SEMANTIC DESCRIPTION OF MULTIMEDIA CONTENT ADAPTATION WEB SERVICES

By
SURAFEL LEMMA

A THESIS SUBMITTED TO
THE SCHOOL OF GRADUATE STUDIES OF ADDIS ABABA UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF SCIENCE
IN COMPUTER SCIENCE

July 2005
ADDIS ABABA

Acknowledgments

First of all, I would like to extend my deepest gratitude and respect to Dr. Dawit Bekele for his invaluable comments and guidance devoting much of his precious time. Had it not been for his support, this work may not reach this stage.

My special thanks also goes to Ato Girma Berhe, Dr. Vasil-Marian Scuturici and Dr. Mulugeta Libsie for providing me with helpful documents and constructive comments and suggestions. I am also grateful to Ato Fekade Getahun who gave me documents that are useful to my work.

Finally, I would like to express my sincere thanks to my family especially Meheret for being there during all my ups and downs. They have been of great help to me throughout my education.

Table of contents

List of tables

List of figures

List of appendices

Abstract

In pervasive computing, multimedia information is accessed using different devices with different processing power, storage size, display size, etc. Hence, the information that needs to be displayed has to be modified in order to meet the devices' requirements. This modification is called content adaptation.

Multimedia content adaptation can be done at various places in different frameworks. Some frameworks propose that content adaptation should be done by content servers, while others consider that it should be done at the client side. Another approach considers that content adaptation should be performed by third party services that can be implemented using web services. The objective of this research is to find ways of facilitating discovery of these third party content adaptation web services by users of these services.

In order to use these web services, one has to be able to find them on the Web. However, today, finding the right web service for a specific task (multimedia content adaptation in our case) is difficult. This is mainly because of the type of description used to advertise the web services.

Current standards that are used to describe web services are syntactic. Syntactic description of web services contains some keywords and signature of the web service, which is later, used for searching the services. However, this information does not allow high rate of success during service discovery. These limitations of the current standards can be improved by using semantic description methods. Semantic description gives the meanings and relationships of the terms and concepts used in describing a web service, which improves the discovery of a web service.

In order to allow semantic description of multimedia content adaptation web services, we have developed an ontology for multimedia content adaptation web services. To develop this ontology, we first studied different ontology development methods based on which we formulated an approach using which we have developed the ontology by identifying different types of audio content adaptation web services and their properties that can be used to increase the semantic information of the audio content adaptation web services.

In this research, we have also developed a prototype to show the usability of the ontology that we developed. The prototype allows a user to advertise a web service using the ontology and look for a specific web service.

Keywords: Web service description, ontology, web services, audio content adaptation

1 Introduction

Pervasive computing, also called ubiquitous computing, is a technological evolution heading to allow computation from anywhere and anytime on various computing devices. This technology is also sometimes referred as the third wave of computing (many computers per person). The idea of this evolution is first conceived in Xerox PARK laboratory by Mark Weiser in 1988 [12].

In pervasive computing, the electronic devices, which are used to access information, vary from small devices like PDAs to much more powerful computational devices such as personal computers. When the information that is going to be accessed is sent to a given person, it should be presented in such a way that it can be displayed using the device she/he currently uses. Changing this information so that it adapts to the device of the user is called content adaptation.

Content adaptations are done at one of the following three places in most of the existing adaptation frameworks: by the client machine at arrival, by the server before it is sent, or by the proxy on the way. A newly proposed approach to adapt the content of the information is using third party adaptation services. In this approach, the adaptation services can be implemented using web services [4].

In frameworks that use the new approach, if a client receives a content that needs to be adapted, it searches for a web service that can perform the specified content adaptation in a web service registry. To search for web services the client uses the advertised (published) description of the web services.

Currently, the most common web service description is the syntactic description. The service is described by giving its signature (name, parameter, data types, etc.) and keywords that identify it. This way of describing services has some major shortcomings. In particular, it requires knowledge of the terminology that is used to describe the services in order to search and find it. For example, the service may not be found just because the right keywords are not used for the search. This limitation can be alleviated by using semantic descriptions. This is because they describe what the web services do using standard

semantics description system, which gives a better success rate for the searches. Hence, they are increasingly considered as better alternatives [15].

The objective of this research is to develop an adequate service description for multimedia content adaptation web services that enhances lookup of these services. We have preferred to use semantic description method that we consider more efficient in this context.

1.1 Motivation

In a pervasive environment, the devices which are used to view multimedia information have different display sizes and different capabilities such as processing power, storage size, etc. The bandwidth of the network that connects the devices also varies greatly. Therefore, in order to access multimedia information using these devices, the content has to be modified in order to meet the device's requirement(s) and the limitations of the network. For example, if a user wants to see an image compressed using JPEG (Joint Picture Experts Group) with a device that is capable of displaying only images compressed using GIF (Graphics Interchange Format), the image has to be modified so as to meet the device's requirement. Similarly, if a user wants to read in Amharic a document written in English, the content has to be converted according to the user's requirement. If a user also wants to hear a radio broadcast over the Internet and is in an area where the bandwidth of the network is very small, then the audio has to be adapted so as to meet the networks requirement. These can be achieved using adaptation techniques which can be implemented using web services.

Web services are well-defined, reusable software components that perform specific, encapsulated tasks via standardized web-oriented mechanisms [17]. The provider of the web service has to make it available on the web and publish (advertise) it so that users can find it. To advertise a web service, different frameworks such as the service-based distributed content adaptation framework described in [4, 5] use UDDI (Universal Description, Discovery and Integration) or UDDI like registries.

These registries however have limitations in service discovery. They allow lookup of services only using keywords such as provider name, service name, or service category. In

addition, they use direct match to identify web services [22, 24]. Hence, these make searching a service difficult as the user has to remember names or keywords of all web services advertised and choose the right one.

Web service interfaces are described using WSDL (Web Service Description Language) which is a W3C[1] recommended language [30]. The description contains information about the operation supported by the service, the kind of data or message being communicated, binding, etc. An example of a WSDL description of a web service that translates a text to equivalent speech is presented in appendix 1[2].

Using elements of WSDL, it is possible to identify what type of messages or data types are exchanged by the web service, how it is technically implemented i.e. how it is bind to a specific communication protocols like SOAP (Simple Object Access Protocol), and where it is located. However, this information is not adequate for automatic web service discovery and composition [5]. It requires semantic information like type of the web service, the meaning of the input/output, how much it costs, or its execution time etc. Due to these, web service discovery and composition requires human intervention.

For example, the text to speech translator web service's WSDL description given in appendix 1, shows that the service provides two operations, *TextToSpeechFromURL* and *TextToSpeechFromText*. The input parameters for the *TextToSpeechFromURL* are *fileURL*, *outPutFormat* and *language*, and all are of type string as shown in figure 1. The description also tells that the service can use both HTTP (Hyper Text Transfer Protocol) and SOAP for communication and is located at http://localhost/TextToSpeech/Service.asmx. But it does not tell anything about the execution time, cost of the service, etc. Moreover, this description does not tell anything to software agents or other services about the parameters except that they are of type string which can be used to contain information (see figure 1). For software agents or other services the names of the parameters does not give any

[1] W3C (World Wide Web Consortium) is a group that develops interoperable technologies (specifications, guidelines, software, and tools). http://www.w3.org/

[2] This web service is developed by Girma B. et al. for the demonstration of the infrastructure presented in [4, 5].

meaning as they may do to humans. So these limit the use of WSDL to describe web services.

Figure 1 Part of the WSDL description of text to speech translation web service

```
<message name="TextToSpeechFromURLHttpPostIn">
  <part name="fileURL" type="s:string" />
  <part name="outputFormat" type="s:string" />
  <part name="language" type="s:string" />
</message>
```

In this research, we try to address the above problems by developing an ontology for semantic description of multimedia content adaptation web services. This ontology will serve as a means to annotate and share the semantic information, such as the service performed by the web service, its inputs/outputs, etc.

1.2 Statement of the problem

Current standards that are used to describe web services use syntactic description. Syntactic description does not tell the meanings and relationships of the terms and concepts used in describing a web service. As a result, different clients can interpret differently the same concept described using syntactic description. Due to this, web service lookup and interoperability have become a difficult task.

Hence, in this research we propose a method to describe semantically multimedia content adaptation web services. For this purpose we develop an ontology that enables the description of multimedia content adaptation web services in a standard way.

1.3 Objective

The objective of this research is to facilitate and increase efficiency of multimedia content adaptation web services lookup by enriching the information used to describe these adaptation web services. To attain this, we will develop an ontology for semantic description of multimedia content adaptation web services which is more powerful than syntactic description. We will then demonstrate the usability of our new description methodology.

1.4 Scope

Multimedia content adaptation web services incorporate image, audio, video, etc content adaptation web services. Given the limited time available for this work, developing ontology for all multimedia content adaptation web services is not possible. Therefore, in this research, we will focus on developing an ontology that is used to semantically describe audio content adaptation web services. Nevertheless, as there is similarity between the different types of multimedia content adaptation web services, the ontology developed for audio content adaptation web services can easily be adapted to other types of multimedia content adaptation web services.

1.5 Methodology

In order to achieve the objective of this research, we started by studying existing web service description methods and looking at various ontology development methodologies. Then appropriate methodology is formulated. Using the formulated methodology, we developed an ontology for semantic description of multimedia content adaptation web services. Finally, the usability of the ontology has been demonstrated using a prototype.

1.6 Organization of the report

This paper is organized in six sections. The first section introduces the paper. The second section provides information on what ontology is, its uses, and languages that are used to represent ontology. The next section describes web services and description methods that are used to describe them. The fourth section presents the proposed ontology and the methodology used to develop the proposed ontology. The implementation which demonstrates the usability of the proposed ontology is then presented in section five. Finally, section six presents the conclusion and future work.

2 Ontology

Today different software products, people, organizations and departments within organizations need to exchange information. Unfortunately, since the same concept may be represented using different terminology or jargons the communication is not always without difficulty. This lack of common understanding has created the following problems: [27]

- Poor communication within and between organizations and between people
- Difficulty in identifying requirements and defining a specification of a system
- Limited interoperability between systems and software tools
- Limited re-use and sharing which in turn results in duplication of efforts

To alleviate these problems, having a common understanding of the concepts or information and their relationship is important. This can be achieved using ontologies.

Ontology as stated in [27] is "the term used to refer to the shared understanding of some domain of interest which may be used as a unifying framework". It is also defined as "an explicit specification of conceptualization" in [6]. Ontology has different components: description of concepts, which are called classes, in a domain, their properties, which are used to describe the concepts and restrictions on the properties. These concepts are expressed explicitly with varying degree of formality [28]:

- *Highly informal:* expressed loosely in natural language
- *Structured informal:* expressed in a restricted and structured form of natural language
- *Semi-formal:* expressed in an artificially formally defined language
- *Rigorously formal:* meticulously defined terms with formal semantics, theorems and proofs of such properties as soundness and completeness

This varying formality with the purpose, intended use of the ontology, and subject matter (i.e. the nature of the subject matter that the ontology is characterized), give rise to different kinds of ontologies.

2.1 Uses of ontology

The use of an ontology can be classified into three main broad categories: communication, interoperability and system engineering.

Communication

Ontology is a means to have shared understanding of concepts in a domain. Hence by using ontology people having different backgrounds and working environments can have common understanding of the concepts. This improves communication by integrating different user perspectives, reducing ambiguity and increasing consistency among the communicating parties.

Interoperability

In order for two different systems to interoperate, there has to be a mechanism for translation between the different languages, modeling methods, and software tools used. This can be achieved by developing an ontology, which serves as an interchange format.

System engineering

Shared understanding of the concepts of a domain helps to have a common view of the problems of a domain and a specification of a system. This increases reusability of concepts and reliability of a system.

2.2 Ontology representation languages

Today, there are different ontology specification languages with varying degree of expressiveness, like RDF (Resource Description Framework), DAML (DARPA Agent Markup Language), DAML+OIL, and OWL (Web Ontology Markup Language). These languages vary form one another in their power to express semantics and relationships between concepts in an ontology. Some of the most important are described below.

2.2.1 RDF (Resource Description Framework)

RDF is a general purpose language mainly used for representing metadata, which is a structured data about data, on the Web [13]. RDF improves discovery and access to globally distributed information, by providing a structure for unambiguous expression of semantics. This structure provides consistent encoding, exchange and machine processing of standardized metadata.

RDF provides a means for extending, reusing and/or publishing both human-readable and machine-processeable vocabularies, ontologies, which are set of properties or metadata elements. These properties or metadata elements are used to describe a resource defined as an object and uniquely identified using a Uniform Resource Identifier (URI).

RDF is known for its simplicity. It is a way to express and process a series of simple assertion statements like "This web service is developed by Girma Berhe". Such statements in RDF have three structural parts: a subject ("This web service") which has to be a resource that is accessible through the Web, a predicate ("is developed by") and an object ("Girma Berhe"), which can be a literal or a resource. Due to this the statements are called triplets.

2.2.2 RDF-S (Resource Description Framework Schema)

RDF-S is an extension of RDF. It includes a larger vocabulary with more complex semantic constraints than RDF [1].

RDF properties represent relationships between resources. However, it does not provide a mechanism to describe these properties and relationships between properties and other resources. This limitation of RDF is improved in RDF-S. It specifies classes, and domain and range of properties that are used to describe the relationships between properties and other resource.

2.2.3 DAML (DARPA Agent Markup Language)

DAML is a language developed by DARPA, Defense Advanced Research Projects Agency, with the objective of having a common ontology language that would facilitate semantic interoperability of various projects which are working on the semantic web [7].

DAML is designed based on and in RDF/RDF-S. DAML, in addition to the features provided by RDF-S, offers different standards such as equivalence of different concepts in different languages (e.g. “child of” in English and “enfantDe” in French) and uniqueness of a particular property (e.g. Identification number) [21]. It also implements a simple infrastructure that allows a machine to make simple inferences.

2.2.4 DAML + OIL (DAML + Ontology Inference Layer)

DAML + OIL is a result of the collaborative effort of the developers of DAML and OIL. It combines language elements of both DAML and OIL [3]. OIL is a Web oriented ontology language developed by a group that is mainly dominated by Europeans. The group has similar objective as that of DAML.

DAML was weak in its semantic specification, and it could result in disagreements on the precise meanings of terms in an ontology [7]. DAML + OIL, however, is designed to solve this problem. It provides a rich set of constructs which are based on description logics, frame based representations and Web based languages. These constructs, thus, can be used to build ontology that is understandable by machines and humans.

2.2.5 OWL (Web Ontology Language)

OWL is a World Wide Consortium (W3C) general purpose recommendation language for defining Web ontologies [2]. This language incorporates additional language elements and the lessons learned from the design and application of DAML + OIL.

OWL, like DAML + OIL, uses description logic to provide an unambiguous meaning of terms defined using it. This can be used by applications that require interoperability. It is also compatible with XML and RDF.

3 Description of web services

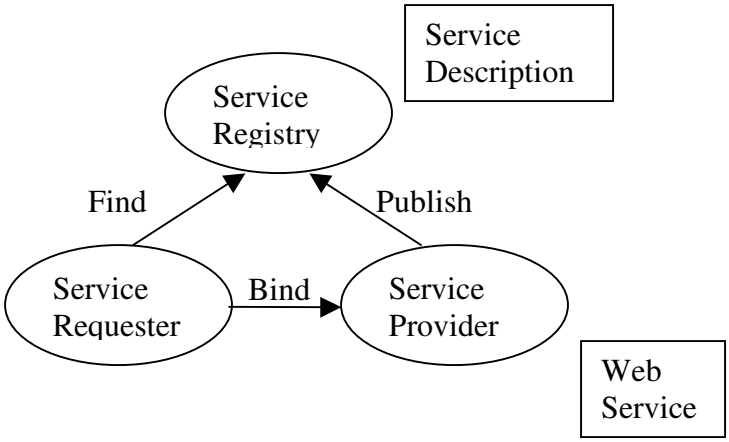
3.1 Web services

Web services have a number of definitions. One definition found in [17] is: "Web services are well-defined, reusable software components that perform specific, encapsulated tasks via standardized web-oriented mechanisms."

The main driving factors for the development of web services are the need for sharing of information and interoperability. These are achieved through a set of well-defined standards that define syntax, communication protocol and invocation signatures. In addition to the above, web services promise to minimize web application development time, cost, maintenance and increase reusability of code [17].

The use of web services involve three activities as indicated in figure 2.

Figure 2 Web service usage scenario



Publish (Advertise): with this activity the service provider will post the description of the web service's functionality and information of its provider onto the service registry, a repository of profile information.

Find (Discover): is an activity where a service requester, which can be a developer or another process, browses or uses a query to find information, such as port on which the required service is found, from the service registry.

Bind (Invoke): The service requester invokes the discovered service.

While developing an application, web service approach provides a number of advantages over other approaches such as object oriented approach. Web services are loosely coupled. Therefore, this allows easy modification of components of an application. They are very easy to integrate, because, they are based on widely accepted and open standards, such as SOAP and WSDL. They are also easily accessible, as they are distributed over the Internet.

3.1.1 Main components of web services

The main reason with which interoperability is achieved in web services is the use of standard protocols and/or technologies. These components are depicted in figure 3.

Figure 3 Web service components

XML (eXtensible Markup Language)

XML is a W3C recommended mark up language that is used to describe data. It allows developers to create their own customized tags. This gives flexibility and helps to easily interpret the meanings of the tags. This language is used to define the web services' protocols.

SOAP (Simple Object Access Protocol)

SOAP is a W3C recommended XML-data transport protocol, used to transfer data, payload, from the source to the destination over the Internet using HTTP (Hyper Text Transfer Protocol). It describes the content of the message, how to process it, and offers a transport binding for exchanging message.

WSDL (Web Services Description Language)

WSDL is W3C's recommended language for describing web service interface. The description contains information about the operation supported by the service, the kind of data or message being communicated, the data types, binding and port. (Described in detail in the next section).

UDDI (Universal Description, Discovery and Integration)

UDDI is an industry initiated web service registry where service providers advertise their services and service requesters look for a specific service. It can be seen as the yellow pages, pointing to registered web services located elsewhere [15]. It provides interface to publish and search for web services. The search engine takes keywords such as provider name, service name, services category, or tModel (technical Model, which refers to standard technical specifications such as WSDL or to abstract specifications of taxonomic schemas such as NAICS[3]) to identify business web services and the services they provide.

The searching, however, is not efficient as it does not consider the relationships, like sub-category of a category, between entities in its directory [22, 24]. Moreover, it bases its search on high level information like industrial service type categorization of business and services [22, 26]. That is it does not allow specifying specific capabilities and input/outputs of a service.

3.2 WSDL (Web Service Description Language)

WSDL is an XML (eXtensible Markup Language) based language used for describing web services [30]. It is currently a W3C standard for describing web services interface.

A WSDL file contains information about the operations supported by the web service, the kind of data or message being communicated, the data types, binding – protocol and data format specification for a particular port type, and port. In order to present these information WSDL uses the following elements.

- *Types:* defines the data types used by the web service for message exchange.

[3] The North American Industry Classification System (NAICS) published by US Census

- *Messages:* are abstract definitions of the data elements of an operation. Each message is composed of one or more parts, which represent a typed parameter. The parts can be compared to the parameters of a function call in a traditional programming language.

- *Operation:* This element is used to group messages. There are four types of operations.

    - *One-way:* The operation can receive a message but will not return a response.

    - *Request-response:* The operation can receive a request and will return a response.

    - *Solicit-response:* The operation can send a request and will wait for a response.

    - *Notification:* The operation can send a message but will not wait for a response.

- *Port type:* defines an abstract set of operations provided by the web service. The port type can be compared to a class in object oriented programming and the operations to a method.

- *Binding:* defines the message format and protocol details for messages and operations defined by a particular port type.

- *Port:* defines an individual endpoint by specifying a single network address for a binding.

- *Service:* holds a collection of ports.

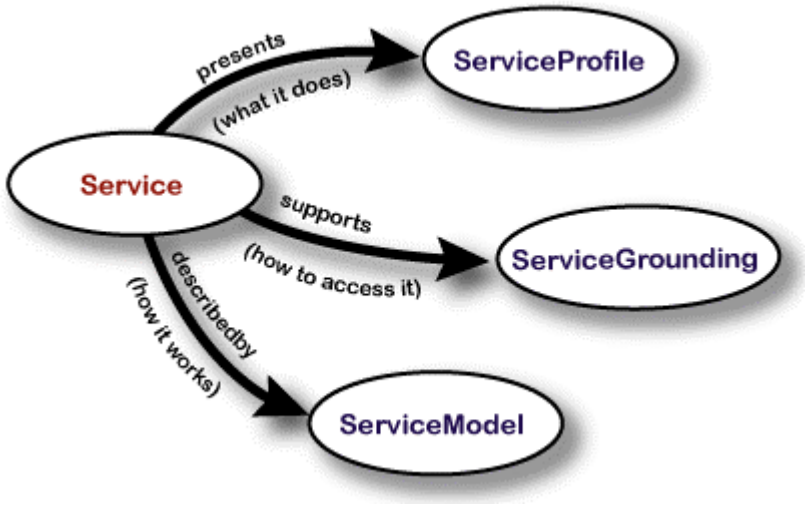
Using the above elements, a WSDL file tells a service requester what type of messages and data types are exchanged, the name of the operations provided by the web service, how the web service is technically implemented i.e. how it is bind to a specific communication

protocols like SOAP (Simple Object Access Protocol) or HTTP (Hyper Text Transfer Protocol), how the messages are expressed in XML (i.e. using literal or encoded), and where the service is actually located.

3.3 OWL-S (Web Ontology Language for Services)

OWL-S (formerly DAML-S) is an ontology consisting of a set of basic classes and properties for declaring and describing services. This ontology tells "What a service provides", "How the service is used" and "How one interacts with it" using its three sub-ontologies (see figure 4) "ServiceProfile", "ServiceModel", and "ServiceGrounding" respectively [18].

Figure 4 Sub-ontologies of OWL-S



Source: OWL-S: Semantic markup for web services

The *Service Profile* sub-ontology helps to present the information required by the service provider to publish a service. It is also used by a service requester to discover a service. As we are interested in semantic description of web services that is used for web service advertisement and discovery, this sub-ontology is discussed in depth in the next sub-section.

The *Service Model* gives a detailed description of the web service's operation. It describes how to ask for the web service and what happens when it is carried out.

The *Service Grounding* specifies the details of how a service can be accessed and presents the concrete specification of the service profile and service model specification. It specifies the communication protocol, message formats, and other service specific details such as port numbers used in contacting the service.

OWL-S is one of the approaches developed to semantically describe web services [19, 20]. The other main similar approaches are IRS (Internet Reasoning Service) and WSMF (Web Service Modeling Framework). IRS has strong user and application integration support but it does not have semantic description of composed services. WSMF is an effort towards developing a comprehensive conceptual architecture, which covers requirements of e-commerce. Compared to these two, OWL-S has a richer service ontology which builds on the semantic web stack of standards. In addition, the service ontology of OWL-S is public.

3.3.1 Service Profile

The service profile does not mandate any representation of services; rather, using the OWL sub-classing, it is possible to create specialized representations of services that can be used as service profiles. OWL-S provides one possible representation through the class *Profile* [18]. The service profile ontology specifies web service descriptions based on their provider, functional and non functional descriptions. The provider information includes the name of the service being offered, a summarized text description on what the service offers, what the service requires to work and any additional information that the compiler of the profile wants to share with the receiver. It also includes contact information of the owner of the service or the customer representative that may provide additional information about the service.

The non-functional description presents information on the characteristics of a service. It incorporates information on the category of a given service, and an unbounded list of service parameters that can contain any type of information about the service like quality guarantee provided by the service.

The functional information has two aspects: the information transformation and the state change. The information transformation is represented by the inputs and outputs. It tells what the web service takes and what it returns to the caller. The state change is represented

by preconditions and the effects. The precondition tells what has to be satisfied before invoking the web service in order to get the desired output. The effect tells the change that's going to happen as a result of a service execution. This functional information about the web service (i.e. inputs, outputs, preconditions and effects) is presented using the service profile sub-ontology. However, this sub-ontology does not provide a schema to describe the instances of this information. Instead it defines how the information can be linked to the service model [18].

The service profile sub-ontology is designed to describe and advertise web services properties and capabilities. However, this representation, as described above, does not provide a means to describe the functionalities of web services which can be used as basic criteria for the multimedia content adaptation web services lookup.

4 Ontology development method

An ontology that can be used to describe web services can be developed in two ways [10, 11]. The first approach provides an extensive ontology of functionalities of web services. An example of such ontology is shown in figure 5. Each class in this kind of ontology corresponds to a specific functionality. Different web services which provide that specific functionality will advertise themselves as an instance of that class. For example a service, S, which gives a speech to text adaptation, can advertise itself as an instance of *AudioToTextAdaptation* class of the ontology given in figure 5.

The second way of developing ontology is based on the description of transformation capability of web services. The transformation can be specified as information transformation and/or state transformation [18]. In information transformation, the information or data which is given to the web service is transformed to the desired output. For example, an audio, which is given to a multi-modal audio translation service, like audio to text web service, will be transformed to a text. This kind of transformation is represented by inputs and outputs of web services which are specified as properties of a class in an ontology. The state transformation, however, represents the external conditions that have to be satisfied for the web service in order to give the desired service and the external effect of the service. For example a service may require a valid credit card in order to give a specified service and as effect it may charge the credit card. This kind of transformation is represented by precondition and effects of web services which are again specified as properties of a class in an ontology. Web services which use such kinds of ontology to advertise themselves are distinguished by their properties.

These two approaches have their own advantages and disadvantages [11]. Ontology developed using the first approach facilitates the discovery process since the matching process is reduced to capability matching. However, this ontology needs to be exhaustive which is difficult and cumbersome. For example, if an ontology is to be developed for audio language translation web services and there are n number of languages, then we need to identify n!/(n-2)! different classes in the ontology which correspond to each pair of languages.

Figure 5 An example of adaptation ontology developed using the first approach[4]

```
<owl:Class rdf:Id=”adaptationService”/>

<owl:Class rdf:Id=”AudioAdaptation”>
  <rdfs:subClassOf rdf:resource=”adapataionService”>
</owl:Class>

<owl:Class rdf:Id=”TextAdaptation”>
  <rdfs:subClassOf rdf:resource=”adapataionService”>
</owl:Class>

<owl:Class rdf:Id=”MultimodalAdaptation”>
  <rdfs:subClassOf rdf:resource=”AudioAdapation”>
</owl:Class>

<owl:Class rdf:Id=”AudioTranslationAdapation”>
  <rdfs:subClassOf rdf:resource=”MultimodalAdapation”>
</owl:Class>

<owl:Class rdf:Id=”AudioToTextAdaptation”>
  <rdfs:subClassOf rdf:resource=” MultimodalAdapation”>
</owl:Class>
```

The second approach, which is based on the transformation capability of web services, simplifies the development of ontology. For example, the classes in the above audio language translation web services ontology can be generalized to one general class that has precondition and effect properties to specify the source and destination languages and input and output properties to describe what it takes and returns (in this specific example both the inputs and outputs are audio files). Even though this approach simplifies ontology development, it has a limitation in identifying services that give a specific functionality. This is because two web services, which have different functionality, can have properties that describe the same transformation capability. For example, a service which takes a text document as an input and produces a text document as an output can be a service which generates the summary of the input text document or a service which generates the text document’s headings.

[4] This ontology of adaptation services is not exhaustive and complete.

As discussed above the two approaches have their own positive aspects in describing web services capabilities. These aspects are important to our ontology. One of the aspects helps the ontology to be general by allowing representation of all possible audio content adaptation web services while the other simplifies the discovery of a web service.

In this research, we have formulated a third approach that incorporates positive aspects of both of the above approaches. This approach is similar to the unified ontology development method described in [28].

The unified methodology requires the ontology developer to first identify purpose and level of formality of the ontology to be developed. Then the developer is required to identify the scope of the ontology based on the purpose and level of formality. The next step in using this methodology is building the ontology. To build the ontology, the methodology suggests four approaches that depend on the level of formality required and the purpose of the ontology. The first approach is less principled approach that is adequate for prototyping. It does not require the developer to follow the above three steps to develop the ontology. The second approach is similar to the above approach but it requires the developer to follow the first three steps of the methodology. The third approach requires the development of a complete intermediate document (an informal ontology) consisting of terms and definitions in a structured form of natural language and then the informal evaluation of this document. The last approach does not require the development of intermediate ontology but identification of formal terms and concepts. After choosing one of the approaches, the last step in this methodology is evaluation and revision cycle.

The steps of the methodology used to develop our proposed ontology are presented in figure 6 and described below.

1. Purpose: At this step, the need for the ontology will be identified.

2. Level of formality: This step requires us to choose the level of formality of the ontology from those stated in chapter 2 (highly informal, structured informal, semi-formal, and rigorously formal). The level of formality of the ontology is determined based on the purposes and users of the ontology. The degree of formality increase with the degree of

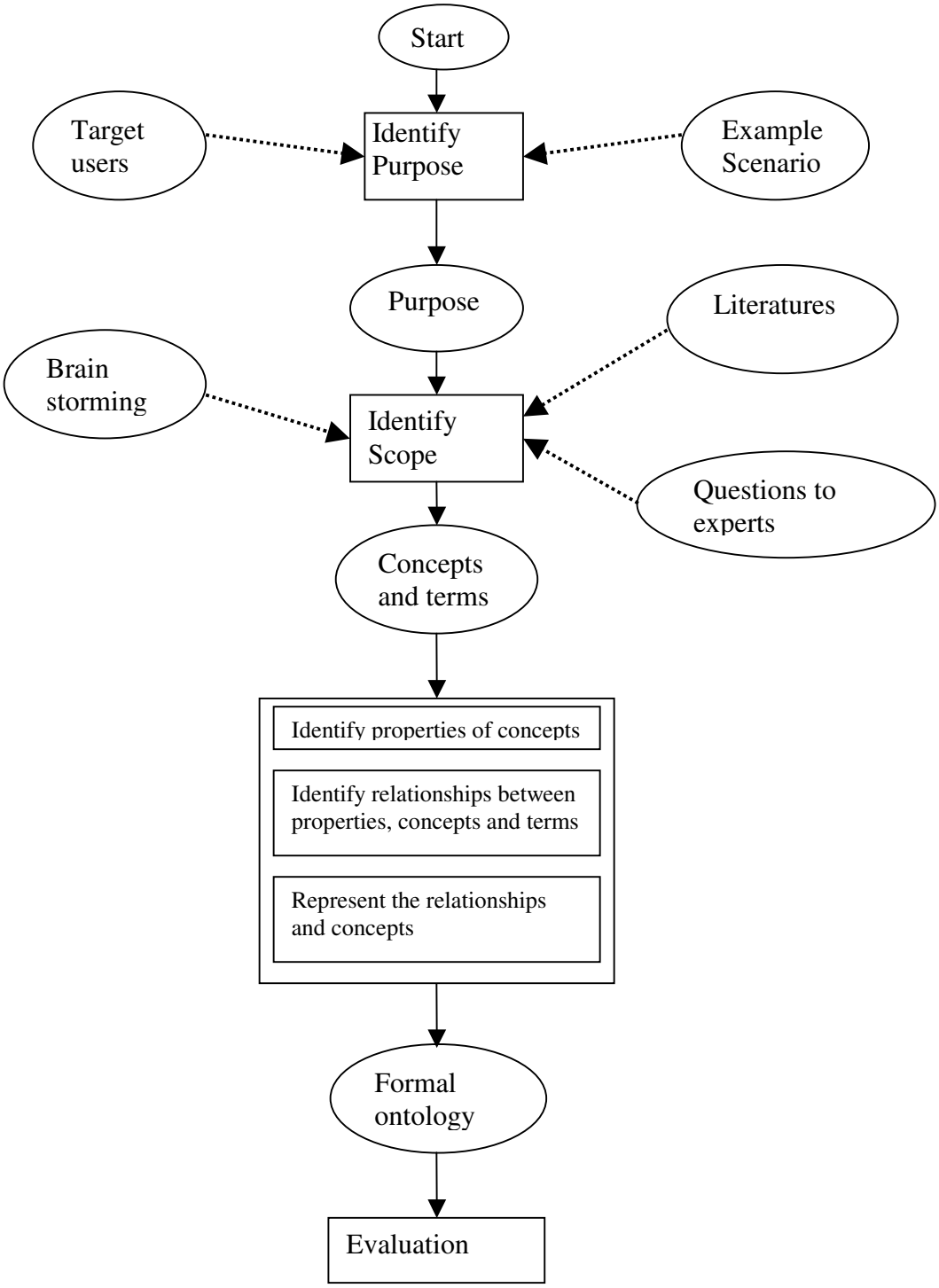
automation in the tasks that the ontology will support. For example, if the users are non-technical people and the primary purpose is to provide a shared vocabulary to facilitate communication between humans, then an informal glossary is enough. However, if the intended use is to support interoperability, which is similar to our case, a more formal representation of the ontology is required.

3. Scope: At this stage the concepts and terms that must be included in the ontology will be identified.

4. Formalizing the ontology: This step identifies properties of concepts and terms listed in the previous step. In addition relationships between the properties, concepts and terms are identified. These identified relationships, properties, and concepts and terms are then represented using ontology representation language.

5. Evaluation: There are various sorts of criteria like consistency and extensibility that can be used to evaluate ontology. This step requires the use of these criteria to evaluate the ontology.

Figure 6 Steps followed while developing the ontology



5 Proposed semantic description for multimedia content adaptation web services

As stated above, the semantic description of multimedia content adaptation web services can be achieved by developing an ontology. Therefore, in this section, we will develop an ontology using the third methodology described in the previous section.

The methodology incorporates five steps. The first step is to define the purpose of the ontology. The second step involves choosing the level of formality of the ontology based on the defined purpose and its users. Then, by selecting the concepts and terms that needs to be incorporated in the ontology, we define the scope. The next step is to formalize the ontology. This is done by identifying the properties of the concepts and their relationships, and representing the ontology using ontology representation language. Finally, the ontology will be evaluated using consistency and extensibility as criteria.

5.1 Purpose

The ontology is developed to have a common understanding of the terms and concepts used in describing and identifying audio content adaptation web services. It also facilitates interoperability and discovery process of audio content adaptation web services. The purpose is described in detail in chapter 1.

5.2 Level of formality

The purpose of this ontology is to have shared understanding of concepts and terms and increase interoperability of web services. The users of the ontology can be both humans and software agents or other services who want to discover an adaptation web service. Therefore, among the four levels of formalities, this ontology is identified to be semi-formal.

5.3 Scope

The ontology which is going to be developed is for audio content adaptation web services. The concepts and terms which are going to be included in the ontology are listed in the next sub section.

5.3.1 Concepts and terms

Different techniques, which are applicable to different requirements of audio adaptation, can be used by audio content adaptation services to achieve their objective. These techniques are listed in various papers [4, 5, 9, 14, 16, 23, 25]. Based on these techniques, we will try to list and describe possible audio adaptation web services as exhaustively as possible.

AudioReductionAdaptationServices: are services that reduce the bit rate of an audio file.

AudioSamplingRateReductionAdaptationServices: are services that reduce the rate at which an audio file is sampled.

StereoToMonoAdaptaionServices: are services that convert an audio which has two channels to a single channel which in effect reduces the bandwidth required to transmit the audio.

AudioFormatTranscodingAdaptionServices: are services that change the format of an audio file. For example, an audio which is in WAV format can be converted to an mp3 format using a service which implements format transcoding techniques.

AudioLanguageTranslationAdaptaionServices: These services translate an audio in one language to another language. For example, an audio file which contains a speech made about pervasive computing in English language can be translated to a speech in Amharic language using a service which implements language translation technique.

AudioMediaTranslationAdaptationServices: Using a service that implements translation adaptation technique an audio file is translated to another media element like text or animation.

AudioToTextAdaptatinServices: These services are specific types of *AudioMediaTranslationAdaptationServices*. They convert an audio file to a text document.

In addition to the above concepts and terms that are used to represent the functionalities of audio content adaptation web services, we need to identify concepts and terms of

multimedia data types. The multimedia data type concepts and terms help to represent the input and/or output of the adaptation web services in a standard way. These concepts are listed below.

MultimediaDatatype: is used to represent different multimedia data types: audio, video, image, text, graphics or animation.

ImageDatatypeFormat: is used to represent image data type formats.

AudioDatatypeFormat: is used to represent audio data type formats.

VideoDatatypeFormat: represents the format of a video data type.

TextDatatypeFormat: represents the format of a text data type.

GraphicsDatatypeFormat: is used to represent the format of graphics data type.

AnimationDatatypeFormat: is used to represent the format of animation data type.

5.3.2 Classification of concepts and terms

The services listed above can be classified based on different perspectives of the adaptation techniques used to list the services. These classifications are presented in [9, 16]. The taxonomy of audio adaptation services will be developed using some of these broad classifications as its base.

The classifications which are stated in [9, 16] and used as a base for this work are summarized as follows.

The adaptation techniques can be classified as static or dynamic based on when they are used to generate the adapted content.

> *Static adaptation techniques*: are techniques used to generate different versions that differ in quality and processing requirement. The techniques which are categorized here have always the same resource requirement and result, for the same input. The method which they use to adapt the content is always fixed irrespective of the

resource available at the given time. For example, if they have to sample a media file in order to adapt, the interval with which they take the sample is always the same, i.e. it does not change with time.

Dynamic adaptation techniques: are techniques that are used to generate a content that meets the user and/or device constraints on-the-fly. Depending upon the resource available, adaptation techniques under this category use various techniques to adapt the content. For example, if the multimedia file has to be sampled in order to be adapted, then the service which implements this technique may sample too many or very few samples depending on the resource available to process the content of the multimedia file.

According to the media types involved, the adaptation techniques can be classified into unimodal or multimodal adaptation.

Unimodal adaptation techniques: This category incorporates techniques used to create different versions of a media element with different qualities, formats and resource requirements. For example, techniques that are used to convert an audio from one format, WMA (Windows Media Audio) file to another, mp3 are considered as unimodal adaptation techniques.

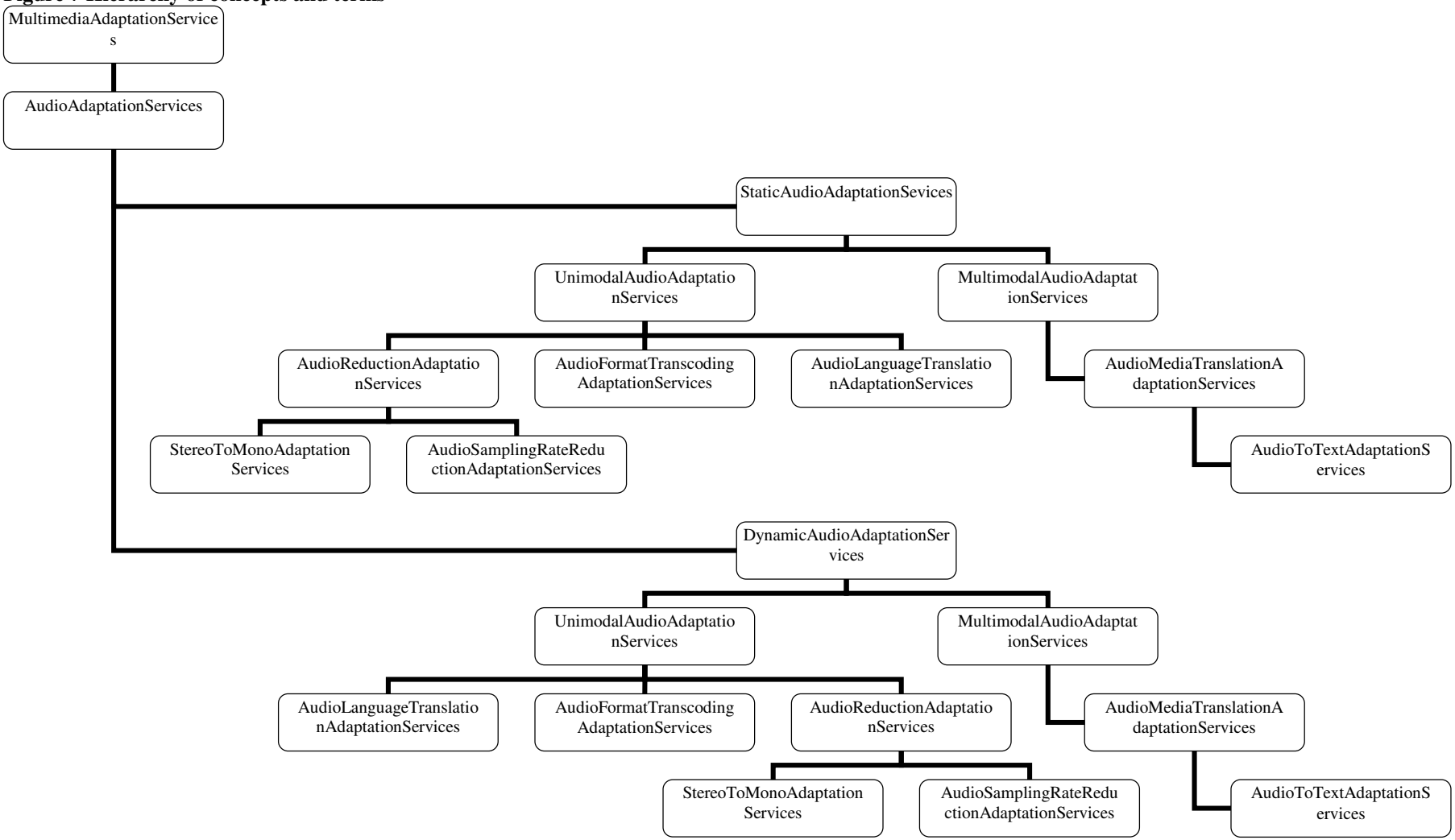
Multimodal adaptation techniques: This category incorporates techniques that are used to convert one media type to another so that the converted content meets the particular device specification. For example, audio to text conversion.

5.4 Formalization of the ontology

5.4.1 Hierarchy of concepts and terms

The concepts identified above are represented as classes in the ontology. These classes can be arranged in a hierarchy that is useful for service discovery. This hierarchy, which is shown in figure 7, is based on the broad classifications described in the previous section. It will be used with the service properties to give a complete ontology of services. The full representation of the ontology using OWL is presented in appendix 2.

Figure 7 Hierarchy of concepts and terms



The super class in the ontology is *MultimediaAdaptationSevices* (see figure 7). This class can have different sub-classes of adaptation services that are specific to each media element like audio adaptation services, text adaptation service, etc. However our scope is limited to audio adaptation services. Therefore, the description of the sub-classes and properties are restricted to audio adaptation services.

AudioAdaptationServices class inherits all properties of *MultimediaAdaptationSevices,* but it restricts the value of the input property, *hasInputType*, to be *Audio*. This is because, all audio adaptation services take an audio file as their input. This class has two sub-classes: *StaticAudioAdaptationServices and DyanamicAudioAdaptationServices*. These two classes have similar sub-classes, which will be discussed subsequently.

One of the sub-classes of *StaticAudioAdaptationServices* and *DynamicAudioAdaptationServices* is *UnimodalAudioAdaptationServices*. *UnimodalAudioAdaptationServices* has additional restriction on its output data type property other than the restriction inherited on its input data type. The instance of this property is restricted to be *Audio*. This class has three subclasses under it: *AudioReductionAdaptationServices*, *AudioLanguageTranslationAdaptationServices* and *AudioFormatTranscodingAdaptationServices*. Among these sub-classes, *AudioFormatTranscodingAdaptationServices* has additional restrictions on its precondition and effect properties. It restricts the values of these properties to be instances of the *MultimediaDatatypeFormat* class. *AudioReductionAdaptationServices* has two sub-classes: *StereoToMonoAdaptationServices* and *AudioSamplingRateReductionAdaptationService*.

The other sub-class of *StaticAudioAdaptationServices* and *DynamicAudioAdaptationServices* is *MultimodalAudioAdaptationServices*. *MultimodalAudioAdaptationServices* has one sub-class: *AudioMediaTranslationAdaptationServices*. This sub-class in turn has one sub-class *AudioToTextAdaptationServices*. The instance of the output data type property of *AudioToTextAdaptationServices* is restricted to text.

5.4.2 Properties of concepts

For audio adaptation web services we have identified the following properties, most of which are similar to those of OWL-S profile class. These properties are shown in table 1. These properties are general to all multimedia adaptation web services, so to make extensibility of this work to other media elements easier we have put them to be properties of the general class *MultimediaAdaptationSevices*

Table 1 Properties of multimedia adaptation web services

MultimediaAdaptationServices	
Property name	**Value**
adaptationServiceName	String
adaptationTime	Duration
adaptedMediaQuality	String
hasEffect	Instance of a class
hasInputType	Instance of MultimediaDatatype class
hasOutputType	Instance of MultimediaDatatype class
hasServicePrice	Instance of a class
hasPrecondition	Instance of a class
hasServiceProvider	Instance of a class
serviceDescription	String

adaptationServiceName: is the property of a service which holds the actual name of the service.

adaptationTime: This property holds the time or duration (in seconds) that the service takes in order to do the adaptation.

adaptedMediaQuality: This property takes one of the five subjective quality rating scales: Excellent, Good, Fair, Poor, or Bad (ITU-T recommendations stated in [29]).

hasEffect: holds a value which is an instance of a class. The class varies depending on the web service. For example, if the effect of the service is to change the language of an audio file, the class can be from an external language ontology which describes the language may be based on ISO639 (i.e. two or three letter codes)[5] or if the effect of the service is to change the format of the service then the class can be an instance of the *MultimediaDataTypeFormat*.

hasInputType: The value of this property is an instance of the *MultimediaDatatype* class. In all of the sub-classes described in this ontology the instance value for this property is audio.

hasOutputType: The value of this property is also an instance of the *MultimediaDatatype* class.

hasServicPrice: is a property which can be used to link the service with an external price ontology's class.

hasPrecondition: holds an instance of a class that can be used to describe the prerequisite of the service like format or language of the audio.

hasServiceProvider: is a property that can be used to give information about the service provider. It can be used as a link to external ontologies like the Actor ontology[6] that is used to record address and other information of the provider.

serviceDescription: This parameter holds any description about the service.

All types of multimedia data type formats have similar properties. Therefore, we have put all these properties to be properties of the general class *MultimediaDatatypeFormat.* These properties are shown in table 2 and described below.

[5] http://www.loc.gov/standards/iso639-2/
[6] http://www.daml.org/services/owl-s/1.1/ActorDefault.owl

Table 2 Properties of *MultimediaDatatypeFormat* class

MultimediaDatatypeFormat	
Property name	**Value**
hasDatatype	Instance of the *MultimediaDatatype* class
hasFormatName	String
additionalInformation	String

hasDatatype: This property is an instance of the class *MultimediaDatatype*. It is used to link the data type format with one of the instances of the multimedia data types.

hasFormatName: is a property which is used to hold the name of the format type such as mp3 or wav for audio data type.

additionalInformation: is a property which is used to hold additional information that the service describer wants to incorporate with the definition of the format for the specified multimedia data type.

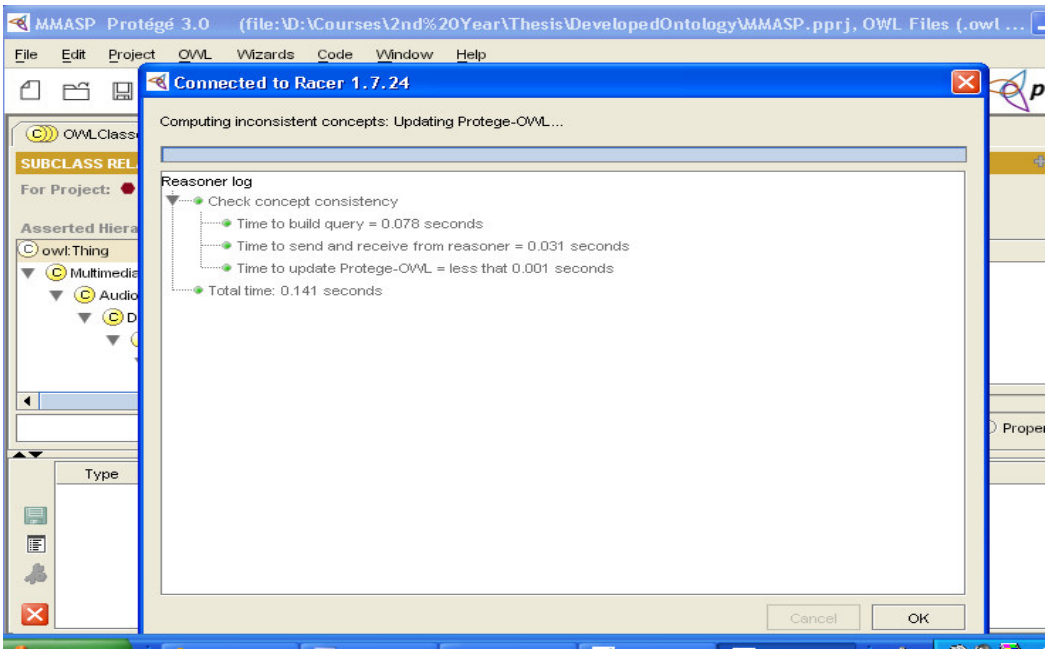
5.5 Evaluation of the ontology

There are some general criteria that can be used to evaluate an ontology [28]. The two most important criteria are: 1) consistency and 2) extensibility and reusability. We have tested our ontology using both evaluation criteria.

Consistency: The consistency of an ontology is determined by checking whether or not a class in the ontology can have any instances. A class is considered to be inconsistent if it cannot possibly have any instances [8]. It is possible to check the consistency of an ontology using Protégé version 3 software, which is developed by Stanford

Medical Informatics[7]. Protégé uses third party reasoners like RACER (Renamed ABox and Concept Expression Reasoner)[8] to perform such checks. The ontology is checked using this feature of Protégé and no inconsistency is detected. A screenshot of the test result is shown in figure 8.

Figure 8 Consistency check result



Extensibility and reusability: The extensibility of an ontology is checked by looking at the generality of the classes defined in the ontology while reusability is checked by looking at the applicability of these classes to other ontologies. Using these criteria, this ontology is found to be extensible as it has general class like *MultimediaAdaptationServices* that can be used to extend to other multimedia adaptation services other than audio adaptation services. It is also found to be reusable as it has classes like *MultimediaDatatype* and *MultimediaDatatypeFormat* that can be used in other ontologies.

[7] http://smi.stanford.edu/
[8] http://www.racer-systems.com/

6 Implementation

To show the usability of the proposed ontology in service discovery, we have developed a UDDI like registry. This registry allows advertisement and lookup of audio content adaptation web services using its two web based interfaces (see figure 9 and 10). It also keeps information about the relationships between the concepts of the ontology.

Figure 9 Web interface used to publish a web service

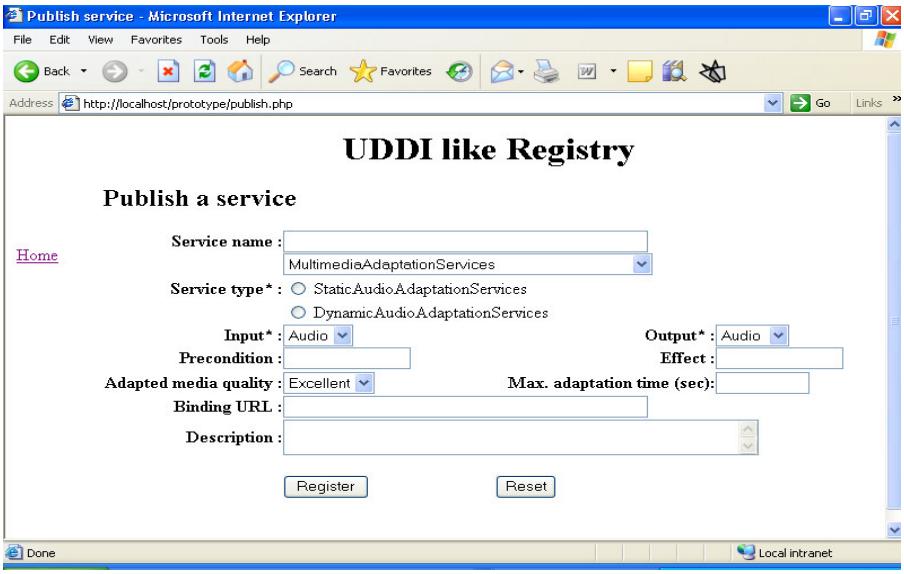
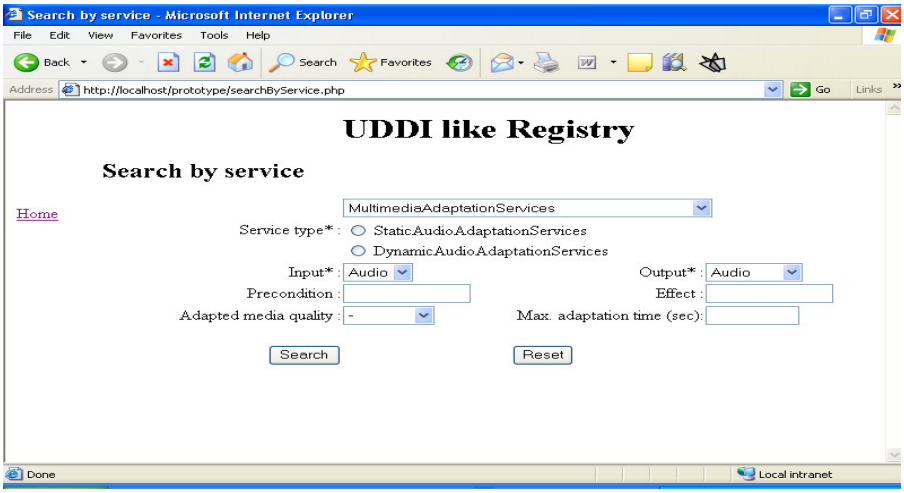


Figure 10 Web interface used to search a web service



To maintain the information about the relationships, the registry has two tables: *tblTaxonomyClasses* and *tblTaxonomyRn*. *tblTaxonomyClasses* holds all concepts that are

related to the functionality of audio content adaptation web services. The hierarchical relationship between these concepts is stored in *tblTaxonomyRn*. This information is used to infer relationships during web service lookup. For example, if a web service, say S, that reduces the bit rate of an audio file is advertised as *AudioSamplingRateRedctionAdaptationServices*, and a user looks for a web service that does the same functionality but under *AudioReductionAdaptationServices*, the registry will display web service S since *AudioSamplingRateRedctionAdaptationServices* is a sub-class of *AudioReductionAdaptationServices*.

The registry also holds information about the functional and non functional properties of the audio content adaptation web services. These properties are identified and depicted in the proposed ontology. To store this information the registry uses tables: *tblService* and *tblMultimediaDatatype.* Information about the service provider is stored in table *tblContactInformation*. This table is structured based on the properties of the external Actor[9] ontology.

6.1 Demonstration

To show the functionalities of the UDDI like registry, we have published 50 web services on the registry and made some searches with different criteria.

Criteria 1:

Table 3 Values for criteria 1

Criteria Fields	Value
Service type	StereoToMonoAdaptationServices under DynamicAudioAdaptationServices
Input	Audio
Output	Audio
Precondition	-
Effect	-
Adapted media quality	Excellent
Max. adaptation time (sec)	-

[9] http://www.daml.org/services/owl-s/1.1/ActorDefault.owl

As can be seen in table 3, we have searched for a web service that converts a stereo audio to mono and returns the audio with an excellent quality. The registry has returned two web services that fulfill these search criteria.

Criteria 2:

Table 4 Values for criteria 2

Criteria Fields	Value
Service type	AudioLanguageTranslationAdaptationServices
Input	Audio
Output	Audio
Precondition	en
Effect	fr
Adapted media quality	-
Max. adaptation time (sec)	-

As shown in table 4, we have searched for a web service that translates a speech in English to French. The registry has returned one web service that fulfills these search criteria.

The above demonstrations show that searching a web service described semantically is very easy and does not require remembering the names of the web services. In addition, the relationship information maintained between the concepts has made the discovery process very easy.

7 Conclusion and future works

In this research, we have shown the limitations of current standards that are used to describe and list web services. These limitations arise mainly due to the fact that web services are described using syntactic descriptions. Syntactic description does not tell the meanings of the words used in describing a web service and their relationship which are important information for web service discovery. In addition, they allow lookup of web services only using keywords which make web service discovery a very difficult task for the user.

To improve these limitations that are especially related to multimedia content adaptation web services, we have developed an ontology that can be used to describe audio content adaptation web services semantically. This ontology allows to describe both functional and non functional properties of audio content adaptation web services. This helps to easily describe these adaptation web services. It has also incorporated a taxonomy that can be used to facilitate searching of the web services. In addition, it has a multimedia data type ontology that can be used to standardize the parameters of the web services.

To develop our ontology, we have first reviewed different ontology representation languages and shown their development. We have also explored current standards and ontologies that can be used to describe web services. Moreover, we have seen different methodologies that can be used to develop ontology and formulated a new approach based on which we have developed our ontology.

Finally, in order to evaluate the proposed ontology, we have developed a prototype. The prototype is a UDDI like registry that uses our ontology to advertise (publish) audio content adaptation web services and search for them. The registry has a user interface and is accessed through the web.

To semantically describe web service we have proposed an ontology for audio content adaptation web services. However, this ontology can be extended to other multimedia content adaptation web services.

Furthermore, the ontology is developed mainly for description of audio content adaptation web services that is important for web service lookup. But it does not provide any information on how one uses the service or interacts with it. Therefore, it needs to be extended or linked to OWL-S's service model and/or service grounding to incorporate this information.

References

[1]. "RDF Vocabulary Description Language 1.0: RDF Schema", edited by Brickley, D. and Guha, R.V., *W3C Proposed Recommendation*, December 15, 2003. (URL: http://www.w3.org/TR/2003/PR-rdf-schema-20031215/).

[2]. Bechhofer, S., Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F. and Stein, L. A., "OWL Web Ontology Language Reference", *W3C Recommendation*, February 10, 2004. (URL: http://www.w3.org/TR/2004/REC-owl-ref-20040210/).

[3]. Connolly, D., Harmelen, F., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F. and Stein, L. A., "DAML+OIL (March 2001) Reference Description", *W3C Note*, December 18, 2001. (URL: http://www.w3.org/TR/2001/NOTE-daml+oil-reference-20011218).

[4]. Girma, B., Brunie, L. and Pierson, J-M., "Modeling Service-Based Multimedia Content Adaptation in Pervasive Computing", *ACM Proceedings of the first conference on computing frontiers (CF'04)*, pp. 60-64, Ischia, Italy, April 14-16, 2004.

[5]. Girma, B., Brunie, L. and Pierson, J-M., "Realization of Distributed Content Adaptation with Service-Based Approach for Pervasive Systems", *Unpublished*, INSA de LYON.

[6]. Gruber, T. R., "Toward Principles for the Design of Ontologies Used for Knowledge Sharing", *Technical Report KSL 93-04*, 1993.

[7]. Harrocks, I., "DAML+OIL: a Reason-able Web Ontology Language", *Proceedings of the 8th International Conference on Extending Database Technology: Advances in Database Technology*, pp. 2-13, Springer-Verlag, London, UK, 2002.

[8]. Horridge, M., Knublauch, H., Rector, A., Stevens, R. and Wroe, C., "A Practical Guide To Building OWL Ontologies Using The Protégé-OWL Plugin and CO-ODE Tools", Edition 1.0, University of Manchester, August 27, 2004.

[9]. Lei, Z. and Georganas, N. D., "Context-Based Media Adaptation in Pervasive Computing", *Proc. ACM Multimedia 2001 Doctoral Symposium*, Ottawa, September 2001.

[10].Lemmens, R. and Vries, M., "Semantic Description of Location Based Web Services Using an Extensible Location Ontology", *Proceedings Munster: Geoinformation and Mobility*, IfGI prints No.22, July 2004.

[11].Martin, D., Paolucci, M., Mcllraith, S., Burstein, M., McDermott, D., McGuinness, D., Parsia, B., Payne, T., Sabou, M., Solanki, M., Srinivasan, N. and Sycara, K., "Bringing Semantics to Web Services: The OWL-S approach", *First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004)*, California, USA, 2004.

[12].Meaning of Ubiquitous Computing. (URL: http://www.hyperdictionary.com).

[13].Miller, E., "An Introduction to Resource Description Framework", D-Lib Magazine, ISSN 1082-9873, May 1998. (URL: http://www.dlib.org/dlib/may98/05contents.html).

[14].Mohan, R., Smith, J. R. and Li, C. S., "Adapting Multimedia Internet Content for Universal Access," *IEEE Transactions on Multimedia*, Vol. 1, No. 1, pp. 104-114, 1999.

[15].Moreau, L., Miles, S., Papay, J., Decker, K. and Payne, T., "Publishing Semantic Description of Services", *Semantic Grid Workshop at GGF9*, 2005.

[16].Mulugeta, L., "Metadata Supported Content Adaptation in Distributed Systems", *Ph. D. Thesis*, University of Klagenfurt, Austria, June 2004.

[17].Ollermann, W. L., "Architecting Web Services", Springer-Verlag New York, Inc., 175 Fifth Avenue, New York, NY, 2001.

[18].Paolucci, M., Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., Mcllraith, S., Narayanan, S., Parsia, B., Payne, T., Sirin, E., Srinivasan, N. and

Sycara, K., “OWL-S: Semantic Markup for Web Services”, *W3C Member Submission*, November 2004. (URL: http://www.w3.org/Submission/2004/subm-owl-s-20041122/).

[19]. Payne, T., Bandara, A., Roure, D. and Clemo, G., “An Ontological Framework for Semantic Description of Devices”, *3rd International Semantic Web Services (ISWC 2004)*, Hiroshima, Japan, November 7-11, 2004.

[20]. Payne, T., Cabral, L., Domingue, J., Motta, E. and Hakimpour, F., “Approaches to Semantic Web Services: An Overview and Comparisons”, *In proceedings of the First European Semantic Web Symposium (ESWS 2004)*, Herakilon, Crete, Greece, May 10-12, 2004.

[21]. Pease, A., “Why use DAML?”, 2002. (URL: http://www.daml.org/2002/04/why.html).

[22]. Roeder, S., Goodwin, R., Doshi, P. and, Akkiraju, R., “A method for semantically Enhancing the Service Discovery capabilities of UDDI”, *In proceedings of the Workshop on Information Integration on the Web (IJCAI 2003)*, Mexico, August 9-10, 2003.

[23]. Shaha, N., Desai, A. and Parashar, M., “Multimedia Content Adaptation for QoS Management over Heterogeneous Networks”, *Proc. of International Conference on Internet Computing 2001*, pp. 642-648, Computer Science Research Education and Applications (CSREA) press, June 2001.

[24]. Siberski, W., Thaden, U., and Nejdl, W., “A Semantic Web Base Peer-to-Peer Service Registry Network”, *Technical Report*, Learning Lab Lower Saxony, University of Hannover, Germany, February 17, 2003.

[25]. Smith, J. R., Mohan R. and Li, C. -S., “Transcoding Internet Content for Heterogeneous Client Devices”, *Proc. IEEE Int. Conf. on Circuits and Syst. (ISCAS)*, May 1998.

[26]. Terziyan, V. and Kononenko, O., "Semantic Web Enabled Web Services: State-of-Art and Industrial Challenges", *Web Services- ICWS –Europe 2003, Lecture notes in Computer Science*, Vol. 2853, pp. 183-197, Springer-Verlag, 2003.

[27]. Uschold, M. and Gruninger, M., "Ontologies: Principles, Methods and Applications", *Knowledge Engineering Review*, Vol. 11, No. 2, February 1996.

[28]. Uschold, M., "Building Ontologies: Towards a Unified Methodology", *To appear in Proceedings of Experts system*, September 1996.

[29]. Voelcker, R. M., Hollier, M. P., Rimell, A. N. and Hands, D. S., "Multi-modal Perception", *BT Technol J*, Vol 17, No 1, January 1999.

[30]. Weerawarana, S., Christensen, E., Curbera, F. and Meredith, G., "Web Services Description Language (WSDL) 1.1", *W3C Note*, March 15, 2001. (URL: http://www.w3.org/TR/2001/NOTE-wsdl-20010315).

Appendices

Appendix 1: WSDL description of a text to speech translation web service

```
<?xml version="1.0" encoding="utf-8"?>
<definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:s0="http://if-5148.insa-
lyon.fr/TextToSpeech/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
targetNamespace="http://if-5148.insa-lyon.fr/TextToSpeech/"
xmlns="http://schemas.xmlsoap.org/wsdl/">
  <types>
    <s:schema elementFormDefault="qualified" targetNamespace="http://if-
5148.insa-lyon.fr/TextToSpeech/">
      <s:element name="TextToSpeechFromURL">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="fileURL"
type="s:string" />
            <s:element minOccurs="1" maxOccurs="1" name="outputFormat"
type="s0:Format" />
            <s:element minOccurs="1" maxOccurs="1" name="language"
type="s0:Language" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:simpleType name="Format">
        <s:restriction base="s:string">
          <s:enumeration value="wav" />
          <s:enumeration value="mp3" />
        </s:restriction>
      </s:simpleType>
      <s:simpleType name="Language">
        <s:restriction base="s:string">
          <s:enumeration value="EnglishMan1" />
          <s:enumeration value="EnglishMan" />
```

```
          <s:enumeration value="EnglishWoman" />
          <s:enumeration value="FrenchMan" />
          <s:enumeration value="DeutschMan" />
          <s:enumeration value="FrenchWoman" />
          <s:enumeration value="EnglishWoman2" />
          <s:enumeration value="EnglishMan2" />
          <s:enumeration value="DeutschWoman" />
        </s:restriction>
      </s:simpleType>
      <s:element name="TextToSpeechFromURLResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1"
name="TextToSpeechFromURLResult" type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="TextToSpeechFromText">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="text"
type="s:string" />
            <s:element minOccurs="1" maxOccurs="1" name="outputFormat"
type="s0:Format" />
            <s:element minOccurs="1" maxOccurs="1" name="language"
type="s0:Language" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="TextToSpeechFromTextResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1"
name="TextToSpeechFromTextResult" type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="string" nillable="true" type="s:string" />
```

```
    </s:schema>
  </types>
  <message name="TextToSpeechFromURLSoapIn">
    <part name="parameters" element="s0:TextToSpeechFromURL" />
  </message>
  <message name="TextToSpeechFromURLSoapOut">
    <part name="parameters" element="s0:TextToSpeechFromURLResponse" />
  </message>
  <message name="TextToSpeechFromTextSoapIn">
    <part name="parameters" element="s0:TextToSpeechFromText" />
  </message>
  <message name="TextToSpeechFromTextSoapOut">
    <part name="parameters" element="s0:TextToSpeechFromTextResponse" />
  </message>
  <message name="TextToSpeechFromURLHttpPostIn">
    <part name="fileURL" type="s:string" />
    <part name="outputFormat" type="s:string" />
    <part name="language" type="s:string" />
  </message>
  <message name="TextToSpeechFromURLHttpPostOut">
    <part name="Body" element="s0:string" />
  </message>
  <message name="TextToSpeechFromTextHttpPostIn">
    <part name="text" type="s:string" />
    <part name="outputFormat" type="s:string" />
    <part name="language" type="s:string" />
  </message>
  <message name="TextToSpeechFromTextHttpPostOut">
    <part name="Body" element="s0:string" />
  </message>
  <portType name="ServiceSoap">
    <operation name="TextToSpeechFromURL">
      <documentation>Traduit le contenu d'un fichier texte stockÃ© a
l'url fournit dans un language choisit et dans un format donnÃ©. Le texte
ne doit pas depasser 5Ko</documentation>
      <input message="s0:TextToSpeechFromURLSoapIn" />
      <output message="s0:TextToSpeechFromURLSoapOut" />
    </operation>
```

```
    <operation name="TextToSpeechFromText">
      <documentation>Traduit le texte dans la langue et le format de
sortie voulu. Le texte ne doit pas depasser 5Ko</documentation>
      <input message="s0:TextToSpeechFromTextSoapIn" />
      <output message="s0:TextToSpeechFromTextSoapOut" />
    </operation>
  </portType>
  <portType name="ServiceHttpPost">
    <operation name="TextToSpeechFromURL">
      <documentation>Traduit le contenu d'un fichier texte stockÃ© a
l'url fournit dans un language choisit et dans un format donnÃ©. Le texte
ne doit pas depasser 5Ko</documentation>
      <input message="s0:TextToSpeechFromURLHttpPostIn" />
      <output message="s0:TextToSpeechFromURLHttpPostOut" />
    </operation>
    <operation name="TextToSpeechFromText">
      <documentation>Traduit le texte dans la langue et le format de
sortie voulu. Le texte ne doit pas depasser 5Ko</documentation>
      <input message="s0:TextToSpeechFromTextHttpPostIn" />
      <output message="s0:TextToSpeechFromTextHttpPostOut" />
    </operation>
  </portType>
  <binding name="ServiceSoap" type="s0:ServiceSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
style="document" />
    <operation name="TextToSpeechFromURL">
      <soap:operation soapAction="http://if-5148.insa-
lyon.fr/TextToSpeech/TextToSpeechFromURL" style="document" />
      <input>
        <soap:body use="literal" />
      </input>
      <output>
        <soap:body use="literal" />
      </output>
    </operation>
    <operation name="TextToSpeechFromText">
      <soap:operation soapAction="http://if-5148.insa-
lyon.fr/TextToSpeech/TextToSpeechFromText" style="document" />
```

```
      <input>
        <soap:body use="literal" />
      </input>
      <output>
        <soap:body use="literal" />
      </output>
    </operation>
  </binding>
  <binding name="ServiceHttpPost" type="s0:ServiceHttpPost">
    <http:binding verb="POST" />
    <operation name="TextToSpeechFromURL">
      <http:operation location="/TextToSpeechFromURL" />
      <input>
        <mime:content type="application/x-www-form-urlencoded" />
      </input>
      <output>
        <mime:mimeXml part="Body" />
      </output>
    </operation>
    <operation name="TextToSpeechFromText">
      <http:operation location="/TextToSpeechFromText" />
      <input>
        <mime:content type="application/x-www-form-urlencoded" />
      </input>
      <output>
        <mime:mimeXml part="Body" />
      </output>
    </operation>
  </binding>
  <service name="Service">
    <documentation>Ceci est un web service de conversion text vers audio
developpÃ© dans le cadre d'un projet sur l'adaptation de
contenu.</documentation>
    <port name="ServiceSoap" binding="s0:ServiceSoap">
      <soap:address location="http://localhost/TextToSpeech/Service.asmx"
/>
    </port>
    <port name="ServiceHttpPost" binding="s0:ServiceHttpPost">
```

```
      <http:address location="http://localhost/TextToSpeech/Service.asmx"
/>
    </port>
  </service>
</definitions>
```

Appendix 2: Representation of the proposed ontology using OWL

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns="http://www.owl-ontologies.com/MMASP.owl#"
  xml:base="http://www.owl-ontologies.com/MMASP.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="AudioLanguageTranslationAdaptationServices">
    <owl:disjointWith>
      <owl:Class rdf:ID="AudioFormatTranscodingAdaptationServices"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="AudioReductionAdaptationServices"/>
    </owl:disjointWith>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="UnimodalAudioAdapationServices"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="StaticAudioAdaptationServices">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="AudioAdaptationServices"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#UnimodalAudioAdapationServices">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="DynamicAudioAdaptationServices"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="hasOutputType"/>
        </owl:onProperty>
        <owl:hasValue>
          <owl:Thing rdf:ID="Audio"/>
```

```
      </owl:hasValue>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:ID="MultimodalAudioAdaptationServices"/>
  </owl:disjointWith>
  <rdfs:subClassOf rdf:resource="#StaticAudioAdaptationServices"/>
</owl:Class>
<owl:Class rdf:ID="AudioToTextAdaptationServices">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:hasValue>
        <owl:Thing rdf:ID="Text"/>
      </owl:hasValue>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#hasOutputType"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="AudioMediaTranslationAdaptationServices"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="StereoToMonoAdaptationServices">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#AudioReductionAdaptationServices"/>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:ID="AudioSamplingRateReductionAdaptationServices"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:ID="ImageDatatypeFormat">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="MultimediaDatatypeForamt"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
```

```
          <owl:ObjectProperty rdf:ID="hasDatatype"/>
        </owl:onProperty>
        <owl:hasValue>
          <owl:Thing rdf:ID="Image"/>
        </owl:hasValue>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="MultimediaAdaptationServices">
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:allValuesFrom>
          <owl:DataRange>
            <owl:oneOf rdf:parseType="Resource">
              <rdf:rest rdf:parseType="Resource">
                <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                >Good</rdf:first>
                <rdf:rest rdf:parseType="Resource">
                  <rdf:rest rdf:parseType="Resource">
                    <rdf:rest rdf:parseType="Resource">
                      <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                      >Bad</rdf:first>
                      <rdf:rest
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                    </rdf:rest>
                    <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                    >Poor</rdf:first>
                  </rdf:rest>
                  <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                  >Fair</rdf:first>
                </rdf:rest>
              </rdf:rest>
```

```
              <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
              >Excellent</rdf:first>
            </owl:oneOf>
          </owl:DataRange>
        </owl:allValuesFrom>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="adaptedMediaQuality"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="AnimationDatatypeFormat">
    <rdfs:subClassOf rdf:resource="#MultimediaDatatypeForamt"/>
  </owl:Class>
  <owl:Class rdf:about="#AudioAdaptationServices">
    <rdfs:subClassOf rdf:resource="#MultimediaAdaptationServices"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:hasValue rdf:resource="#Audio"/>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="hasInputType"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="GraphicsDatatypeFormat">
    <rdfs:subClassOf rdf:resource="#MultimediaDatatypeForamt"/>
  </owl:Class>
  <owl:Class rdf:about="#DynamicAudioAdaptationServices">
    <rdfs:subClassOf rdf:resource="#AudioAdaptationServices"/>
  </owl:Class>
  <owl:Class rdf:about="#AudioReductionAdaptationServices">
    <rdfs:subClassOf rdf:resource="#UnimodalAudioAdapationServices"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#AudioFormatTranscodingAdaptationServices"/>
    </owl:disjointWith>
```

```
    <owl:disjointWith
rdf:resource="#AudioLanguageTranslationAdaptationServices"/>
  </owl:Class>
  <owl:Class rdf:ID="TextDatatypeFormat">
    <rdfs:subClassOf rdf:resource="#MultimediaDatatypeForamt"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#hasDatatype"/>
        </owl:onProperty>
        <owl:hasValue rdf:resource="#Text"/>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="MultimediaDatatype">
    <owl:equivalentClass>
      <owl:Class>
        <owl:oneOf rdf:parseType="Collection">
          <owl:Thing rdf:about="#Audio"/>
          <owl:Thing rdf:about="#Image"/>
          <owl:Thing rdf:about="#Text"/>
          <owl:Thing rdf:ID="Video"/>
          <owl:Thing rdf:ID="Graphics"/>
          <owl:Thing rdf:ID="Animation"/>
        </owl:oneOf>
      </owl:Class>
    </owl:equivalentClass>
  </owl:Class>
  <owl:Class rdf:about="#AudioMediaTranslationAdaptationServices">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#MultimodalAudioAdaptationServices"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#MultimodalAudioAdaptationServices">
    <rdfs:subClassOf rdf:resource="#DynamicAudioAdaptationServices"/>
    <rdfs:subClassOf rdf:resource="#StaticAudioAdaptationServices"/>
    <owl:disjointWith rdf:resource="#UnimodalAudioAdapationServices"/>
  </owl:Class>
```

```
  <owl:Class rdf:about="#AudioSamplingRateReductionAdaptationServices">
    <rdfs:subClassOf rdf:resource="#AudioReductionAdaptationServices"/>
    <owl:disjointWith rdf:resource="#StereoToMonoAdaptationServices"/>
  </owl:Class>
  <owl:Class rdf:about="#AudioFormatTranscodingAdaptationServices">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:allValuesFrom>
          <owl:Class rdf:ID="AudioDatatypeFormat"/>
        </owl:allValuesFrom>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="hasPrecondition"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <owl:disjointWith
rdf:resource="#AudioLanguageTranslationAdaptationServices"/>
    <owl:disjointWith rdf:resource="#AudioReductionAdaptationServices"/>
    <rdfs:subClassOf rdf:resource="#UnimodalAudioAdapationServices"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:allValuesFrom>
          <owl:Class rdf:about="#AudioDatatypeFormat"/>
        </owl:allValuesFrom>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="hasEffect"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#AudioDatatypeFormat">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:hasValue rdf:resource="#Audio"/>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#hasDatatype"/>
        </owl:onProperty>
      </owl:Restriction>
```

```
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#MultimediaDatatypeForamt"/>
</owl:Class>
<owl:Class rdf:ID="VideoDatatypeFormat">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:hasValue rdf:resource="#Video"/>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#hasDatatype"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#MultimediaDatatypeForamt"/>
</owl:Class>
<owl:ObjectProperty rdf:about="#hasPrecondition">
  <rdfs:domain rdf:resource="#MultimediaAdaptationServices"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasEffect">
  <rdfs:domain rdf:resource="#MultimediaAdaptationServices"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasOutputType">
  <rdfs:domain rdf:resource="#MultimediaAdaptationServices"/>
  <rdfs:range rdf:resource="#MultimediaDatatype"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasDatatype">
  <rdfs:range rdf:resource="#MultimediaDatatype"/>
  <rdfs:domain rdf:resource="#MultimediaDatatypeForamt"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasServiceProvider">
  <rdfs:domain rdf:resource="#MultimediaAdaptationServices"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasServicePrice">
  <rdfs:domain rdf:resource="#MultimediaAdaptationServices"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasInputType">
  <rdfs:domain rdf:resource="#MultimediaAdaptationServices"/>
  <rdfs:range rdf:resource="#MultimediaDatatype"/>
</owl:ObjectProperty>
```

```
  <owl:DatatypeProperty rdf:ID="serviceDescription">
    <rdfs:domain rdf:resource="#MultimediaAdaptationServices"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="adaptationServiceName">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#MultimediaAdaptationServices"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="adaptationTime">
    <rdfs:domain rdf:resource="#MultimediaAdaptationServices"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#duration"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="additionalInformation">
    <rdfs:domain rdf:resource="#MultimediaDatatypeForamt"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="foramtName">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#MultimediaDatatypeForamt"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="#adaptedMediaQuality">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#MultimediaAdaptationServices"/>
  </owl:DatatypeProperty>
</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 1.3, Build 225.4)
http://protege.stanford.edu -->
```

Declaration

I, the undersigned, declare that this thesis is my original work and has not been presented for a degree in any other university, and that all source of materials used for the thesis have been duly acknowledged.

Declared by:

Name: _____

Signature: _____

Date: _____

Confirmed by advisor:

Name: _____

Signature: _____

Date: _____

Place and date of submission: Addis Ababa, July 2005.