.

Resource Requirements

As stated earlier, a user may define the requirements of the DAS and associated data sources that would be used to process the WDDL document and instantiate the resulting schema, respectively. Such constructs are made available in the *resourceRequirementElement*. This element contains predicate calculus statements that use predicates of four terms to define the requirements of the DAS or the data sources.
The calculus used in the *resourceRequirement* structure is loosely based on the work done with ConChat [22]. A first-order predicate *Resource* is defined with four terms:

Resource(<EntityType>,<Subject>,<Relater>,<Object>)

A *Resource* predicate relates a <*Subject*> to an <*Object*> with a <*Relater*> relative to a specific <*EntityType*>.A <*Subject*> is any type of quality parameter, whether it refers to data quality, service quality, etc. A <*Relater*> is any comparison operator and is used to relate the <*Subject*> to the <*Object*>.The <*Subject*> and <*Object*> must be of the same type and the <*Relater*> must support that type. The <*EntityType*> is one of *Localization*or *Selection*, meaning that the predicate relation should be applied to determining a DAS or choosing the appropriate data sources, respectively.
The source for the <*Subject*> term comes from a predetermined set of quality attributes. This predetermined set may come from any source as long as the WDDL document attributes will be understood and mapped to DAS entities. It is suggested that, in addition to a DAS publishing OWL *ontologies* that it uses in its data descriptions, it also publishes a set of quality attributes useful for resource descriptions. Some examples of Resource predicates are security, latency, differential, reliability, and accuracy.

Semantic Resources

The semantic requirements of a WDDL data requirement file are implicitly represented in a tree structure. The structure is analogous to that of an XML schema. In essence, a user creates a tree that defines a semantic schema that he/she wishes to receive instances of. The nodes of the tree are represented as classReferenceElements. Each classReference logically represents an OWL *class* and may contain zero or more classReferenceElements as associating entities, thus forming the schematic network structure. In this respect,

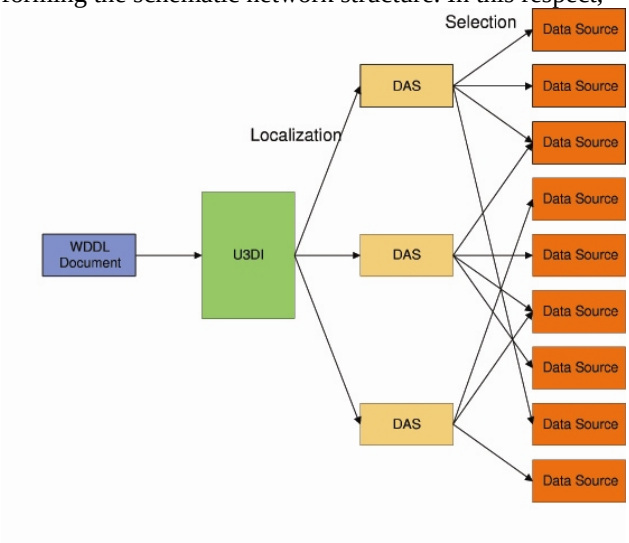

Figure 5. U3DI scope and role.

containment means that the XML data instance of a classReferenceElement will have, as sub-elements, XML data instances of those classReferenceElements that are associated of the parent classReference. For example, a classReference may represent a Car *class* and has an association with an Engine *class*. Thus, XML instances of the Car *class* will have, as a subelement, an XML instance of the Engine *class*. The XML instances are XML representations of OWL individuals. To give more descriptive power to the user, WDDL incorporates AND/OR trees into its definition by allowing disjunctions. WDDL uses AND/OR trees to allow classReferenceElements the ability to define containment with Boolean expressions.

So far, WDDL has focused on defining the semantic requirements of conceptual entities by using OWL *ontologies*. The conceptual schematic structure will be of little value if there is no simple data associated with it. In this respect, simple data refers to data based on simple types such as strings or integers.

Generally, a client would author a WDDL document that uses static *dataProperties* in its dataRestrictionElements and uses the IncludeAttribute with seperate *dataProperties* that are expected to be dynamic. This way, the client will receive streams of data that may be constantly changing.

4.3. U3DI Specification

U3DI is based on the Universal Description Discovery and Integration (UDDI) initiative.

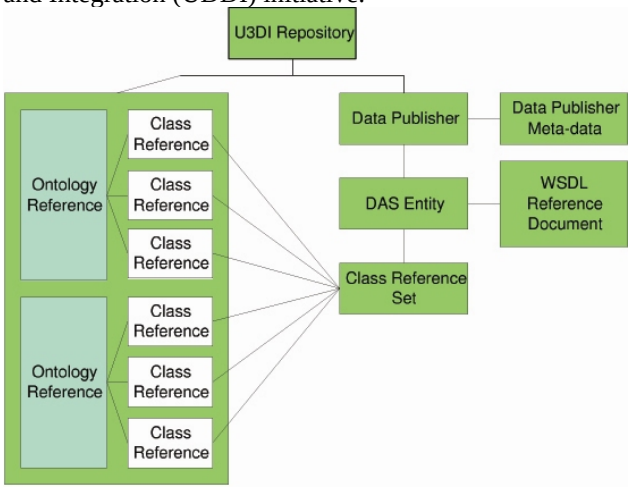

Figure 6. U3DI data structure overview.

UDDI is a repository server that acts as a “white pages” for web services. A user or automated process can query the server for the location and link to a full description of a web service based on parameters relating to how that service might be accessed and what service it offers, among other parameters. Our work on U3DI has adopted the concept of UDDI and altered it to serve specifcally as a white papers for DAS entities.
Figure 5 illustrates the role of U3DI relative to a DAS. The U3DI is used to locate DAS entities that may conform to the WDDL document semantic and resource requirements. The U3DI cannot determine whether a DAS is able to provide an instance of the particular WDDL document. Rather, like UDDI, it acts as a general guide for discovering DAS entities.

Similarly, Figure 6 illustrates an abstract view of the U3DI data structures. To summarize, a U3DI contains any number of DAS publishers. Each DAS publisher references any number of DAS entity references that logically represent existence of a DAS entity. A DAS entity reference, in turn, references any number of OWL classes.
As stated earlier, a U3DI stores information on any number of DAS Publishers. A DAS Publisher is represented in XML as the dasPublisherElement.

Each dasEntityElement has a unique dasEntityKeyAttribute that identifies that DAS entity. In addition, the dasEntity has a dasPublisherKey that uniquely identifies the owning DAS Publisher. Each dasEntity has a name and optional description as well.

The nature of a DASEntity is represented in the classReferenceElement. This element contains availableClassElements that list what OWL *classes* are relative *dataProperties* are implementable by this particular DAS entity by using the classKeyattribute and dataPropertyKeyElement. Like the dasPublisherElement, a dasEntityElement contains descriptive information in the form of the dasEntityName and dasEntityDescriptionElements. In addition, there may be any number of qualityAttributeElements that contains quality attributes of the particular DAS Entity and its associated data sources. The dasEntityElement also contains a wsdlReferenceDocumentElement that points to a WSDL document on the web. The purpose of this document is to describe how the relative DAS Entity can be used.
Every U3DI repository should contain a database of OWL *ontology* references. For each *ontology* reference, there is a set of OWL *class* references. Each *class* reference should point to an existing DAS Entity on the repository. This implies that the target DAS Entity can provide instances of the *class* that references that DAS Entity.

Furthermore, each existing classReferenceelement logically represents an OWL *class* that is implemented by one or more of the DAS entities represented at the U3DI repository. Each classReference is uniquely identified by a classReferenceKey. It also contains a classNameElement that defines the name of the *class* and one or more dataPropertyReferenceElements. The dataProperty ReferenceElements act as references to the available *dataProperties* that the DAS Publisher claims are available for instantiation. These *dataProperties* must exist within the OWL definition of the relative OWL *class*.

A dataPropertyReferenceElement contains the name of the OWL *dataProperty* it represents in the form of the dataPropertyNameElement and a unique identifier key in the form of the dataPropertyKeyAttribute.

A dasPublisherElement contains metadata in the form of the dasPublisherName, dasPublisherDescription, and dasPublisherContactInformationElements. Each DAS Publisher must have a dasPublisherKeyattribute that uniquely identifies it from other DAS Publishers on the repository. A DAS Publisher also contains multiple references to unique DAS Entities using the dasEntityKeyElement. It is understood that a DAS Entity referenced by a DAS Publisher is the responsibility of that DAS Publisher. Furthermore, a DAS Entity may be the responsibility of only one DAS Publisher.

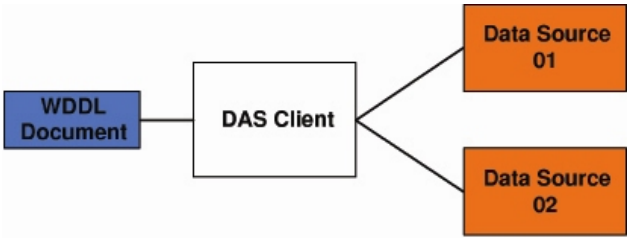

Figure 7. Summary of test components structure.

5. Prototype System

The prototype system is related to the provision of medical information and associated polymorphic services on this data to various users with diverse personas and profiles (epidemiologists, ambulance crews, family doctors, ER doctors). This section presents two sets of statistics that were computed on a working implementation of an individual DAS. The first statistic was captured under experimentation and provided time statistics regarding WDDL parsing and data retrieval. The second statistic descirbes space allocation required relative to the DAS specifications.
The implementation that was conducted was limited to a single DAS registered with two data sources. A WDDL file was constructed such that one data source would not be able to satisfy the WDDL file, but the other would. Figure 7 provides an overview of the components involved. Each data source component in the figure represents a data source entry in the DAS that may be capable of implementing the given WDDL document.

The DAS itself was limited to extending its WDDL parsing by involving only subclasses/subproperties and equivalent classes/properties, in addition to searching for the original entity. The DAS parsed the provided WDDL file and locally examined if a known data source could implement the said file. Once parsing was completed and a schema was generated, the DAS examined two resource requirements to decide if a data source was available.

5.1 TimeAllocation

Four iterations were executed with the same WDDL file and ten different WDDL files were created, each having one root and X number of children. The number of children were between ten and 100 inclusive, each being a factor of ten. Furthermore, all of the children were contained in a conjunction, so each child had to be satisfiable. For each trial, an average time required to

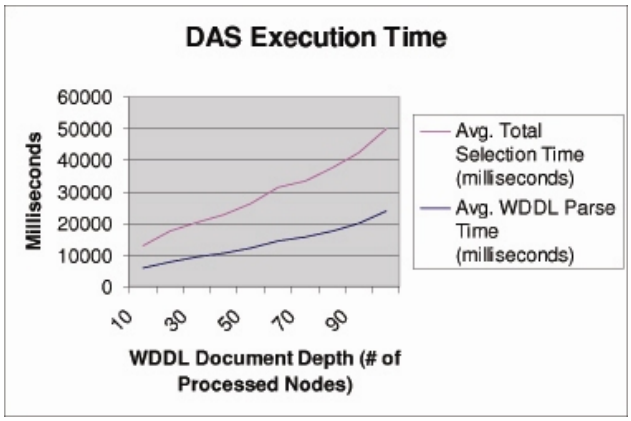

Figure 8. Average WDDL parse and DAS retrieval time.

parse the WDDL document and the total time required to execute the implementation were computed. Figure 8 illustrates the time performance results for the experimentation suite.

The parse time is the time taken to traverse the WDDL document and decide what classes and data properties can be satisfied by the known data sources. In the WDDL files, each of the children are identical. As such, a linear growth of time based on the number of nodes was expected and found. In fact, traversal of the tree is always linear with respect to the number of nodes in the tree since any pre-order traversal with a constant-time node visit is linear. Of course, node visits are not always constant. Rather, they depend on the classes or data properties that are available to the DAS. In a worst-case scenario, a given node would be compared to all nodes in a given ontology before returning positive (or negative). Thus, a node visit in the algorithm is linear also.

5.2 Space Allocation

The second statistic gathered concerns the size of a DAS specification. In effect, how much space is required to store the information needed to decide whether a provided WDDL is implementable or not.
In a DAS specification, the only dynamic entity is the storage entity used to archive known data sources and what they provide. In total, a DAS must store knowledge of each data source it is associated with, including information on how to access that data source, an identifier for each unique ontology class provided by the DAS, an identifier for each unique data property provided by the DAS, an identifier for each unique object property provided by the DAS, and an identifier for each unique resource requirement defined by the data sources. In addition, each data source must reference those resource requirements that it can satisfy, along with a value, and

DAS space requirement entities	
Requirement	Symbol
Associated Data Source	D
Ontology Class	C
Ontology Data Property	P
Ontology Object Property	O
Resource Requirement	R

Table 1. DAS space requirement entities.

must reference those ontology entities that it can provide. Table 1 above summarizes these requirements. For simplicity, assume that all identifiers, references, and values have the same storage requirements, X. The space required to store all identifiers is linear among the five different identifiers.

$$SpaceReq.=X(D+C+P+O+R)$$

In a worst-case scenario, each data source is able to provide information regarding each class, property, data property, and resource requirement. For simplicity, let X also be the space required to store a reference from a data source entity to one of the other four entities. The new space requirement becomes:

$$SpaceReq.=X(D+C+P+O+R)+X(DC+DP+DO+DR)$$

Therefore, space requirement in a DAS grows relative to $max(DC,DP,DO,DR)$.

We are currently working on extending and deploying the prototype with multiple Data-as-Service elements, and more complex application domains.

6. Conclusion

This paper proposes a framework whereby emerging semantic and web service technology can come together to address the problem of customizability and personalization in web services and web data. It has focused on two primary issues. First, the Persona, a concept that allows an individual to define a unique web space based on the semantically-described needs of the individual has been presented.

Second, to facilitate the idea of personalized data, the Data-As-Service and Web Data Description Language was created. The Data-As-Service provides semantic markup for raw data based on an individual's specific needs for the data. The individual defines his/her semantic needs using the Web Data Description Language. Data-as-Service elements can be published in a U3DI binder that extends the UDDI standard.

The work presented in this paper opens some opportunities for future work, both in the realm of research and implementation technologies.
Research wise, more work can be done regarding the definition of a Context. It was noted that several, mildly disjoint definitions of context in computer science have been offered in literature. It may be beneficial to define the concept of a Persona and Context in a well-defined language (possibly an XML language) that could be standardized (such as how SOAP and WSDL have become relative standards). More work should also be done on the DAS architecture and Data Pool. In conjunction with the DAS and Data Pool, additional work is required to solidify a strong storage requirement for Personas and related components. However, most importantly, work should be done to integrate workflows into the Persona, combining the idea of template/concrete workflows with the concept of Behaviours. This would allow testing of the proposed communication protocol within the architecture.

Overall, we believe that this is an emerging area of research that holds the potential of interesting applications and industrial uses.
Concluding, we would like to acknowledge and thank Bell Canada University Laboratories at the University of Waterloo, the Consortium for Software Engineering Research, and the Natural Sciences and Engineering Research Council of Canada for their continuous support for this research.

References

[1] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. Scientific American, 284(5):35–42, May 2001.

[2] Jeff Heflin, James Hendler, and Sean Luke. SHOE: A Knowledge Representation Language for Internet Applicateions. Institue for Advanced Computer Studies, University of Maryland, College Park, http://www.cs.umd.edu/projects/plus/SHOE/pubs/techrpt99.pdf/, 1999.

[3] World Wide Web Consortium. Resource Description Framework. World Wide Web, http://www.w3.org/RDF/.

[4] World Wide Web Consortium. Web Ontology Language. World Wide Web, http://www.w3.org/2001/sw/.

[5] Ramnivas Laddad. I want my AOP!, Part 1. Java World, http://www.javaworld.com/javaworld/jw-01-2002/jw-0118-aspect.html/ , 2002.

[6] Elizabeth A. Kendall. Role Model Designs and Implementations with Aspect-oriented Programming. In Proceedings of the 1999 ACMSIGPLAN conference on Object-oriented programming, systems, languages, and applications, pages 353– 369. ACM, November 1999.

[7] Thomas P. Moran and Paul Dourish. Introduction to This Special Issue on Context-Aware Computing. Human-Computer Interaction, 16:1–8, 2001.
[8] Jason I. Hong. The Context Fabric: An Infrastructure for Context-Aware Computing. In ProceedingsofCHI2002, pages 554–555. ACM, April 2002.
[9] Manos Theodorakis, Anastasia Analyti, Panos Constantopoulos, and Nikos Spyratos. Context in Information Bases. In Proceedings of the Third IFCIS Conference on Cooperative Information Systems. IEEE, August 1998.
[10] John Mylopoulos and Renate Motschnig-Pitrik. Partitioning Information Bases with Contexts. In Proceedings of the Third International Conference on Cooperative Information Systems, pages 1–12, May 1995.
[11] M. Theodorakis and P. Constantopoulos. Context-Based Naming in Information Bases. International Journal of Cooperative Information Systems, 6(3, 4), 1997.
[12] Norman W. Paton and Oscar D iaz. Active Database Systems. ACM Computing Surveys, 31(1):63–103, March 1999.
[13] James Bailey, Alexandra Poulovassilis, and Peter T. Wood. An Event-Condition-Action Language for XML. In WWW2002, pages 7–11. ACM, May 2002.
[14] Hidenari Kiyomitsu, Atsunori Takeuchi, and Katsumi Tanaka. ActiveWeb: XML-Based Active Rules for Web View Derivations and Access Control. In Proceedings of the Workshop on Information Technology for Virtual Enterprises, pages 31–39. IEEE, 2001.
[15] Angela Bonifati, Stefano Ceri, and Stefano Paraboschi Pushing Reactive Services to XML Repositories using Active Rules. In WWW10, pages 1–5. ACM, May 2001.
[16] World Wide Web Consortium. Simple Object Access Protocol. World Wide Web, http://www.w3c.org/2000/xp/Group/.
[17] World Wide Web Consortium. Web Services Description Language (WSDL) 1.1. World Wide Web, http://www.w3.org/TR/wsdl.
[18] Universal Description Discovery and Integration. World Wide Web, http://www.uddi.org/specification.html/.
[19] Peter Brittenham, Francisco Cubera, Dave Ehnebuske, and Steve Graham. Understanding WSDL in a UDDI registry. World Wide Web, ftp://www6.software.ibm.com/software/developer/library/ws-wsdl.pdf/, 2001.
[20] Kelvin H. Cheung. A Customizable Web Services Integration Environment. Mas-ter's thesis, University of Waterloo, Ontario, Canada, 2002.
[21] John Mylopoulos, Lawrence Chung, and Brian Nixon. Representing and Using Nonfunctional Requirements: A Process-Oriented Approach. *IEEE Transactions On Software Engineering*, 18(6):483–497, June 1992.
[22] Anand Ranganathan, Roy H. Campbell, Arathi Ravi, and Anupama Mahajan. ConChat: A Context-Aware Chat Program. *Pervasive Computing*, 1(3):51–57, July-September 2002.
[23] Gerald Tarcisius. A Specification and Enactment Framework for Context-Aware Template-Based Workflows. Master's thesis, University of Waterloo, Ontario, Canada, 2002.
[24] Tony Andrews, Francisco Curbera, Hitesh Dholakia, Yaron Goland, Johannes Klein, Frank Leymann, Kevin Liu, Dieter Roller, Doug Smith, Satish Thatte, Ivana Trickovic, and Sanjiva Weerawarana. Business Process Execution lan guage for Web Services Version 1.1. World Wide Web, http://www.siebel.com/bpel/.
[25] DAML-S: Semantic Markup for Web Services. World Wide Web, http://www.daml.org/services/.
[26] HP Labs.Jena A Semantic Web Framework for Java. World Wide Web, http://jena.sourceforge.net/.