A Microformatted Registry Alternative

Thomas R. Warne
Brigham Young University - Provo

A MICROFORMATTED REGISTRY ALTERNATIVE

by

Thomas R. Warne

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Computer Science Department

Brigham Young University

August 2006

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Thomas R. Warne

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date	Phillip J. Windley, Chair
Date	Kevin D. Seppi
Date	Eric G. Mercer

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Thomas R. Warne in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date	Phillip J. Windley Chair, Graduate Committee

Accepted for the Department

Parris K. Egbert
Graduate Coordinator

Accepted for the College

Thomas W. Sederberg
Associate Dean, College of Physical and Mathematical Sciences

ABSTRACT

A MICROFORMATTED REGISTRY ALTERNATIVE

Thomas R. Warne

Computer Science Department

Master of Science

To effectively use Web services, providers and consumers need to be connected by a registry. Several registry solutions exist today, including UDDI and WSIL. Also, many organizations simply use Web pages to list available Web services and their descriptions. This research describes a microformat for representing Web service description documents. These microformatted documents can be converted back to the original format for use by machines. They can also contain additional information, making them more useful to people. A registry, allowing indexing and searching of microformatted service descriptions, is also described. The benefits of this solution include: using existing standards; allowing customization of presentation for people; allowing

existing search tools to find descriptions; and providing for future, context-based searching to make locating Web services even easier.

ACKNOWLEDGMENTS

I am grateful for the wonderful support of my wife, Emily, through this entire degree process. I also wish to thank Dr. Windley for his encouragements and ideas that helped to shape this thesis.

Table of Contents

Table of Figures

List of Listings

Chapter 1 Introduction

Since the advent of the Internet, many different methods have been devised for disparate computers to communicate and collaborate. Among these, proprietary protocols, as well as protocols such as CORBA and RMI, have dominated the landscape for many years [Campbell 1999, Adamopoulos 1999, Lemahieu 2001]. Recently, there has been a move towards using Web services in order to achieve inter-organization collaboration [Kreger 2003].

Web services refer to services made available over the Internet in standard formats. Although the name has certain implications, standards related to Web services focus more on the formats of the messages than on the transport protocol (such as HTTP) that those messages use [Kreger 2003]. As Web services began to enable businesses to interact (both internally and externally) more easily, additional technologies and concepts were developed to ease using the Web services. The Web service registry is one of the most important of these concepts. The registry is also the focus of this thesis.

1.1 Registries

In the introduction to the UDDI Specification, Clement states, "Web services are meaningful only if potential users may find information sufficient to permit their execution" [Clement 2004]. Helping users to find this information is the purpose of Web service registries. Having a registry is important for several reasons. In some cases, it enables applications to move the service binding decision to runtime. Then as Web services come and go, or when the interface changes, the application can still use the new services. Also, a registry functions as a single point-of-reference for registered web services. Whenever updates are needed or desired, the registry is the place that developers and machines can go to find out about those updates. Since developers and machines can find out all of the relevant information about a web service in a registry, the registry makes web services more usable.

Several different implementations of registries exist today. These can be grouped into three categories. First, is Universal Description, Discovery and Integration (UDDI). Second is the Web Services Inspection Language (WSIL). Third is a broad grouping of ad-hoc solutions. The following sections will briefly introduce each registry. A more in-depth discussion of each registry is contained in Chapter 4.

1.1.1 UDDI

UDDI was proposed by Microsoft and IBM in 2000 and is now a standard administered by the OASIS group. The specification is currently in version 3.0 [Clement 2004]. UDDI is often described as having three different roles: discovery, description and integration [Bloomberg 2004]. For discovery, UDDI acts as the white pages, containing contact and basic information about businesses whose services are contained in the registry. The description aspect of UDDI refers to the registry being like the yellow pages, advertising offered services and classifying both the businesses and services. Finally, with respect to integration, UDDI can be like "green pages," containing technical information about web services. UDDI is widespread, though its complexity and bulk make it unsuitable for many organizations.

1.1.2 WSIL

In November 2001, IBM and Microsoft proposed a new technology called Web Services Inspection Language (WSIL or WS-Inspection). WSIL was meant to complement UDDI [Ballinger 2001]. Targeted to those organizations who either could not or did not want to use UDDI, WSIL allows providers to publish information about Web services on their own website in a simple XML document. WSIL documents contain location information about Web service descriptions and links to other WSIL documents. Unlike UDDI, WSIL documents do not describe the web service – they only indicate where the description can be found. Though a simple solution with a low entry barrier, WSIL is not widely used today.

1.1.3 Ad-hoc

Many organizations with widely used Web services do not publish their information in a formal registry. Instead, some organizations that provide publicly accessible web services, such as Amazon and Google, publish a human-readable API on their Web site describing the methods and endpoints available through their Web services [Google, Amazon]. Other organizations provide downloadable descriptions and code libraries so that people can use their services. These pseudo-registries fulfill part, but not all, of the duties of a Web services registry.

1.2 Problem Statement

Due to the shortcomings of each of these registry solutions, an alternative registry solution is needed. Regardless of the intended consumer, an organization that wants to allow others access to their Web services, whether internally, externally, or publicly, must provide some form of human documentation as well as the technical information needed to connect to the service. Combining new technologies with existing standards, this thesis will create an alternative registry that will allow organizations to publish information about their Web services in a way that is more convenient for both humans and computers. This alternative solution is named AMRA (A Microformatted Registry Alternative).

1.3 Thesis Statement

Many organizations, rather than using heavyweight UDDI, simply create lists of the Web services that are available. Often these lists are created in human readable form and are distributed around the organization's Web. Microformatted Web pages can be created that allow organizations to create human readable lists of Web services in a way that also allows machines to find and bind to services. These Web pages constitute a simple and novel method for machines and humans to find and use Web services.

1.4 Thesis Organization

The thesis will proceed as follows. First, background information will be presented that will lay the groundwork for the remainder of the thesis. Next, will be a review related research and present additional information on the shortcomings of UDDI, WSIL and ad-hoc registries. Following this in-depth discussion of the three competing solutions, the new registry format will be discussed. The demonstration of the new format's viability will immediately precede the conclusions that can be drawn from this work.

Chapter 2 Background

To provide context for the rest of the discussion, this chapter will introduce the key technologies that will be used and discussed in the completion of this thesis. These technologies include various XML-based technologies, microformats, WSIL and WSDL.

2.1 XML

Extensible Markup Language (XML) is a flexible, structured markup language. Of itself, XML has no elements, or tags. Rather, XML defines how documents should be formatted so that they can be parsed in a standard manner. XML tags are simple text strings. As a result, XML documents are often said to be self-describing. This can be misleading, however. Although tags are useful for humans to read and decipher using context clues, a computer cannot decipher the meaning of a tag. Between documents (and sometimes within a document) a given tag can mean more than one thing. To a computer the tags are merely identifiers, uniquely identifiable by name and position.

2.2 XSD

In order to provide some meaning to XML documents, XML schema (XSD) can be provided. A schema defines how elements (tags) and attributes can be combined to create a valid XML document. Schemas are useful so that software can be written which can use XML documents from any source as long as they validate against a certain schema. A schema document is also valid XML.

While a schema document might describe the structure of a valid XML document, it cannot provide any computer-usable information about the semantics of the format. It can only be used to describe how the document should be built so that others can read it. This does not minimize the importance of schema documents. Rather, it provides the basis on which semantic interpretation of XML documents can proceed. Without a common, documented structure, it would be impossible for a computer to interpret an XML document.

2.3 XHTML

A very important schema is the XHTML schema. XHTML 1.0 is similar to HTML, with some very specific differences. The most important difference is that XHTML is valid XML, whereas HTML is not. This allows XHTML to be manipulated and used in many of the same ways as XML. Modern browsers support the XHTML standard.

XHTML is not just valid XML, it is a specific form of XML, conforming to a specific schema. Since it conforms to a well-known schema, a valid XHTML document can be displayed in a Web browser. The browser has been written to understand and display XHTML, though it cannot meaningfully display an arbitrary XML document.

2.4 XSLT

Another important XML-based technology is XML Stylesheets (XSLT). XSLT documents describe a transformation from one XML format to another. One common use of stylesheets is to transform an XML document conforming to a certain schema into XHTML. Since XSLT is also valid XML, described by a schema, it too can be the subject or result of a transformation, as can an XSD (XML Schema) document. Related to XSLT is XPath. XPath provides a sort of query and path specification language for use both within and without XSLT. Version 1.0 of both of these specifications has been integrated into the major browsers and programming environments. Version 2.0, released in November, 2005, builds on the foundations of 1.0 and adds far greater functionality [Kay 2005]. Support for version 2.0 of XSLT and XPath is currently limited to a few implementations, limiting its usefulness.

2.5 Microformats

There are many different ways of combining the preceding XML-based technologies. An important emerging method is to use microformats. A microformat is the name given to data formats that adhere to certain principles. These principles guide the development of microformats [microformats A]. Microformats should:

- Solve a specific problem
- Start as simple as possible

- Design for humans first, machines second
- Reuse building blocks from widely adopted standards
- Embrace Modularity/embeddability
- Enable and encourage decentralized development, content, services

The goal of microformats is to look at existing uses of information and adapt it so that it is more usable by both humans and machines, with human consumption receiving the primary focus. Microformats are not concerned with developing new ways of doing things, but rather codifying the existing methods. This has been referred to in the microformats community as "paving the cow-paths" [Celik 2005a]. In other words, rather than creating a new standard and trying to get people to follow, developers of microformats attempt to look at what people are currently doing and to see if it is possible to make it easier for everyone to do that same thing. At the same time, it is important to realize that microformats do not, and should not, apply to every problem. Instead, the problem at hand should be analyzed for the applicability of microformats.

Microformats are strongly related to XHTML. This is due in large part to the focus on presentation to people. By using XHTML, the information can be easily formatted for display. The information can also be used by a computer, since XHTML is valid XML.

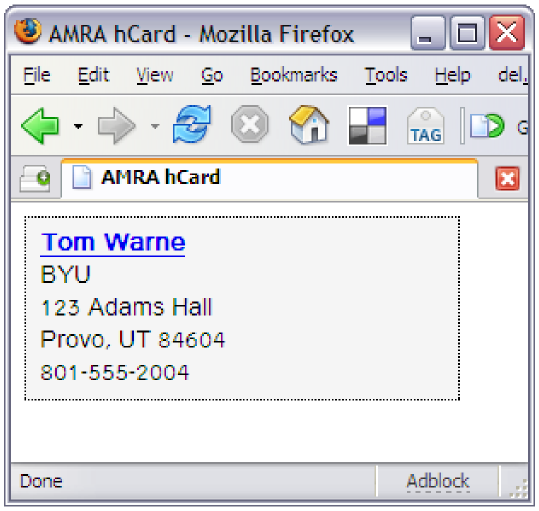
Several microformats standards have been proposed. These include hCard and hCalendar, XOXO, VoteLinks, and rel-tag. The hCard and hCalendar standards define the representation of vCard [Dawson 1998a] and iCal [Dawson 1998b] information in XHTML. For example, the hCard listed in Listing 1 represents the vCard listed in Listing 2. Figure 1 shows the microformat as rendered in a Web browser.

```
        <div class="vcard">
         <a class="url fn" href="http://www.cs.byu.edu/">Tom Warne</a>
         <div class="org">BYU</div>
         <div class="adr">
          <span class="locality">Provo</span>,
          <span class="region">UT</span>
          <span class="postal-code">84602</span>
         </div>
         <div class="tel">801-555-1234</div>
        </div>
```

Listing 1: hCard Example

```
        BEGIN:VCARD
        PRODID:-//suda.co.uk//X2V 0.6.26
(BETA)//EN
        SOURCE:
        NAME:
        VERSION:3.0
        N;CHARSET=UTF-8:Warne;Tom;;;;
        FN;CHARSET=UTF-8:Tom Warne
        ORG:BYU
        ADR;CHARSET=UTF-8:;;;Provo;UT;84602;;
        TEL:801-555-1234
        URL:http://www.cs.byu.edu/
        END:VCARD
```

Listing 2: vCard Example


AMRA hCard - Mozilla Firefox
File Edit View Go Bookmarks Tools Help del.
TAG
AMRA hCard
Done
Adblock

Tom Warne
BYU
123 Adams Hall
Provo, UT 84604
801-555-2004

Figure 1: Rendered hCard

Some items of interest with respect to this thesis are the use of the XHMTL class attribute to define the meaning of the various XHTML tags. Also, the authors of this standard strove to use semantically appropriate tags. In other words, rather than use a <p> or a <br> or any of a number of other tags, they decided to use the <div> and <span> tags. These provide the necessary styling flexibility without attaching too much erroneous information about the meaning of the tag. From the point of view of semantics, though the <p> tag has no meaning in and of itself, the developer of the browser has caused the browser to interpret that <p> in a certain way. Using that same interpretation with the microformat would attach incorrect semantics to the microformat.

2.6 WSDL

Web Services Description Language (WSDL) documents are perhaps one of the most important pieces of a Web services deployment today. WSDL documents are provided to describe a multitude of Web services that are both publicly and privately available. Though not required since it has its own description format, UDDI registries can point to WSDL descriptions. WSIL registries are specifically targeted towards pointing to WSDL files as descriptors. Online services, such as Google, provide a WSDL file for simple access to their Web services. Perhaps most importantly, developers who create Web services often use WSDL files for describing them.

In a Web services environment, the description document is the common link that ties the various pieces together. The service provider publishes the service's description in the registry and the consumer uses the registry to locate the description document. While this description may be in any format, most registries provide access to WSDL documents for the description.

Once a consumer has located the WSDL document, many tools are available to create a binding to the Web service described in the document. For example, Apache Axis includes tools to generate Java data types and method stubs to access Web services through Java. Similarly, Microsoft's Visual Studio can generate bindings that allow programmatic access to Web services through C#. The wide availability of these tools make WSDL documents incredibly valuable. A developer can easily use any Web service that has a valid WSDL description.

Chapter 3 Related Work

This thesis focuses on two primary technology areas. The first is Web services. The second is microformats. This chapter will discuss external work related to each of these areas.

3.1 Web services

Semantic Web [Narayanan 2002] is an increasingly popular attempt to imbue Web pages with semantic information so that machines can make use of the information. Although the goals may be similar, this thesis is very different. The semantic Web relies on additional formats, such as RDF and OWL, to give meaning to the data for both people and machines. This thesis seeks to merge an existing machine-readable data format (a Web service description) and an existing human-readable format (XHTML). This thesis makes no evaluation on the merits of the semantic Web; it is outside of the scope of this research.

Some Web sites, most notably Xmethods.com, have created common Web-based repositories of Web services. These sites provide listings of the services and allow the services to be searched. These types of sites are a combination of the centralized UDDI (as descriptions are hosted by the aggregator) and ad hoc Web-based registries. Although these are valid and useful services, this thesis attempts to break away from the centralization notion. This thesis also tries to provide a standard way of defining the relationship between human- and machine-readable descriptions.

3.2 Microformats

As discussed previously, several standards exist for microformats. The XOXO microformat is used centrally in the AMRA solution and will be discussed in depth later on. Two other microformats in particular also seem to relate to the topic of this thesis. These are the REST Web services microformat and the search results microformat.

The REST microformat is design work that has been done with a microformat for REST Web services [microformats C]. The goal of the design work is to create a specification for simplifying access to RESTful services in XHTML, by making the services more standardized. Though this is an interesting problem, this thesis is focusing on providing simplified access to Web service descriptions in a microformatted registry. RESTful Web services that conform to the eventual result of the microformat community's work in this regard could be included, along with any other web service, in this thesis' AMRA registry.

The search results microformat was designed to codify XHTML for search results [microformats B]. Though the registry proposed by this thesis includes a search component that returns search results, this component is merely a proof-of-concept and does not include this search results microformat. A production implementation could incorporate the search results microformat.

Chapter 4 Shortcomings of Existing Registries

UDDI, WSIL and ad-hoc solutions represent the most common implementations of Web service registries. Each of these, however, has flaws. Beginning with UDDI, followed by WSIL and finally ad-hoc solutions, this chapter will present the shortcomings of each of these registries along with some background information on each.

4.1 UDDI

The Universal Description, Discovery and Integration standard is the primary industry standard for registry services. With initial development and backing from Microsoft and IBM in 2000, UDDI has now gained the support of numerous companies (such as Infravio and Systinet). Many enterprise server solutions (including J2EE application servers like Jboss, BEA WebLogic, and IBM WebSphere as well as Microsoft's Windows Server 2003) include an implementation of UDDI for publishing information about Web services contained or provided by that server.

The UDDI standard consists of a set of data structures and services. The data structures allow description and categorization of providers and services. For each provider, contact information, related providers, and categorization information is kept. For each service, the associated provider, technical access information, and categorization information is kept. UDDI provides services that allow search and retrieval of providers and services.

As the primary standard for Web services registries, UDDI has support from many companies. Although support is high, the barriers to using UDDI are also high. This section will discuss three main problems with UDDI that raise the barriers to use.

4.1.1 Information Centralization

Originally, UDDI was envisioned as a centralized registry [Bloomberg 2004]. Providers would register their services and business information with common registries run by Microsoft, IBM, and others. Those who wished to use a service could go to just one place to find information on services and providers from across the industry. These registries would help to fulfill the vision of Web services – namely businesses easily seamlessly interoperating.

However, the Internet is not well suited to centralized control. Instead, each organization maintains control over its own data and information. Whether out of a desire to maintain control over their own data or for other reasons, businesses did not support centralized registries and instead set up internal UDDI registries [Mimoso 2004, Rodgers 2003]. However, these internal registries are difficult to setup and use. They are also very expensive. For organizations with relatively few Web services, deploying a full scale UDDI server is excessive [Rodgers 2003].

4.1.2 Human Readability

As noted above, UDDI provides services to access Web service descriptions and provider information. These services are not human readable. Complex XML files are required as input and similarly complex XML-based results are returned. Many implementations do provide a user interface to ease use of these services.

4.1.3 Aggregation

The original UDDI proposal called for unified registries. Such a scheme made searching for a Web service simple – since all of the services were listed in one place. With individual and disparate registries, though, searching became much more difficult. Each registry must be located individually and queried for information on its services. However, this is rarely done. Instead, providers resort to advertising Web services on Web sites or through word of mouth.

4.2 WSIL

The Web Services Inspection Language (WSIL or WS-Inspection) proposal [Ballinger 2001] provides a lightweight alternative to UDDI. WSIL documents use a simple yet extensible XML format to describe the location of Web service descriptions. Although UDDI provides extensive information about a Web service, including provider information and service classification information, WSIL focuses only on providing the location of the description of the Web service. Version 1 of the WSIL standard [Ballinger 2001] includes two means of locating description documents. The first is through a simple URL. The second is through an existing UDDI registry. The specification also allows additional future service description and repository types to be included, with no need to modify the specification [Appnel 2002].

The WSIL specification was released in November, 2001. In that time, few tools have been created for it and no major success stories have come to light. It seems that many people do not understand the relationship between UDDI and WSIL, due to the number of articles written attempting to demystify it [Appnel 2002, Modi 2002, Nagy 2001]. The perception that WSIL is merely an inferior, less full featured alternative to UDDI may be hurting adoption. Lack of standardization, human readability and aggregation is also damaging to WSIL's adoption rates.

A Web service provider that wants to use WSIL simply creates a hierarchy of WSIL documents on their Web server. These documents contain both references to service descriptions and links to other WSIL documents. The service description references are primarily Web-accessible WSDL documents or UDDI service keys. A simple WSIL file is shown in Listing 3. Two services are described. The first contains two descriptions, one from a WSDL file, the other from a UDDI registry. The second service is unnamed, and includes only a

WSDL description. Finally, there is a link to another WSIL document.

```
      <inspection xmlns="http://schemas.xmlsoap.org/ws/2001/10/inspection/"

xmlns:wsiluddi="http://schemas.xmlsoap.org/ws/2001/10/inspection/uddi/">
        <service>
          <abstract>A stock quote service with two descriptions</abstract>
          <name>Stocks</name>
          <description
            referencedNamespace="http://schemas.xmlsoap.org/wsdl/"
            location="http://example.com/stockquote.wsdl" />
          <description referencedNamespace="urn:uddi-org:api">
            <wsiluddi:serviceDescription
              location="http://www.example.com/uddi/inquiryapi">
              <wsiluddi:serviceKey>
                4FA28580-5C39-11D5-9FCF-BB3200333F79
              </wsiluddi:serviceKey>
            </wsiluddi:serviceDescription>
          </description>
        </service>
        <service>
          <description
            referencedNamespace="http://schemas.xmlsoap.org/wsdl/"
            location="ftp://anotherexample.com/tools/calculator.wsdl" />
        </service>
        <link

 referencedNamespace="http://schemas.xmlsoap.org/ws/2001/10/inspection/"
          location="http://example.com/moreservices.wsil" />
      </inspection>
```

Listing 3: Sample WSIL

Part of the allure of using WSIL as an alternative to a Web service registry is the ability to access the documents over the Web. Although this is an important feature, WSIL has other problems that are difficult to work around. The following sections describe these problems.

4.2.1 Meta Information

In addition to the technical specification of a service, additional information is often needed. This meta data about the service, such as the service provider's contact information, is as important to the human consumer of the registry information as the technical information is to the machine using the service. Such information may be included in the description tag of a WSIL service block, but the specification does not dictate inclusion of such information.

4.2.2 Human Readability

Web service registries should provide human-readable documentation. Though XML files such as WSDL and WSIL documents use meaningful tag names, such files are meant primarily for machine consumption. A person who is unfamiliar with the format would have a hard time understanding the relationships expressed in the document. WSIL relies solely on XML files. The human-readability of WSIL is low.

4.2.3 Standardization

WSIL is meant to be a distributed solution. As with more traditional Web content, WSIL allows each provider to maintains its own WSIL documents on its Web server. One major difference between the two types of content is that starting documents use standard names on the Web, easing document discovery. For example, while allowing for some variation based on platform, if a request is made to http://www.example.com/, that request will return the index.html document located on the server. No such standard exists for WSIL. Appnel notes that using the same index naming convention would be good [Appnel 2002]. However, the specification does not require that. Due to the lack of standardization, locating a WSIL document is difficult.

4.3 Ad-hoc

The final category of Web service registry is the ad-hoc registry. This diverse collection of solutions includes many alternatives. Each alternative attempts to solve the same problem – providing a Web service consumer with the information he needs to use the service.

An oft-repeated pattern in ad-hoc Web service registries is for a Web service provider (such as Google or Amazon) to provide Web service information on a Web site. This Web site often includes human readable documentation. The Web site also often contains links to access the WSDL as well as bindings for various programming languages.

Another common solution is to use a Web service framework such as Apache Axis [Apache Software Foundation]. Axis includes functionality to list each Web service provided within the framework. This list is on a simple, dynamically generated Web page. Service description information, in the form of a WSDL document, can also be obtained by appending "?wsdl" to the end of the service address.

These various ad-hoc Web service registry solutions are common place because they are simple to set up and maintain. As with WSIL and UDDI, these solutions suffer from problems that impact their overall usability.

4.3.1 Standardization

Because of their very nature, ad-hoc solutions are not standardized. Although there is some standardization within a solution, such as with Apache Axis, there is no standardization between solutions. This causes several difficulties. Since each solution is implemented differently, there can be no common discovery, as discussed in the next section. Human consumers must adapt to the differences in each implementation. People are good at adapting; machines are not. Due to the lack of standardization, machines are unable to use ad-hoc registry solutions.

4.3.2 Aggregation

A major side effect of the lack of standardization is the inability of a third party to aggregate service information. Information presented in ad-hoc registries is usually Web based. Standard Web content aggregators, such as Google or Yahoo, can index this information for searching. However, the aggregator knows only that the ad-hoc registry is presenting Web pages, and so they are indexed as such. There is no way to identify search results as containing Web service description information. Consumers are forced to locate Web services by contacting each provider individually.

4.4 Summary

UDDI, WSIL and ad-hoc registry solutions each have problems that make them difficult to use. To summarize, these problems include:

1. Information centralization: Web service providers should be able to maintain control over their own service information.
2. Supporting Information: Web service consumers need additional, non-technical information about a Web service.
3. Human-readability: Humans must be able to read and understand the Web service description as easily as machines.
4. Aggregation: Web service descriptions should be available for aggregation, similar to Web searching.
5. Standardization: Web service registries should follow standards to allow for simpler, more uniform access.

In order to solve these problems, a new solution is required. The following chapters will discuss the proposal of this thesis, A Microformatted Registry Alternative (AMRA), a microformat-based solution.

Chapter 5 AMRA Architecture and Overview

The technologies and tools described in the preceding sections can be combined to create a powerful new form of registry, based on microformatted documents. The name of this registry is "A Microformatted Registry Alternative" (AMRA). The AMRA registry addresses the five main shortcomings of other registry solutions. It is a simple and novel solution for a web service registry.

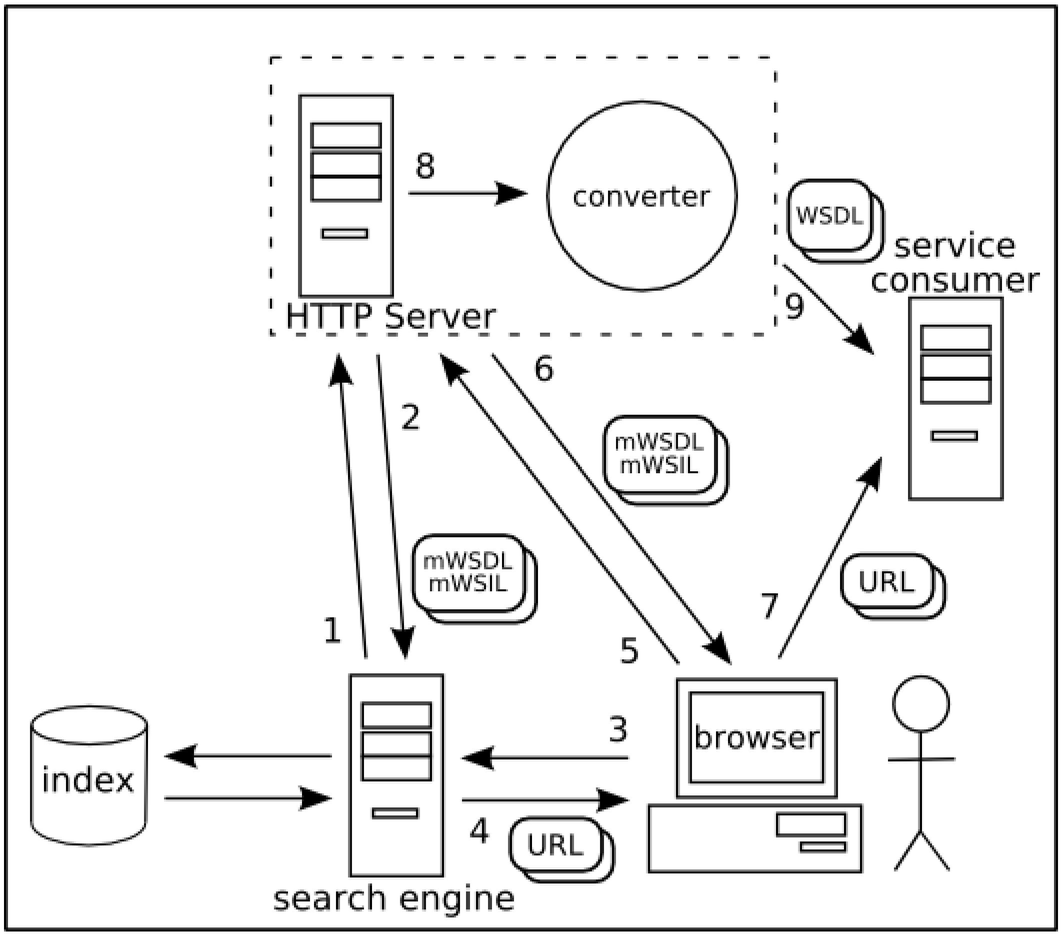


Figure 2: AMRA Overview

An overview of the AMRA registry is depicted in Figure 2. The registry functions as follows:

1. The search engine, which may be a general search engine or a specific Web service search engine, crawls Web servers to find microformatted documents.

2. The Web server responds with the mWSDL and mWSIL documents, which are then indexed by the search engine.

3. A user can query the search engine to find information about existing Web services. The query may include specific technical information about the service as well as information on the provider and any other information included in the microformatted description.

4. The search engine responds with a list of results matching the user's criteria. The results include a link to the human-readable microformatted description residing on the Web server.

5. The user can request the microformatted description from the Web server.

6. The Web server provides the human-readable service description. The user can use this document to determine whether the service fulfills his needs or not.

7. The URL provided in the search results refers both to the human-readable description and the machine readable description, since they are the same document. This URL is provided to the service consumer to

locate the machine readable description.

8. When the service consumer requests the service description, the Web server converts the microformatted version into the machine-readable version before returning it.

9. The service consumer receives the standard, machine-readable service description which it can easily integrate to use the described Web service.

The flexibility and power of this AMRA registry solution will be discussed and demonstrated in the following chapters. Chapter 6 outlines the structure of the microformats, mWSDL and mWSIL, and discusses their flexibility. Chapter 7 discusses the converter and generator and Chapter 8 reviews the sample search engine. Finally in Chapter 9 a demonstration is described that shows that the registry has overcome the shortcomings noted above.

Chapter 6 AMRA Microformat

The AMRA microformat allows WSDL and WSIL documents to be contained in standard XHTML. A microformatted WSDL document is referred to as an mWSDL and a microformatted WSIL document is referred to as an mWSIL. The microformat allows a lossless transformation of the information contained in the WSDL and WSIL documents. Many parts of this microformat are common; those that are specific to either source format will be noted. After talking briefly about the basis for this microformat, the specific components that make up the microformat will be reviewed, followed by a discussion of additional information that may be contained in the microformat.

6.1 Base Microformat

The AMRA Microformat is an extension of the XOXO microformat [Celik 2004]. The XOXO specification describes a format for outlined or hierarchical data. These hierarchies are contained in XHTML lists (<ul> or <ol>). Each element in the list is represented by an XHTML list entry (<li>) element, which may contain another list. The XOXO microformat also notes that attributes may be added to a list entry by means of a definition list (<dl>). The AMRA microformat takes advantage of these features.

6.2 Microformat Structure

This section introduces the structure of the microformat. The important components of this microformat are the root, elements, text elements, attributes, namespaces, and class names. Each of these components will be discussed in the following sections.

6.2.1 Root

A document that is formatted according to the AMRA microformat will begin with the top level XOXO-classed list element. Inside of this list element are one or more document representations. Each document representation consists of a list entry element, with one of the class entries as the name of the top level element in the source format, with the name modified as specified below. For a WSIL document, the class will be "amra_wsil_inspection." For a WSDL document, the class will be "amra_wsdl_inspection."

6.2.2 Elements

Each element, including the root, will be formatted as follows. The element will be formatted as a list entry (<li>) element. The class of the list entry element will include the prefixed name of the element, formatted as specified below. If the element has any attributes, they must be included in a definition list (<dl>) inside the list entry element, with contents as defined below. Any namespace declarations that occur on this element (not those declared in a parent) must also be included in this definition list, as specified below. After the attribute definition list, a list (<ul> or <ol>) element is provided to include any children elements. This list does not have any AMRA-specific class.

6.2.3 Text elements

For elements whose content is defined to be a single text node, a single list entry element will be present as a child of the list (<ul> or <ol>) element in the parent element. This single list entry element will have the value of the text node for its content. The single list entry element will also have a class that contains the string "amra_text."

6.2.4 Attributes

Any attributes that are present on an element must be included as children of the definition list (<dl>) element. Each attribute should be represented by a definition term/definition (<dt>/<dd>) pair. The definition term (<dt>) element should contain a human-readable form of the name of the attribute. No AMRA-specific class is attached to the definition term element. The definition term element is not used when converting the document back to the source format. The definition element should contain the unmodified value of the attribute. The definition element should also have a class which contains the prefixed name of the attribute, formatted as specified below.

6.2.5 Namespaces

Any namespaces that are declared on the element must be represented in the attribute definition list. To indicate that namespaces are present, a definition term/definition pair will mark the beginning of the namespaces. The definition term in this pair can include any text, and no AMRA-specific class is required. The definition element must include the class "amrans" to indicate that the contents of the definition will be namespace information. The contents of the definition element must include a definition list (<dl>) element. The definition list element does not need any AMRA-specific class. The definition list element must contain a definition term/definition pair for each namespace declared on the element. The definition term (<dt>) element may include any markup representing the human-readable form of the namespace prefix. Note that this should usually just be the prefix, as various portions of the document may refer specifically to these namespaces. Providing the prefix in human-readable form will allow users to identify the referenced namespace. The definition element must contain the unmodified value of the namespace. The definition element must also include a class that contains the prefix of the namespace, formatted as specified below.

6.2.6 Class names

The various class names should be formatted as follows. The class names for microformatted elements and attributes should consist of three pieces, separated by underscores. Note that this does not restrict the use of the underscore character in element or attribute names. The first piece is the string "amra." The second piece is the declared prefix of the namespace of the element or attribute. When an element or attribute is in the default namespace, this second piece will be empty. The third piece is the local name of the element or attribute.

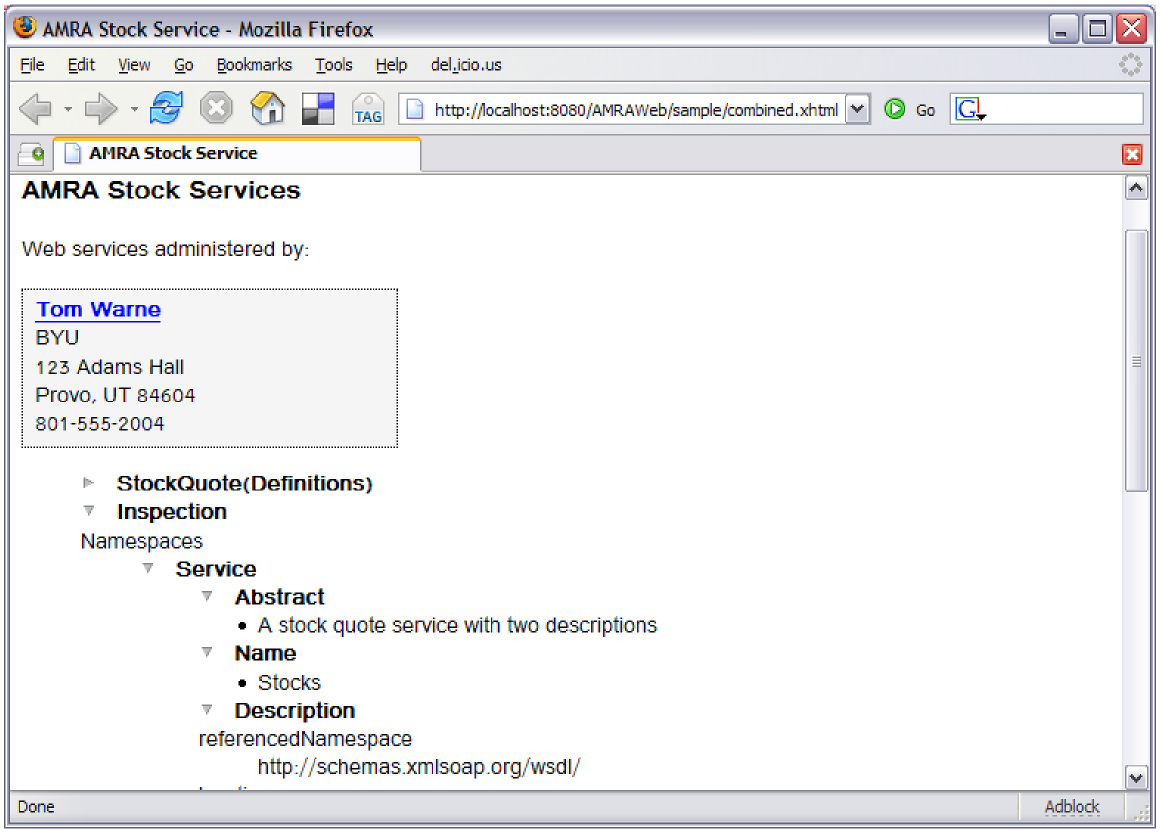
The class names for microformatted namespace declarations consist of two pieces, separated by an underscore. The first piece is the string "amrans." The second piece is the prefix of the namespace that is being declared.

6.3 Additional information

Additional information can be included in the microformatted document. This information is not required for the microformat. It is used to enhance the experience of a person using the microformatted document. Except for certain places specified above (such as the content of a definition attribute), additional markup can be included in any element. For example, a human-meaningful name can be placed at the beginning of any list entry element. Hyperlinks can be included to link to resources referenced in the microformat. Additional classes (e.g. for styling) can be used on any element in the microformat in addition to the classes defined by this microformat.

As the standard for describing Web services, WSDL provides the information that a computer needs in order to use the service. Humans can use additional information to give context to the service and to make decisions about its use. Since standard WSDL documents must conform to the schema, additional information cannot be added easily. In an mWSDL or mWSIL document, though, such information can be included. It can also be formatted (using XHTML) in a way that makes sense for human, rather than machine, consumption. This chapter explores some additional information that may be added to microformatted Web service descriptions.

As information about Web services is aggregated and made searchable, an important piece of information is the identity of the service provider. This information may be used to collect information on services provided by a certain organization. It may also be used to judge the reliability of the service and its


AMRA Stock Service - Mozilla Firefox
File Edit View Go Bookmarks Tools Help del.icio.us
http://localhost:8080/AMRAWeb/sample/combined.xhtml
Go
AMRA Stock Service
Done
Adblock

AMRA Stock Services

Web services administered by:

Tom Warne
BYU
123 Adams Hall
Provo, UT 84604
801-555-2004

StockQuote(Definitions)
Inspection
Namespaces
Service
Abstract
- A stock quote service with two descriptions
Name
- Stocks
Description
referencedNamespace
http://schemas.xmlsoap.org/wsdl/

Figure 3: Microformatted Description with hCard

results. Though this information is not included in a WSDL or WSIL document, the hCard microformat may be used to embed this information directly into the microformatted document, as shown in Figure 3.

The hCard microformat specification describes a means of encoding vCard information in XHTML. Standard transforms exist to return this information to the vCard format. A vCard contains essential contact information and is a good fit for partnering with microformatted descriptions.

For a person reading a microformatted service description, such as the one in Figure 3, the hCard is immediately identifiable on an XHTML page. Styling can enhance the hCard appearance. Many existing computer systems that may be able to take advantage of a converted microformatted Web service description will not be able to take advantage of this additional piece of information. Providing the information for human consumption alone would be sufficient justification for inclusion on the page. New systems, though, may be made aware that this information is available. Then they may take advantage of it by filtering available services by known providers, or automatically updating contact information for a service that is being used. Uses such as these increase the value of the description – it is no longer simply a technical document, but includes information related to the organization behind the service.

6.4 Examples

See Appendix A for a sample WSDL document and the corresponding mWSDL document. Appendix B contains a sample WSIL document and the corresponding mWSIL document. The final section in each of these appendices shows a side-by-side comparison of the source format and the microformat.

Chapter 7 Converter and Generator

The converter and generator are tools used to generate microformatted documents and to convert microformatted documents back to their source format. Two factors have motivated the creation of these tools.

First, the microformatted documents described in the previous chapter are geared towards human readability. Since they use standard XHTML, human readability is high. All of the information required by machines is contained in these documents – machines are tuned to read and understand the source formats, though. The converter is necessary so that legacy software can continue to use Web service descriptions contained in this new registry.

The second motivating factor is the vast pool of existing WSDL documents. Since it is the primary standard for describing Web services, almost any organization that provides Web services uses WSDL documents. In addition, many tools exist to create WSDL documents. Though equivalent mWSDL documents could be created by hand, a generator eases the process of converting existing descriptions into the new format.

7.1 Background

Due to their similar structures, the converter and generator were largely developed in parallel. Initially, these tools were implemented using Java and the Document Object Model (DOM), but some problems appeared. It may have been possible to resolve the problems through a careful examination and optimization of the code, but instead the needs and purposes of the conversion system were reexamined and the decision was made to use XML Stylesheets.

7.2 XSLT-based Implementation

XML Stylesheets (XSLT) simply transform XML documents from one format to another. Since the documents to be converted and generated are XHTML and WSIL or WSDL, the use of XSLT is a natural fit.

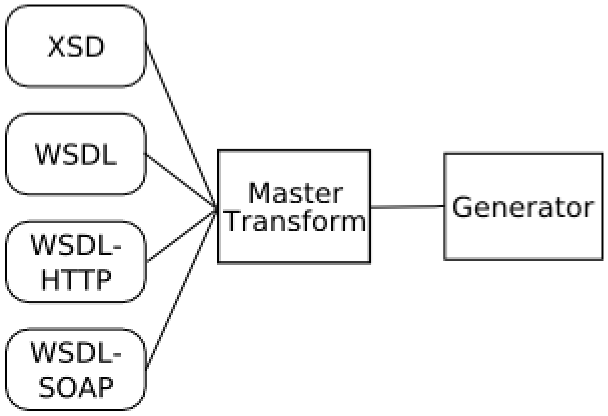


Figure 4: Master Transform for Generator

The structure of a microformat generator and converter is very repetitive – each element and attribute are handled the same way. The structure of the generator or converter is necessarily based on the format specified in the relevant XML Schema. XML Schema itself is also described by an XML Schema document. The converter and generator are themselves generated from an XSLT transform that transforms the various XML Schema files required for a given format into the set of transforms that would actually work on the format itself. In Figure 4, the schemas, on the left, that contribute to the WSDL generator are shown. A master transform converts those schema documents into the transform that is used as the mWSDL generator. The same is done to create a converter to convert from the microformat back to the original format. Since the converter and generator are both XSLT stylesheets, they can be used from any service that provides XSLT processing support. This section will first discuss the generator created by this process, followed by the converter.

7.2.1 Generator

This section discusses the generator, though much of the discussion is also relevant to the converter. The next section will review those pieces that are specific to the converter.

The generator is based on the elements defined in the schema. For each element, the generator creates an XSLT template to match the element and output the correct XHTML to create a list element (<li>) in the output. The XHTML class of this list element is based on the name of the element, as described in Section 6.2.5. The generated template then calls a generic template to convert any attributes that exist on the element into the appropriate definition term/definition form. It also converts any namespace declarations on the element to the appropriate form. After generating this simple list element, the template instructs the XSLT processor to continue applying additional templates. The transform that is created from the XML Schema is very repetitive. A few common templates are referenced throughout the rest of the templates. The XSLT comprising the mWSIL generator is contained in Appendix C.

Although the XML Schema document is used to create the generator transform, there is no reliance on the specific structure of the various types defined in the XML Schema. Instead, the responsibility falls to the document maintainer to ensure that the source format document conforms to the schema. This is similar to standard tools requiring that documents conform to existing schemas.

7.2.2 Converter

The transform to convert back to the original format is even simpler than the generator. Each element in the schema is represented, and each merely outputs an element with the correct name and namespace and then calls a generic template to add any attributes contained in definition term/definition elements into the element. The processor is then requested to continue applying templates. These simple transformations allow a lossless conversion to occur between the original format and the microformat. The XSLT comprising the WSIL converter is contained in Appendix D.

As with the generator, the converter performs no validation. Any attribute represented in the microformatted document is added to the resulting source format document, regardless of validity. External validation tools must be used to ensure that erroneous documents are not used.

7.3 Converter Service

The converter transformation may be used to manually convert microformatted documents into source formats. It may also be used by other tools to allow microformatted descriptions to be converted and used as source format descriptions. Within the registry Web application, the converter is exposed as a service. Any URL may be passed to this service. If a valid microformatted document is contained at that location, the corresponding source document is returned.

Chapter 8 Search Engine

The power of any registry lies in its ability to help people and machines find the services that they are looking for. As simple XHTML documents that can be converted back to their source format, microformatted WSDL and WSIL documents are immediately more usable than as XML documents. To further explore the usability of these formats, a Java-based Web application was created to allow the documents to be crawled, indexed, and searched – to function as a true Web services registry.

The application described here is merely a demonstration of using microformatted documents as a registry. Though usable, this application was created to facilitate this research. Ideally, these registry services of crawling, indexing and searching will be provided by services such as Google or Yahoo. Such services could be made to recognize properly formatted mWSDL and mWSIL documents and create a global registry, as they currently do with other types of information, such as directory and product data [Google B, Yahoo]. As will be discussed later, one of the advantages of using mWSDL and mWSIL is their ability to be used with standard web tools and online services immediately.

8.1 Overview

The registry is structured as a Java-based Web application using JSPs and servlets. See Appendix E for a list of the specific technologies and tools used.

The three main pieces of this registry, comprising the "Search Engine" shown in Figure 2, are the crawler, indexer and searcher. These components are discussed in the following sections. An example of using these components is contained in Chapter 9.

8.2 Crawler

The purpose of the crawler is to locate microformatted WSDL and WSIL documents on the Internet or on an intranet. Although more powerful general purpose crawlers are available (and may well be suitable), this crawler fulfills its intended function – finding documents to be indexed. Even though the crawler is meant only as a proof-of-concept application, the implementation strives to be a good net citizen. The crawler observes standard restrictions, such as the robots.txt restrictions, and keeps track of the URLs that have been crawled to prevent endless crawling of cyclical references.

The crawler presents a simple Web-based interface. The user can enter the URL of any XHTML document. The crawler retrieves this document and extracts any microformatted WSDL or WSIL documents contained in the page. It also locates an hCard instance, if present. Each of these documents is converted to the source format (using the transforms described above) and presented for indexing as described in the next section. The crawler also follows any links that might be specified in the WSIL document and adds those documents to the index.

8.3 Indexer

The indexer is an important part of the overall strategy for providing a registry service. The purpose of the indexer is to extract relevant information from the documents and add it to the document index. The current implementation of the registry uses the open source Lucene text search engine, though not as a general Web search engine. Instead, since a registry is primarily concerned with helping users to locate services, the indexer focuses on services. Both WSDL and WSIL documents have portions that describe a service. For each service description that is found (in either format), the indexer adds the following information to the index:

1. Source URL – The URL of the document that contained the service description.
2. Retrieval Date – The date that the document was retrieved.
3. Name(s) – The names that are provided for the service.
4. Description(s) – The descriptions (or abstracts for WSIL) of the service.
5. Source document type – The type of the converted source document (WSIL or WSDL).
6. Source format index on page – The index of the service source

document within the set of source documents of the same type (WSIL or WSDL)

7. vCard name – Value of the full name field of the first hCard appearing on the page.

8. vCard organization – Value of the org field on the first hCard appearing on the page.

Since the indexer is aware of the formats being indexed, these specific pieces of information can be extracted. Indexing this set of information for each document allows meaningful queries to be performed by the searcher described in the next section.

8.4 Searcher

Since the Web application uses the Lucene search engine, searching the indexed documents is relatively simple. A simple Web-based interface is provided that allows the user to enter queries. Lucene has an advanced query parser that allows complex searches to be performed. Once the search has been performed, the results are formatted for display to the user. Each result is scored by the search engine. The search results page presents the following information to the user:

1. Name of the service

2. Score

3. Description
4. Source document type (WSDL or WSIL)
5. Link to the original document (microformatted)
6. Link to the converted source document
7. Name and organization of the service provider

This set of information allows the user to make simple decisions about the suitability of certain services (based on the description or provider, for example). It also provides the user with resources, such as the location of the microformatted original document, to find more information about the service (still in a human readable format). Finally, it provides the user with the location of the technical information that is needed to use external tools for integration with the Web service. The user of the registry can quickly find the information he needs to use the Web services that he wants.

Chapter 9 Demonstration

The microformatted registry is a simple way for humans and machines to find and use Web services. To verify this statement, a simple experiment was performed. The registry described above was used to index WSDL documents from several sources. A search was performed to find a desired service. The URL of the converted description was used in Microsoft Visual Studio 2005 to create a binding to the Web service found in the registry. The bound application is able to use the Web service. Each of these steps is described in the following sections.

9.1 Populating the search index

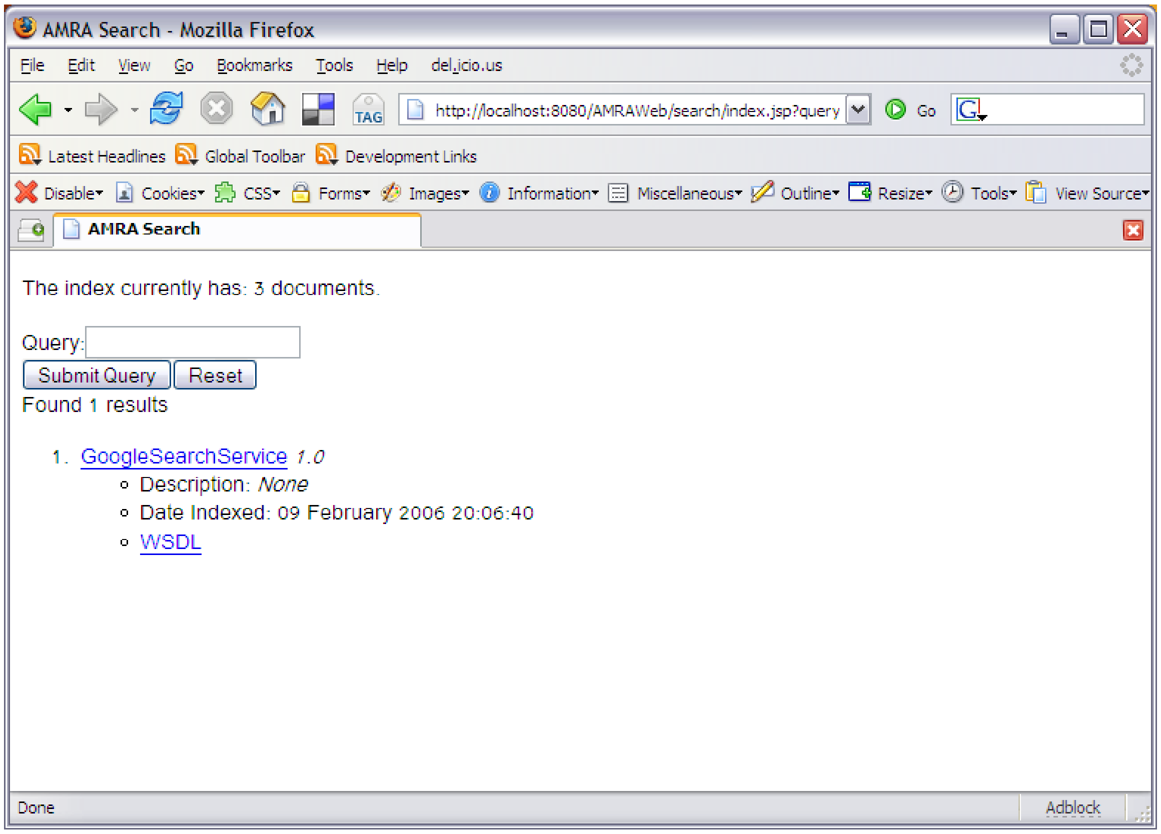
A registry is of no use if there are no Web services listed in it. To fill the experimental registry, several WSDL documents were found on the Internet. The WSDLs used were: Google Search API [Google A]; Amazon's AlexaTopSites service [Amazon]; and Invesbot's CompanySearch [Invesbot]. These WSDLs describe real services that may be found and used today. None of these documents was microformatted, so the generator was used to create microformatted versions of them. As would be done by an organization using this microformat, the human readable names were updated to be meaningful and an hCard, containing the actual provider's information, was added to each page. These microformatted descriptions were placed on a locally hosted Web server.

Although each organization would normally host their own Web service descriptions, co-hosting the descriptions does not change the use or functionality of the microformatted registry. From the point of view of the registry, these descriptions could just as easily be on disparate servers.

Once the descriptions were microformatted, updated and locally hosted, each local URL was entered into the crawler, which added the documents to the index. The indexed documents were then ready to be searched. This test focused on the use of WSDL documents within the AMRA microformat registry.

9.2 Searching the Index

The Google Web search service was used for testing. For this service, Google provides a download package containing the WSDL and sample code for various programming languages. The Google Web Search service provides various RPC-style calls that allow a user to perform Web searches over SOAP. Specifically, the service provides three operations: doGetCachedPage, doSpellingSuggestion, and doGoogleSearch. In the query field on the search screen, the term "Google*" was entered to search for all services related to Google. As expected, the search service returned one result. The result, shown in Figure 5, listed the name of the service as a hyperlink to the microformatted entry, the score, retrieval date, and a link to the original format WSDL (converted on-the-fly from the microformatted version).


AMRA Search - Mozilla Firefox
File Edit View Go Bookmarks Tools Help del.icio.us
http://localhost:8080/AMRAWeb/search/index.jsp?query
Go
Latest Headlines
Global Toolbar
Development Links
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source
AMRA Search
Query:
Submit Query
Reset
Done
Adblock

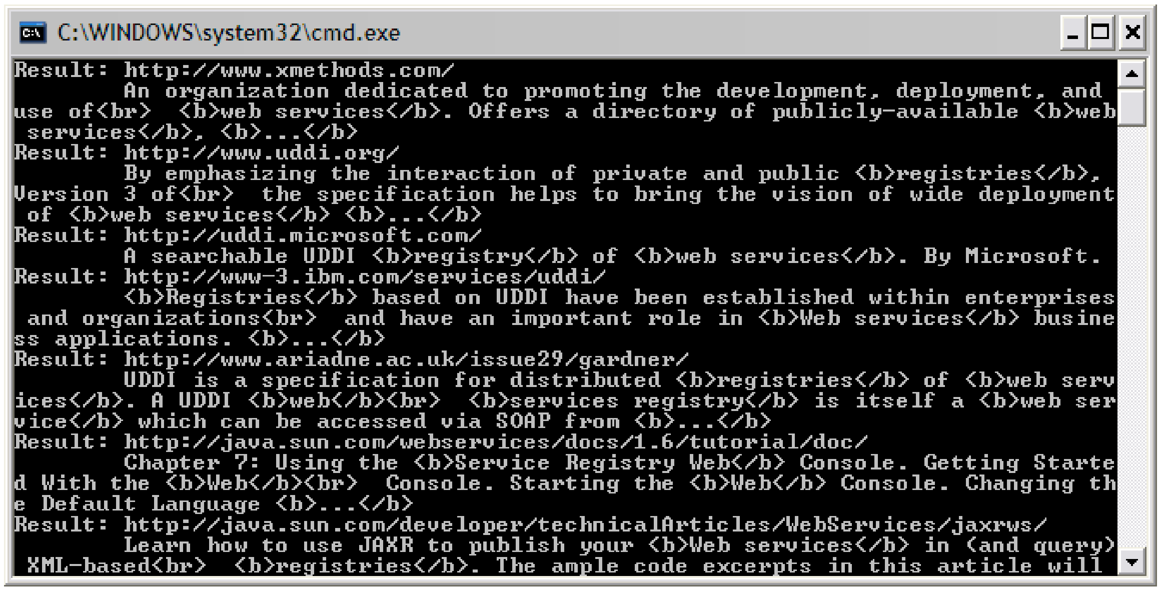
The index currently has: 3 documents.

Found 1 results

1. GoogleSearchService *1.0*
   - Description: *None*
   - Date Indexed: 09 February 2006 20:06:40
   - WSDL

Figure 5: Search Results

9.3 Integration



```
Result: http://www.xmethods.com/
        An organization dedicated to promoting the development, deployment, and
use of<br>  <b>web services</b>. Offers a directory of publicly-available <b>web
 services</b>, <b>...</b>
Result: http://www.uddi.org/
        By emphasizing the interaction of private and public <b>registries</b>,
Version 3 of<br>  the specification helps to bring the vision of wide deployment
 of <b>web services</b> <b>...</b>
Result: http://uddi.microsoft.com/
        A searchable UDDI <b>registry</b> of <b>web services</b>. By Microsoft.
Result: http://www-3.ibm.com/services/uddi/
        <b>Registries</b> based on UDDI have been established within enterprises
 and organizations<br>  and have an important role in <b>Web services</b> busine
ss applications. <b>...</b>
Result: http://www.ariadne.ac.uk/issue29/gardner/
        UDDI is a specification for distributed <b>registries</b> of <b>web serv
ices</b>. A UDDI <b>web</b><br>  <b>services registry</b> is itself a <b>web ser
vice</b> which can be accessed via SOAP from <b>...</b>
Result: http://java.sun.com/webservices/docs/1.6/tutorial/doc/
        Chapter 7: Using the <b>Service Registry Web</b> Console. Getting Starte
d With the <b>Web</b><br>  Console. Starting the <b>Web</b> Console. Changing th
e Default Language <b>...</b>
Result: http://java.sun.com/developer/technicalArticles/WebServices/jaxrws/
        Learn how to use JAXR to publish your <b>Web services</b> in (and query)
 XML-based<br>  <b>registries</b>. The ample code excerpts in this article will
```

Figure 6: Results of the running the integration application

A registry provides access to service descriptions so that the services described may be integrated into external applications. A simple integration application was created to use the result found by the registry search. The goal of the application was to search the Web for the string “Web services registry.” Using Microsoft Visual Studio 2005, a new C# project was created. A new Web Reference was added to the project. Using the URL from the WSDL link in the search results, which points to the converted version of the microformatted WSDL, the address of the description was entered. Using that dynamically converted WSDL, Visual Studio was able to generate the necessary bindings to allow the program to use the discovered Web service. The Web service was then used to search for the string, “Web services registry.” The results are shown in Figure 6.

Chapter 10 Analysis of AMRA

AMRA is a viable alternative to UDDI, WSIL and ad-hoc registries. This chapter summarizes the advantages of a microformatted registry and then specifically addresses the five shortcomings listed above.

10.1 Advantages of a Microformat

Using microformats offers several advantages over other solutions. In particular, microformats increase the human understandability of service descriptions. They also lend themselves to fitting established patterns. These two specific advantages will be discussed in this section.

10.1.1 Readability

One powerful advantage that microformatted WSIL and WSDL documents have is the inclusion of additional information. Although the original format documents contain all of the information that a computer needs (and, frankly, can handle), a human can benefit from the inclusion of additional information. For example, rather than relying on the simple service name allowed by the WSDL specification, a more expressive, human readable name can be maintained in the microformatted document. The original name is still available (and visible in the microformatted document), but additional information is provided to the people who read the document.

In a WSIL document, both links and descriptions reference external resources. Often, these resources are accessible through an HTTP interface. For these resources, it is trivial to add a hyperlink to the resource. Once again, this information is specifically for the humans reading the document. The conversion back to the original format will only include those parts that are necessary; it will not include parts that have been extrapolated.

Using these microformatted documents has a distinct advantage over merely providing a stylesheet to present a WSDL or WSIL document. The conversion from the original format document to the microformatted document and back again is lossless. However, often the conversion from the microformatted document to the original format document and back again will be lossy. This is due to the inclusion of additional information in the microformatted document. This information was not present in the original format document and, in fact, there is no place for it in that document. A microformatted document, then, provides great added benefits to the humans, while maintaining the current level of support required by machines.

10.1.2 Patterns

When using the Web today, people rely on certain patterns. They expect that they will be able to find human-readable information about whatever it is they are looking for. They also expect to be able to search to find what they are looking for. If they know the provider of the information, they should be able to search for it directly on the provider's site. If they do not care who the provider is, they should be able to use an Internet-wide search engine, such as Google or Yahoo to find the information that they seek.

The AMRA Microformat allows the Web service registry to fit this same, established pattern. Organizations that list their Web services using this microformat can easily support internal indexing and searching of those services. Since these are microformatted documents, the searching extends beyond just indexing a Web page and returning relevant results. An AMRA-aware search engine, as was shown above, can take advantage of its knowledge of the format to provide better, more meaningful searches and to return more appropriate information (such as a link to the converted document).

To summarize, the advantages of a microformatted registry are:

- Continues to use standard description formats
- Builds on existing Web paradigms such as Web search
- Enables more powerful, content-specific searches
- Includes additional, human-centric information
- Layout and style of descriptions can be customized according to user's needs without affecting machine readability

10.2 Resolution of Shortcomings

After analyzing each of the competing registry solutions, Chapter 4 noted five major shortcomings. These shortcomings were:

1. Information Centralization
2. Supporting Information
3. Human Readability
4. Aggregation
5. Standardization

The previous chapters described the AMRA registry. This registry overcomes each of these shortcomings. This section will address each shortcoming and how AMRA overcomes it.

10.2.1 Information Centralization and Aggregation

Businesses on the Internet desire to maintain control of their own information. Centralization of Web service information runs counter to the notion of distributed control. AMRA allows each provider to retain all descriptions. Access to those descriptions is available only through the provider's Web site.

Seemingly contradictory to a desire to stay away from information centralization is the desire to aggregate Web service information to make it globally searchable. In this regard, AMRA follows the model provided by the general Web. Provider information, such as Web service descriptions and contact information, is maintained on the provider's Web server. Since this information is Web-accessible and in a standard format and location, third parties can easily aggregate the information. Just as Google provides a single source for searching all Web sites, another tool could be used to search all service descriptions. Thus, providers do not need to centralize their information, but aggregation is still possible.

10.2.2 Supporting Information

Standard service descriptions, such as WSDL, contain just enough information for a machine to be able to use the Web service. Additional information is not, and cannot be, included in the description. For a person, information regarding the service provider is just as important as the technical information. Using microformatted documents, AMRA allows additional information to be included in the service description. Using the converter, this information is removed when returning the document to its source format for use by a machine. Though this thesis only described including provider contact information, any other information desired by a provider could be included. Those tools that know how to take advantage of it can; those that do not can ignore it.

10.2.3 Human Readability

As noted in the previous section, people must be able to read service descriptions just as much as machines need to. Though WSDL and WSIL documents are XML and use descriptive tags, they are not easily human readable. XHTML, on the other hand, was designed expressly to present information to people. Formatting, hyperlinking, and dynamic display all increase the readability and usability of a document on the Web. AMRA uses microformatted service descriptions. These documents, in mWSDL and mWSIL format, are XHTML. Cascading Style Sheets (CSS) can be easily and naturally used to change the look and feel of the human readable description without affecting the machine's ability to read the converted document. Javascript can also be added to allow more user friendly interaction with service descriptions. Being microformatted, these documents take advantage of the power of XHTML for presenting information to people. AMRA is human readable.

10.2.4 Standardization

In some respects, AMRA is similar to ad-hoc solutions. Ad-hoc solutions, as noted in Chapter 4, suffer from a lack of standardization. AMRA, on the other hand, uses microformatted documents so a standard is in place at the level that standards are important – for inter-machine communication. Formatting and display of the documents can be customized by the provider. The underlying microformat is unaffected by such customization and so can continue to be used automatically by machines. Standardization is present within AMRA, allowing powerful third party uses of the Web service information.

10.2.5 Summary

The registries reviewed in Chapter 4 each suffered from shortcomings. The AMRA registry overcomes each of these problems through using microformatted documents. This registry is a powerful, yet simple approach to the registry problem.

Chapter 11 Conclusion and Future Work

For Web services to be useful, they must be described and discovered. In any organization, this is done using a registry. A Microformatted Registry Alternative presents an alternative to UDDI, WSIL and ad-hoc registry solutions. This chapter describes areas of future, related research and presents a summary of AMRA.

11.1 Future Directions

Several additional areas should be investigated in relation to this work. Additional microformats could be incorporated into the microformatted Web page. For example, standards exist for microformats that tag pages. Such microformat tags could be useful for categorizing microformatted service descriptions. These categories could become another criteria for searching against microformatted serviced descriptions. Another standardized microformat allows authors of Web pages to indicate a vote for or against a certain linked Web page. A Web crawler could determine the value of a Web service by the votes for or against the microformatted service description.

Research is also needed to investigate the general nature of the AMRA microformat. The microformat was designed generally and could technically be applied to any XML document. What document types would benefit from being made accessible in this way? Can general transforms be created to handle any type of document? The benefits to Web service descriptions are numerous. Such benefits may be available to other types of documents also.

11.2 Summary

A Microformatted Registry Alternative presents a novel approach to the problem of providing Web service description information. The solution combines microformatted documents with converters, generators and indexers to create a simple and powerful registry.

Using the AMRA microformat, Web service description documents can be formatted for display to people. These same documents can then be converted back to the original format so that machines can continue to use them. The AMRA registry solution was built around these microformats.

The relationship between the machine-readable service description and the human-meaningful Web page is clearly established. In accordance with the principles of microformats, the description is made useful for people first, and then machines. In other words, design paths were taken that made the description more accessible to people and care was taken to ensure that the conversion remained lossless. The process of obtaining a valid service description (WSDL) from a microformatted description is simple and standard, using the standard conversion transform.

Using a microformatted service description allows additional information to be included for those to whom it means something – people. Although computers cannot accept or use more information than is in a standard description, people can handle more information. Oftentimes, more information is required to make complete sense of a document. For example, the registry solution described in this document adds provider information (in the form of an hCard). Rather than simply a description, the microformatted description allows a person to find the provider of the Web service in a simple and easily identifiable way. Since the microformat contains all of the technical information that is needed to consume the Web service, as well as additional information useful to people, it is more valuable than a standard description.

Web services are not generally useful unless they can be found. Service descriptions formatted according to the AMRA microformat are Web pages in standard XHTML. Because of this, they can, as can other Web-based descriptors, be indexed and searched using standard Web search tools. Microformatted descriptions have the added advantage of using a standard format for describing Web services. General Web search tools can be made to be aware of this format and can present these pages specifically to users searching for Web services.

As a result of this thesis, the community has proposals for WSDL and WSIL microformats, as well as sample code to convert these microformats into the original formats. They also have transforms capable of generating microformatted Web Service descriptions and a crawler that is able to index the microformatted documents. By combining these tools, as was demonstrated, individuals and organizations will be able to publish human and machine readable descriptions of their Web services quickly and easily and consumers will be able to find and use Web services more easily.

Appendix A Microformatted WSDL

A.1 mWSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/1999/xhtml"
  xmlns:wsil="http://schemas.xmlsoap.org/ws/2001/10/inspection/"
  xmlns:wsiluddi="http://schemas.xmlsoap.org/ws/2001/10/inspection/uddi/">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>AMRA</title>
</head>
<body>
<ul class="xoxo">
  <xhtml:li xmlns:xhtml="http://www.w3.org/1999/xhtml"
    class="amra_wsdl_definitions">
    <xhtml:span xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
class="element_name">Definitions</xhtml:span>
    <xhtml:dl xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
      xmlns:xs="http://www.w3.org/2001/XMLSchema">
      <dt>name</dt>
      <dd class="amra__name">StockQuote</dd>
      <dt>targetNamespace</dt>
      <dd class="amra__targetNamespace">http://example.com/stockquote.wsdl</dd>
      <dt>Namespaces</dt>
      <dd class="amrans">
      <dl>
        <dt>xsd1</dt>
        <dd class="amrans_xsd1">http://example.com/stockquote.xsd</dd>
        <dt>xml</dt>
        <dd class="amrans_xml">http://www.w3.org/XML/1998/namespace</dd>
        <dt>soap</dt>
        <dd class="amrans_soap">http://schemas.xmlsoap.org/wsdl/soap/</dd>
        <dt>Default</dt>
        <dd class="amrans_Default">http://schemas.xmlsoap.org/wsdl/</dd>
        <dt>tns</dt>
        <dd class="amrans_tns">http://example.com/stockquote.wsdl</dd>
      </dl>
      </dd>
    </xhtml:dl>
    <xhtml:ul xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
      xmlns:xs="http://www.w3.org/2001/XMLSchema">
      <xhtml:li class="amra_wsdl_types">
        <xhtml:span class="element_name">Types</xhtml:span>
        <xhtml:dl></xhtml:dl>
        <xhtml:ul>
          <xhtml:li class="amra_xs_schema">
            <xhtml:span
              xmlns:hfp="http://www.w3.org/2001/XMLSchema-hasFacetAndProperty"
              class="element_name">Schema</xhtml:span>
            <xhtml:dl
              xmlns:hfp="http://www.w3.org/2001/XMLSchema-hasFacetAndProperty">
              <dt>targetNamespace</dt>
              <dd
class="amra__targetNamespace">http://example.com/stockquote.xsd</dd>
            </xhtml:dl>
            <xhtml:ul
              xmlns:hfp="http://www.w3.org/2001/XMLSchema-hasFacetAndProperty">
              <xhtml:li class="amra_xs_element">
                <xhtml:span class="element_name">Element</xhtml:span>
                <xhtml:dl>
                  <dt>name</dt>
                  <dd class="amra__name">TradePriceRequest</dd>
                </xhtml:dl>
                <xhtml:ul>
```

```
                        <xhtml:li class="amra_xs_complexType">
                          <xhtml:span class="element_name">ComplexType</xhtml:span>
                          <xhtml:dl></xhtml:dl>
                          <xhtml:ul>
                            <xhtml:li class="amra_xs_all">
                              <xhtml:span class="element_name">All</xhtml:span>
                              <xhtml:dl></xhtml:dl>
                              <xhtml:ul>
                                     <xhtml:li class="amra_xs_element">
                                            <xhtml:span
class="element_name">Element</xhtml:span>
                                            <xhtml:dl>
                                                   <dt>name</dt>
                                                   <dd
class="amra__name">tickerSymbol</dd>
                                                   <dt>type</dt>
                                                   <dd class="amra__type">string</dd>
                                            </xhtml:dl>
                                            <xhtml:ul></xhtml:ul>
                                     </xhtml:li>
                              </xhtml:ul>
                            </xhtml:li>
                          </xhtml:ul>
                        </xhtml:li>
                      </xhtml:ul>
                    </xhtml:li>
                    <xhtml:li class="amra_xs_element">
                      <xhtml:span class="element_name">Element</xhtml:span>
                      <xhtml:dl>
                        <dt>name</dt>
                        <dd class="amra__name">TradePrice</dd>
                      </xhtml:dl>
                      <xhtml:ul>
                        <xhtml:li class="amra_xs_complexType">
                          <xhtml:span class="element_name">ComplexType</xhtml:span>
                          <xhtml:dl></xhtml:dl>
                          <xhtml:ul>
                            <xhtml:li class="amra_xs_all">
                              <xhtml:span class="element_name">All</xhtml:span>
                              <xhtml:dl></xhtml:dl>
                              <xhtml:ul>
                                     <xhtml:li class="amra_xs_element">
                                            <xhtml:span
class="element_name">Element</xhtml:span>
                                            <xhtml:dl>
                                                   <dt>name</dt>
                                                   <dd class="amra__name">price</dd>
                                                   <dt>type</dt>
                                                   <dd class="amra__type">float</dd>
                                            </xhtml:dl>
                                            <xhtml:ul></xhtml:ul>
                                     </xhtml:li>
                              </xhtml:ul>
                            </xhtml:li>
                          </xhtml:ul>
                        </xhtml:li>
                      </xhtml:ul>
                    </xhtml:li>
                  </xhtml:ul>
                </xhtml:li>
              </xhtml:ul>
            </xhtml:li>
            <xhtml:li class="amra_wsdl_message">
              <xhtml:span class="element_name">Message</xhtml:span>
              <xhtml:dl>
                <dt>name</dt>
                <dd class="amra__name">GetLastTradePriceInput</dd>
              </xhtml:dl>
              <xhtml:ul>
```

```
    <xhtml:li class="amra_wsdl_part">
      <xhtml:span class="element_name">Part</xhtml:span>
      <xhtml:dl>
        <dt>name</dt>
        <dd class="amra__name">body</dd>
        <dt>element</dt>
        <dd class="amra__element">xsd1:TradePriceRequest</dd>
      </xhtml:dl>
      <xhtml:ul></xhtml:ul>
    </xhtml:li>
  </xhtml:ul>
</xhtml:li>
<xhtml:li class="amra_wsdl_message">
  <xhtml:span class="element_name">Message</xhtml:span>
  <xhtml:dl>
    <dt>name</dt>
    <dd class="amra__name">GetLastTradePriceOutput</dd>
  </xhtml:dl>
  <xhtml:ul>
    <xhtml:li class="amra_wsdl_part">
      <xhtml:span class="element_name">Part</xhtml:span>
      <xhtml:dl>
        <dt>name</dt>
        <dd class="amra__name">body</dd>
        <dt>element</dt>
        <dd class="amra__element">xsd1:TradePrice</dd>
      </xhtml:dl>
      <xhtml:ul></xhtml:ul>
    </xhtml:li>
  </xhtml:ul>
</xhtml:li>
<xhtml:li class="amra_wsdl_portType">
  <xhtml:span class="element_name">PortType</xhtml:span>
  <xhtml:dl>
    <dt>name</dt>
    <dd class="amra__name">StockQuotePortType</dd>
  </xhtml:dl>
  <xhtml:ul>
    <xhtml:li class="amra_wsdl_operation">
      <xhtml:span class="element_name">Operation</xhtml:span>
      <xhtml:dl>
        <dt>name</dt>
        <dd class="amra__name">GetLastTradePrice</dd>
      </xhtml:dl>
      <xhtml:ul>
        <xhtml:li class="amra_wsdl_input">
          <xhtml:span class="element_name">Input</xhtml:span>
          <xhtml:dl>
            <dt>message</dt>
            <dd class="amra__message">tns:GetLastTradePriceInput</dd>
          </xhtml:dl>
          <xhtml:ul></xhtml:ul>
        </xhtml:li>
        <xhtml:li class="amra_wsdl_output">
          <xhtml:span class="element_name">Output</xhtml:span>
          <xhtml:dl>
            <dt>message</dt>
            <dd class="amra__message">tns:GetLastTradePriceOutput</dd>
          </xhtml:dl>
          <xhtml:ul></xhtml:ul>
        </xhtml:li>
      </xhtml:ul>
    </xhtml:li>
  </xhtml:ul>
</xhtml:li>
<xhtml:li class="amra_wsdl_binding">
  <xhtml:span class="element_name">Binding</xhtml:span>
  <xhtml:dl>
    <dt>name</dt>
```

```
                <dd class="amra__name">StockQuoteSoapBinding</dd>
                <dt>type</dt>
                <dd class="amra__type">tns:StockQuotePortType</dd>
              </xhtml:dl>
              <xhtml:ul>
                <xhtml:li class="amra_soap_binding">
                  <xhtml:span xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
                    class="element_name">Binding</xhtml:span>
                  <xhtml:dl xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
                    <dt>style</dt>
                    <dd class="amra__style">document</dd>
                    <dt>transport</dt>
                    <dd class="amra__transport">http://schemas.xmlsoap.org/soap/http</dd>
                  </xhtml:dl>
                  <xhtml:ul
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"></xhtml:ul>
                </xhtml:li>
                <xhtml:li class="amra_wsdl_operation">
                  <xhtml:span class="element_name">Operation</xhtml:span>
                  <xhtml:dl>
                    <dt>name</dt>
                    <dd class="amra__name">GetLastTradePrice</dd>
                  </xhtml:dl>
                  <xhtml:ul>
                    <xhtml:li class="amra_soap_operation">
                      <xhtml:span xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
                        class="element_name">Operation</xhtml:span>
                      <xhtml:dl xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
                        <dt>soapAction</dt>
                        <dd
class="amra__soapAction">http://example.com/GetLastTradePrice</dd>
                      </xhtml:dl>
                      <xhtml:ul
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"></xhtml:ul>
                    </xhtml:li>
                    <xhtml:li class="amra_wsdl_input">
                      <xhtml:span class="element_name">Input</xhtml:span>
                      <xhtml:dl></xhtml:dl>
                      <xhtml:ul>
                        <xhtml:li class="amra_soap_body">
                          <xhtml:span xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
                            class="element_name">Body</xhtml:span>
                          <xhtml:dl xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
                            <dt>use</dt>
                            <dd class="amra__use">literal</dd>
                          </xhtml:dl>
                          <xhtml:ul
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"></xhtml:ul>
                        </xhtml:li>
                      </xhtml:ul>
                    </xhtml:li>
                    <xhtml:li class="amra_wsdl_output">
                      <xhtml:span class="element_name">Output</xhtml:span>
                      <xhtml:dl></xhtml:dl>
                      <xhtml:ul>
                        <xhtml:li class="amra_soap_body">
                          <xhtml:span xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
                            class="element_name">Body</xhtml:span>
                          <xhtml:dl xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
                            <dt>use</dt>
                            <dd class="amra__use">literal</dd>
                          </xhtml:dl>
                          <xhtml:ul
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"></xhtml:ul>
                        </xhtml:li>
                      </xhtml:ul>
                    </xhtml:li>
                  </xhtml:ul>
                </xhtml:li>
```

```
        </xhtml:ul>
      </xhtml:li>
      <xhtml:li class="amra_wsdl_service">
        <xhtml:span class="element_name">Service</xhtml:span>
        <xhtml:dl>
          <dt>name</dt>
          <dd class="amra__name">StockQuoteService</dd>
        </xhtml:dl>
        <xhtml:ul>
          <xhtml:li class="amra_wsdl_documentation">
            <xhtml:span class="element_name">Documentation</xhtml:span>
            <xhtml:dl></xhtml:dl>
            <xhtml:ul>
              <li class="text">My first service</li>
            </xhtml:ul>
          </xhtml:li>
          <xhtml:li class="amra_wsdl_port">
            <xhtml:span class="element_name">Port</xhtml:span>
            <xhtml:dl>
              <dt>name</dt>
              <dd class="amra__name">StockQuotePort</dd>
              <dt>binding</dt>
              <dd class="amra__binding">tns:StockQuoteSoapBinding</dd>
            </xhtml:dl>
            <xhtml:ul>
              <xhtml:li class="amra_soap_address">
                <xhtml:span xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
                  class="element_name">Address</xhtml:span>
                <xhtml:dl xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
                  <dt>location</dt>
                  <dd class="amra__location">http://example.com/stockquote</dd>
                </xhtml:dl>
                <xhtml:ul
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"></xhtml:ul>
              </xhtml:li>
            </xhtml:ul>
          </xhtml:li>
        </xhtml:ul>
      </xhtml:li>
    </xhtml:ul>
  </xhtml:li>
</ul>
</body>
</html>
```

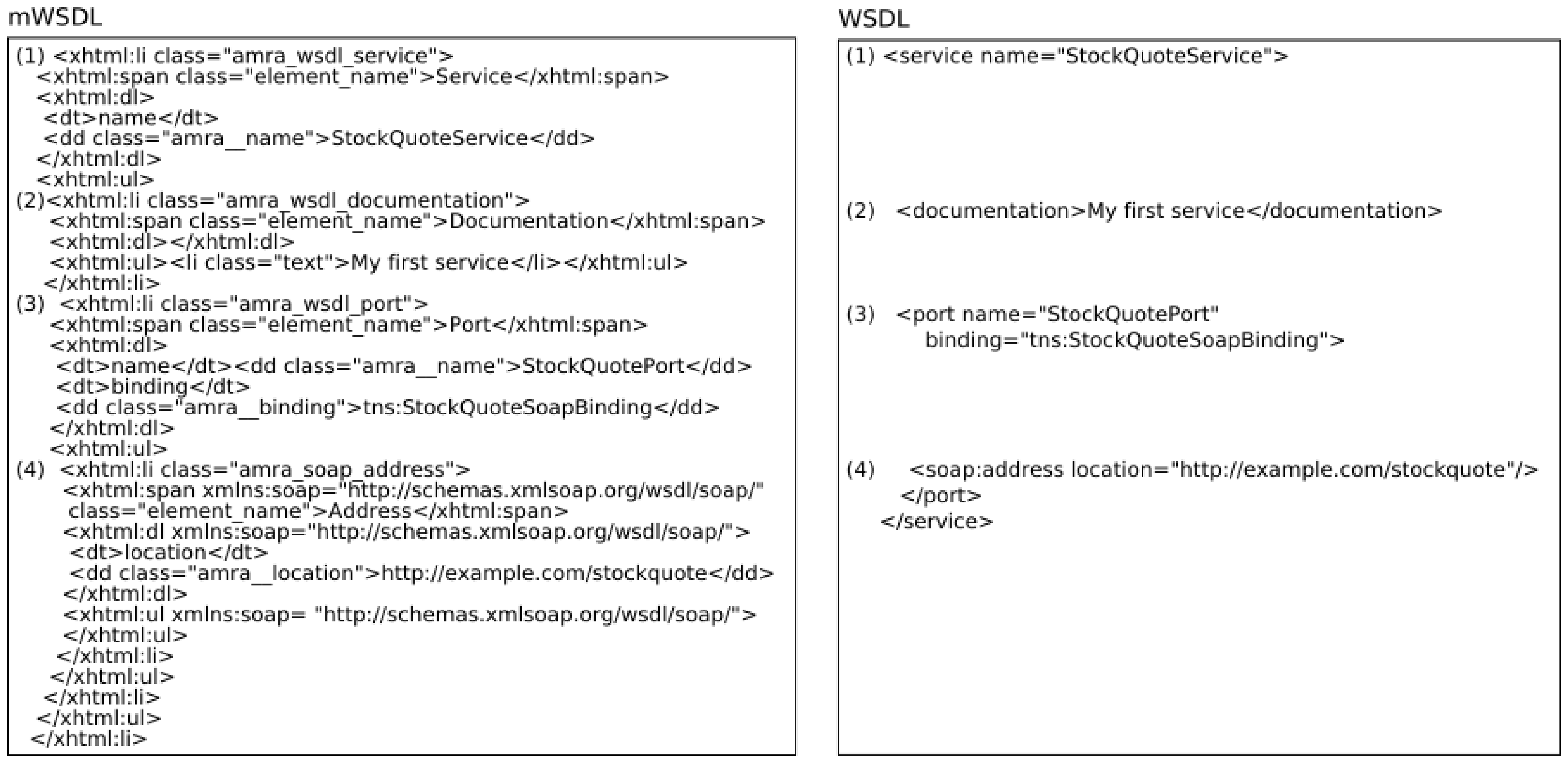
A.2 WSDL

```
<definitions name="StockQuote"
             targetNamespace="http://example.com/stockquote.wsdl"
             xmlns:tns="http://example.com/stockquote.wsdl"
             xmlns:xsd1="http://example.com/stockquote.xsd"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
             xmlns="http://schemas.xmlsoap.org/wsdl/">

  <types>
    <schema targetNamespace="http://example.com/stockquote.xsd"
           xmlns="http://www.w3.org/2001/XMLSchema">
      <element name="TradePriceRequest">
        <complexType>
          <all>
            <element name="tickerSymbol" type="string"/>
          </all>
        </complexType>
      </element>
      <element name="TradePrice">
         <complexType>
           <all>
```

```
                    <element name="price" type="float"/>
                  </all>
                </complexType>
             </element>
           </schema>
         </types>

         <message name="GetLastTradePriceInput">
           <part name="body" element="xsd1:TradePriceRequest"/>
         </message>

         <message name="GetLastTradePriceOutput">
           <part name="body" element="xsd1:TradePrice"/>
         </message>

         <portType name="StockQuotePortType">
           <operation name="GetLastTradePrice">
             <input message="tns:GetLastTradePriceInput"/>
             <output message="tns:GetLastTradePriceOutput"/>
           </operation>
         </portType>

         <binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
           <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
           <operation name="GetLastTradePrice">
             <soap:operation soapAction="http://example.com/GetLastTradePrice"/>
             <input>
               <soap:body use="literal"/>
             </input>
             <output>
               <soap:body use="literal"/>
             </output>
           </operation>
         </binding>

         <service name="StockQuoteService">
           <documentation>My first service</documentation>
           <port name="StockQuotePort" binding="tns:StockQuoteSoapBinding">
             <soap:address location="http://example.com/stockquote"/>
           </port>
         </service>

       </definitions>
```

A.3 Correspondence

This diagram shows the correspondence between the WSDL and the mWSDL.



mWSDL

```
(1) <xhtml:li class="amra_wsdl_service">
   <xhtml:span class="element_name">Service</xhtml:span>
   <xhtml:dl>
    <dt>name</dt>
    <dd class="amra__name">StockQuoteService</dd>
   </xhtml:dl>
   <xhtml:ul>
(2)<xhtml:li class="amra_wsdl_documentation">
     <xhtml:span class="element_name">Documentation</xhtml:span>
     <xhtml:dl></xhtml:dl>
     <xhtml:ul><li class="text">My first service</li></xhtml:ul>
    </xhtml:li>
(3)  <xhtml:li class="amra_wsdl_port">
     <xhtml:span class="element_name">Port</xhtml:span>
     <xhtml:dl>
      <dt>name</dt><dd class="amra__name">StockQuotePort</dd>
      <dt>binding</dt>
      <dd class="amra__binding">tns:StockQuoteSoapBinding</dd>
     </xhtml:dl>
     <xhtml:ul>
(4)  <xhtml:li class="amra_soap_address">
       <xhtml:span xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        class="element_name">Address</xhtml:span>
       <xhtml:dl xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
        <dt>location</dt>
        <dd class="amra__location">http://example.com/stockquote</dd>
       </xhtml:dl>
       <xhtml:ul xmlns:soap= "http://schemas.xmlsoap.org/wsdl/soap/">
       </xhtml:ul>
      </xhtml:li>
     </xhtml:ul>
    </xhtml:li>
   </xhtml:ul>
  </xhtml:li>
```

WSDL

```
(1) <service name="StockQuoteService">

(2)   <documentation>My first service</documentation>

(3)   <port name="StockQuotePort"
          binding="tns:StockQuoteSoapBinding">

(4)     <soap:address location="http://example.com/stockquote"/>
      </port>
    </service>
```

Appendix B Microformatted WSIL

B.1 mWSIL

```
        <?xml version="1.0" encoding="UTF-8"?>
        <html xmlns="http://www.w3.org/1999/xhtml"
          xmlns:wsil="http://schemas.xmlsoap.org/ws/2001/10/inspection/"
          xmlns:wsiluddi="http://schemas.xmlsoap.org/ws/2001/10/inspection/uddi/">
        <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
        <title>AMRA</title>
        </head>
        <body>
        <ul class="xoxo">
          <xhtml:li xmlns:xhtml="http://www.w3.org/1999/xhtml"
            class="amra_wsil_inspection">
            <xhtml:span xmlns:xs="http://www.w3.org/2001/XMLSchema"
              xmlns="http://www.w3.org/2001/XMLSchema"
class="element_name">Inspection</xhtml:span>
            <xhtml:dl xmlns:xs="http://www.w3.org/2001/XMLSchema"
              xmlns="http://www.w3.org/2001/XMLSchema">
              <dt xmlns="http://www.w3.org/1999/xhtml">Namespaces</dt>
              <dd xmlns="http://www.w3.org/1999/xhtml" class="amrans">
              <dl>
                <dt>xml</dt>
                <dd class="amrans_xml">http://www.w3.org/XML/1998/namespace</dd>
                <dt>wsiluddi</dt>
                <dd
class="amrans_wsiluddi">http://schemas.xmlsoap.org/ws/2001/10/inspection/uddi/</dd>
                <dt>Default</dt>
                <dd
class="amrans_Default">http://schemas.xmlsoap.org/ws/2001/10/inspection/</dd>
              </dl>
              </dd>
            </xhtml:dl>
            <xhtml:ul xmlns:xs="http://www.w3.org/2001/XMLSchema"
              xmlns="http://www.w3.org/2001/XMLSchema">
              <xhtml:li class="amra_wsil_service">
                <xhtml:span class="element_name">Service</xhtml:span>
                <xhtml:dl></xhtml:dl>
                <xhtml:ul>
                  <xhtml:li class="amra_wsil_abstract">
                    <xhtml:span class="element_name">Abstract</xhtml:span>
                    <xhtml:dl></xhtml:dl>
                    <xhtml:ul>
                      <li xmlns="http://www.w3.org/1999/xhtml" class="amra_text">A
                      stock quote service with two descriptions</li>
                    </xhtml:ul>
                  </xhtml:li>
                  <xhtml:li class="amra_wsil_name">
                    <xhtml:span class="element_name">Name</xhtml:span>
                    <xhtml:dl></xhtml:dl>
                    <xhtml:ul>
                      <li xmlns="http://www.w3.org/1999/xhtml"
class="amra_text">Stocks</li>
                    </xhtml:ul>
                  </xhtml:li>
                  <xhtml:li class="amra_wsil_description">
                    <xhtml:span class="element_name">Description</xhtml:span>
                    <xhtml:dl>
                      <dt xmlns="http://www.w3.org/1999/xhtml">referencedNamespace</dt>
                      <dd xmlns="http://www.w3.org/1999/xhtml"

  class="amra__referencedNamespace">http://schemas.xmlsoap.org/wsdl/</dd>
                      <dt xmlns="http://www.w3.org/1999/xhtml">location</dt>
                      <dd xmlns="http://www.w3.org/1999/xhtml"
```

```
class="amra__location">http://example.com/stockquote.wsdl</dd>
                    </xhtml:dl>
                    <xhtml:ul></xhtml:ul>
                  </xhtml:li>
                  <xhtml:li class="amra_wsil_description">
                    <xhtml:span class="element_name">Description</xhtml:span>
                    <xhtml:dl>
                      <dt xmlns="http://www.w3.org/1999/xhtml">referencedNamespace</dt>
                      <dd xmlns="http://www.w3.org/1999/xhtml"
                        class="amra__referencedNamespace">urn:uddi-org:api</dd>
                    </xhtml:dl>
                    <xhtml:ul>
                      <xhtml:li class="wsiluddi_serviceDescription">
                        <xhtml:span xmlns:uddi="urn:uddi-org:api"
class="element_name">ServiceDescription</xhtml:span>
                        <xhtml:dl xmlns:uddi="urn:uddi-org:api">
                          <dt xmlns="http://www.w3.org/1999/xhtml">location</dt>
                          <dd xmlns="http://www.w3.org/1999/xhtml"
class="amra__location">http://www.example.com/uddi/inquiryapi</dd>
                        </xhtml:dl>
                        <xhtml:ul xmlns:uddi="urn:uddi-org:api">
                          <xhtml:li class="amra_wsiluddi_serviceKey">
                            <xhtml:span class="element_name">ServiceKey</xhtml:span>
                            <xhtml:dl></xhtml:dl>
                            <xhtml:ul>
                              <li xmlns="http://www.w3.org/1999/xhtml" class="amra_text">
                              4FA28580-5C39-11D5-9FCF-BB3200333F79</li>
                            </xhtml:ul>
                          </xhtml:li>
                        </xhtml:ul>
                      </xhtml:li>
                    </xhtml:ul>
                  </xhtml:li>
                </xhtml:ul>
              </xhtml:li>
              <xhtml:li class="amra_wsil_service">
                <xhtml:span class="element_name">Service</xhtml:span>
                <xhtml:dl></xhtml:dl>
                <xhtml:ul>
                  <xhtml:li class="amra_wsil_description">
                    <xhtml:span class="element_name">Description</xhtml:span>
                    <xhtml:dl>
                      <dt xmlns="http://www.w3.org/1999/xhtml">referencedNamespace</dt>
                      <dd xmlns="http://www.w3.org/1999/xhtml"

  class="amra__referencedNamespace">http://schemas.xmlsoap.org/wsdl/</dd>
                      <dt xmlns="http://www.w3.org/1999/xhtml">location</dt>
                      <dd xmlns="http://www.w3.org/1999/xhtml"
class="amra__location">ftp://anotherexample.com/tools/calculator.wsdl</dd>
                    </xhtml:dl>
                    <xhtml:ul></xhtml:ul>
                  </xhtml:li>
                </xhtml:ul>
              </xhtml:li>
              <xhtml:li class="amra_wsil_link">
                <xhtml:span class="element_name">Link</xhtml:span>
                <xhtml:dl>
                  <dt xmlns="http://www.w3.org/1999/xhtml">referencedNamespace</dt>
                  <dd xmlns="http://www.w3.org/1999/xhtml"

  class="amra__referencedNamespace">http://schemas.xmlsoap.org/ws/2001/10/inspection/</dd>
                  <dt xmlns="http://www.w3.org/1999/xhtml">location</dt>
                  <dd xmlns="http://www.w3.org/1999/xhtml"
class="amra__location">http://localhost:8080/AMRAWeb/sample/combined.xhtml</dd>
                </xhtml:dl>
                <xhtml:ul></xhtml:ul>
              </xhtml:li>
            </xhtml:ul>
          </xhtml:li>
```

```
</ul>
</body>
</html>
```

B.2 WSIL

```
<?xml version="1.0"?>
<inspection xmlns="http://schemas.xmlsoap.org/ws/2001/10/inspection/"
  xmlns:wsiluddi="http://schemas.xmlsoap.org/ws/2001/10/inspection/uddi/">
  <service>
    <abstract>A stock quote service with two descriptions</abstract>
    <name>Stocks</name>
    <description
      referencedNamespace="http://schemas.xmlsoap.org/wsdl/"
      location="http://example.com/stockquote.wsdl" />
    <description referencedNamespace="urn:uddi-org:api">
      <wsiluddi:serviceDescription
        location="http://www.example.com/uddi/inquiryapi">
        <wsiluddi:serviceKey>
          4FA28580-5C39-11D5-9FCF-BB3200333F79
        </wsiluddi:serviceKey>
      </wsiluddi:serviceDescription>
    </description>
  </service>
  <service>
    <description
      referencedNamespace="http://schemas.xmlsoap.org/wsdl/"
      location="ftp://anotherexample.com/tools/calculator.wsdl" />
  </service>
  <link
    referencedNamespace="http://schemas.xmlsoap.org/ws/2001/10/inspection/"
    location="http://example.com/moreservices.wsil" />
</inspection>
```

B.3 Correspondence

This diagram shows the correspondence between the WSIL and the mWSIL.

mWSIL

```
(1) <xhtml:li class="amra_wsil_description">
      <xhtml:span class="element_name">Description</xhtml:span>
      <xhtml:dl>
        <dt xmlns="http://www.w3.org/1999/xhtml">referencedNamespace</dt>
        <dd xmlns="http://www.w3.org/1999/xhtml"
          class="amra__referencedNamespace">urn:uddi-org:api</dd>
      </xhtml:dl>
      <xhtml:ul>
(2)     <xhtml:li class="wsiluddi_serviceDescription">
          <xhtml:span xmlns:uddi="urn:uddi-org:api"
            class="element_name">ServiceDescription</xhtml:span>
          <xhtml:dl xmlns:uddi="urn:uddi-org:api">
(3)         <dt xmlns="http://www.w3.org/1999/xhtml">location</dt>
            <dd xmlns="http://www.w3.org/1999/xhtml"
              class="amra__location">http://www.example.com/uddi/inquiryapi</dd>
          </xhtml:dl>
          <xhtml:ul xmlns:uddi="urn:uddi-org:api">
(4)         <xhtml:li class="amra_wsiluddi_serviceKey">
              <xhtml:span class="element_name">ServiceKey</xhtml:span>
              <xhtml:dl></xhtml:dl>
              <xhtml:ul>
(5)             <li xmlns="http://www.w3.org/1999/xhtml" class="amra_text">
                  4FA28580-5C39-11D5-9FCF-BB3200333F79</li>
              </xhtml:ul>
            </xhtml:li>
          </xhtml:ul>
        </xhtml:li>
      </xhtml:ul>
  </xhtml:li>
```

WSIL

```
(1) <description referencedNamespace="urn:uddi-org:api">




(2)   <wsiluddi:serviceDescription


(3)     location="http://www.example.com/uddi/inquiryapi">




(4)     <wsiluddi:serviceKey>


(5)       4FA28580-5C39-11D5-9FCF-BB3200333F79

        </wsiluddi:serviceKey>

      </wsiluddi:serviceDescription>

    </description>
```

Appendix C Generator

This appendix presents the XSLT code for the generator for WSIL. For brevity, the XSLT for the UDDI and XML Schema schemas have been omitted.

C.1 WSILXHTML.xsl

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:xs="http://www.w3.org/2001/XMLSchema"
                xmlns:xhtml="http://www.w3.org/1999/xhtml"
                version="2.0">
   <xsl:include href="amra_xhtml.xsl"/>
   <xsl:include href="WSIL/inspection.xsl"/>
   <xsl:include href="WSIL/uddi.xsl"/>
   <xsl:include href="WSIL/wsil-uddi.xsl"/>
   <xsl:include href="WSIL/wsil-wsdl.xsl"/>
   <xsl:include href="WSIL/XMLSchema.xsl"/>
</xsl:stylesheet>
```

C.2 amra_xhtml.xsl

```
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:wsil="http://schemas.xmlsoap.org/ws/2001/10/inspection/"
  xmlns:wsiluddi="http://schemas.xmlsoap.org/ws/2001/10/inspection/uddi/"
  xmlns="http://www.w3.org/1999/xhtml">

  <xsl:template match="/">
    <html>
      <head>
        <meta http-equiv="Content-Type"
          content="application/xhtml+xml; charset=ISO-8859-1" />
        <title>AMRA</title>
      </head>
      <body>
        <ul class="xoxo">
          <xsl:apply-templates />
        </ul>
      </body>
    </html>
  </xsl:template>

  <xsl:template match="text()">
    <xsl:call-template name="AMRA_text_node" />
  </xsl:template>

  <xsl:template name="AMRA_text_node">
    <li class="text">
      <xsl:value-of select="." />
    </li>
  </xsl:template>

  <xsl:template name="AMRA_add_def">
    <xsl:param name="attribute" />
    <xsl:param name="label" />
    <xsl:param name="class" />
    <xsl:if test="$attribute">
      <dt>
        <xsl:value-of select="$label" />
      </dt>
    <dd class="{$class}">
      <xsl:value-of select="$attribute" />
    </dd>
  </xsl:if>
</xsl:template>

<xsl:template name="AMRA_generate_namespace_defs">
  <xsl:param name="context" />
  <xsl:variable name="distinct_prefixes" as="item()*">
    <xsl:variable name="parent_prefixes" as="item()*">
      <xsl:if test="$context/.. instance of element()">
        <xsl:sequence
          select="in-scope-prefixes($context/..)" />
      </xsl:if>
    </xsl:variable>
    <xsl:for-each select="in-scope-prefixes($context)">
      <xsl:if
        test="not(index-of($parent_prefixes,current()))">
        <xsl:sequence select="current()" />
      </xsl:if>
    </xsl:for-each>
  </xsl:variable>
  <xsl:message>
    <xsl:text>Distinct prefixes:</xsl:text>
    <xsl:value-of select="$distinct_prefixes" />
  </xsl:message>
  <xsl:if test="not(empty($distinct_prefixes))">
    <dt>Namespaces</dt>
    <dd class="amrans">
      <dl>
        <xsl:for-each select="$distinct_prefixes">
          <xsl:variable name="prefix">
            <xsl:choose>
              <xsl:when test="current() eq ''">
                <xsl:value-of select="'Default'" />
              </xsl:when>
              <xsl:otherwise>
                <xsl:value-of select="current()" />
              </xsl:otherwise>
            </xsl:choose>
          </xsl:variable>
          <dt>
            <xsl:value-of select="$prefix" />
          </dt>
          <dd>
            <xsl:attribute name="class">
              <xsl:value-of
                select="concat('amrans_', $prefix)" />
            </xsl:attribute>
            <xsl:value-of
              select="namespace-uri-for-prefix(current(), $context)" />
          </dd>
        </xsl:for-each>
      </dl>
    </dd>
  </xsl:if>
</xsl:template>
<xsl:template name="AMRA_generate_attribute_defs">
  <xsl:for-each select="./@*">
    <xsl:variable name="attribute_name"
      select="name(current())" />
    <xsl:variable name="label">
      <xsl:call-template name="AMRA_create_label">
        <xsl:with-param name="node" select="current()" />
      </xsl:call-template>
    </xsl:variable>
    <xsl:variable name="class">
      <xsl:call-template name="AMRA_create_class">
        <xsl:with-param name="node" select="current()" />
      </xsl:call-template>
    </xsl:variable>
    <dt>
      <xsl:value-of select="$label" />
    </dt>
    <dd class="{$class}">
      <xsl:value-of select="current()" />
    </dd>
  </xsl:for-each>
</xsl:template>

<xsl:template name="AMRA_create_label">
  <xsl:param name="node" />
  <!-- With a 2.0 processor, we could at least capitalize the first letter -->
  <xsl:variable name="name">
    <xsl:value-of select="local-name($node)" />
  </xsl:variable>
  <xsl:value-of select="$name" />
</xsl:template>

<xsl:template name="AMRA_create_class">
  <xsl:param name="node" />
  <xsl:variable name="prefix"
    select="substring-before(name($node),':')" />
  <xsl:variable name="post_prefix">
    <xsl:choose>
      <xsl:when test="contains(name($node),':')">
        <xsl:value-of
          select="substring-after(name($node),':')" />
      </xsl:when>
      <xsl:otherwise>
        <xsl:value-of select="name($node)" />
      </xsl:otherwise>
    </xsl:choose>
  </xsl:variable>
  <xsl:value-of select="concat('amra_',$prefix,'_',$post_prefix)" />
</xsl:template>
</xsl:stylesheet>
```

C.3 inspection.xsl

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:xs="http://www.w3.org/2001/XMLSchema"
                xmlns:xhtml="http://www.w3.org/1999/xhtml"
                xmlns:wsil="http://schemas.xmlsoap.org/ws/2001/10/inspection/"
                xmlns="http://www.w3.org/2001/XMLSchema"
                version="2.0">
   <xsl:strip-space elements="*"/>
   <xsl:template match="wsil:abstract">
      <xsl:element name="xhtml:li">
         <xsl:attribute name="class" select="'wsil_abstract'"/>
         <xhtml:span class="element_name">
            <xsl:choose>
               <xsl:when test="./@name">
                  <xsl:value-of select="./@name"/>
                  <xsl:text>(</xsl:text>Abstract<xsl:text>)</xsl:text>
               </xsl:when>
               <xsl:otherwise>Abstract</xsl:otherwise>
            </xsl:choose>
         </xhtml:span>
         <xhtml:dl>
            <xsl:call-template name="AMRA_generate_attribute_defs"/>
            <xsl:call-template name="AMRA_generate_namespace_defs">
               <xsl:with-param name="context" select="."/>
            </xsl:call-template>
         </xhtml:dl>
         <xhtml:ul>
         <xsl:message>
            <xsl:for-each select="child::*">
               <xsl:variable name="cur_name" select="name()"/>
               <xsl:text/>
               <xsl:value-of select="name()"/>
               <xsl:text>:</xsl:text>
               <xsl:value-of select="count(../$cur_name)"/>
               <xsl:text/>
            </xsl:for-each>
         </xsl:message>
         <xsl:apply-templates/>
      </xhtml:ul>
   </xsl:element>
</xsl:template>
<xsl:template match="wsil:name">
   <xsl:element name="xhtml:li">
      <xsl:attribute name="class" select="'wsil_name'"/>
      <xhtml:span class="element_name">
         <xsl:choose>
            <xsl:when test="./@name">
               <xsl:value-of select="./@name"/>
               <xsl:text>(</xsl:text>Name<xsl:text>)</xsl:text>
            </xsl:when>
            <xsl:otherwise>Name</xsl:otherwise>
         </xsl:choose>
      </xhtml:span>
      <xhtml:dl>
         <xsl:call-template name="AMRA_generate_attribute_defs"/>
         <xsl:call-template name="AMRA_generate_namespace_defs">
            <xsl:with-param name="context" select="."/>
         </xsl:call-template>
      </xhtml:dl>
      <xhtml:ul>
         <xsl:message>
            <xsl:for-each select="child::*">
               <xsl:variable name="cur_name" select="name()"/>
               <xsl:text/>
               <xsl:value-of select="name()"/>
               <xsl:text>:</xsl:text>
               <xsl:value-of select="count(../$cur_name)"/>
               <xsl:text/>
            </xsl:for-each>
         </xsl:message>
         <xsl:apply-templates/>
      </xhtml:ul>
   </xsl:element>
</xsl:template>
<xsl:template match="wsil:description">
   <xsl:element name="xhtml:li">
      <xsl:attribute name="class" select="'wsil_description'"/>
      <xhtml:span class="element_name">
         <xsl:choose>
            <xsl:when test="./@name">
               <xsl:value-of select="./@name"/>
               <xsl:text>(</xsl:text>Description<xsl:text>)</xsl:text>
            </xsl:when>
            <xsl:otherwise>Description</xsl:otherwise>
         </xsl:choose>
      </xhtml:span>
      <xhtml:dl>
         <xsl:call-template name="AMRA_generate_attribute_defs"/>
         <xsl:call-template name="AMRA_generate_namespace_defs">
            <xsl:with-param name="context" select="."/>
         </xsl:call-template>
      </xhtml:dl>
      <xhtml:ul>
         <xsl:message>
            <xsl:for-each select="child::*">
               <xsl:variable name="cur_name" select="name()"/>
               <xsl:text/>
               <xsl:value-of select="name()"/>
               <xsl:text>:</xsl:text>
               <xsl:value-of select="count(../$cur_name)"/>
               <xsl:text/>
            </xsl:for-each>
         </xsl:message>
         <xsl:apply-templates/>
      </xhtml:ul>
   </xsl:element>
</xsl:template>
<xsl:template match="wsil:inspection">
   <xsl:element name="xhtml:li">
      <xsl:attribute name="class" select="'wsil_inspection'"/>
      <xhtml:span class="element_name">
         <xsl:choose>
            <xsl:when test="./@name">
               <xsl:value-of select="./@name"/>
               <xsl:text>(</xsl:text>Inspection<xsl:text>)</xsl:text>
            </xsl:when>
            <xsl:otherwise>Inspection</xsl:otherwise>
         </xsl:choose>
      </xhtml:span>
      <xhtml:dl>
         <xsl:call-template name="AMRA_generate_attribute_defs"/>
         <xsl:call-template name="AMRA_generate_namespace_defs">
            <xsl:with-param name="context" select="."/>
         </xsl:call-template>
      </xhtml:dl>
      <xhtml:ul>
         <xsl:message>
            <xsl:for-each select="child::*">
               <xsl:variable name="cur_name" select="name()"/>
               <xsl:text/>
               <xsl:value-of select="name()"/>
               <xsl:text>:</xsl:text>
               <xsl:value-of select="count(../$cur_name)"/>
               <xsl:text/>
            </xsl:for-each>
         </xsl:message>
         <xsl:apply-templates/>
      </xhtml:ul>
   </xsl:element>
</xsl:template>
<xsl:template match="wsil:service">
   <xsl:element name="xhtml:li">
      <xsl:attribute name="class" select="'wsil_service'"/>
      <xhtml:span class="element_name">
         <xsl:choose>
            <xsl:when test="./@name">
               <xsl:value-of select="./@name"/>
               <xsl:text>(</xsl:text>Service<xsl:text>)</xsl:text>
            </xsl:when>
            <xsl:otherwise>Service</xsl:otherwise>
         </xsl:choose>
      </xhtml:span>
      <xhtml:dl>
         <xsl:call-template name="AMRA_generate_attribute_defs"/>
         <xsl:call-template name="AMRA_generate_namespace_defs">
            <xsl:with-param name="context" select="."/>
         </xsl:call-template>
      </xhtml:dl>
      <xhtml:ul>
         <xsl:message>
            <xsl:for-each select="child::*">
               <xsl:variable name="cur_name" select="name()"/>
               <xsl:text/>
               <xsl:value-of select="name()"/>
               <xsl:text>:</xsl:text>
```

```
                  <xsl:value-of select="count(../$cur_name)"/>
                  <xsl:text/>
               </xsl:for-each>
            </xsl:message>
            <xsl:apply-templates/>
         </xhtml:ul>
      </xsl:element>
   </xsl:template>
   <xsl:template match="wsil:link">
      <xsl:element name="xhtml:li">
         <xsl:attribute name="class" select="'wsil_link'"/>
         <xhtml:span class="element_name">
            <xsl:choose>
               <xsl:when test="./@name">
                  <xsl:value-of select="./@name"/>
                  <xsl:text>(</xsl:text>Link<xsl:text>)</xsl:text>
               </xsl:when>
               <xsl:otherwise>Link</xsl:otherwise>
            </xsl:choose>
         </xhtml:span>
         <xhtml:dl>
            <xsl:call-template name="AMRA_generate_attribute_defs"/>
            <xsl:call-template name="AMRA_generate_namespace_defs">
               <xsl:with-param name="context" select="."/>
            </xsl:call-template>
         </xhtml:dl>
         <xhtml:ul>
            <xsl:message>
               <xsl:for-each select="child::*">
                  <xsl:variable name="cur_name" select="name()"/>
                  <xsl:text/>
                  <xsl:value-of select="name()"/>
                  <xsl:text>:</xsl:text>
                  <xsl:value-of select="count(../$cur_name)"/>
                  <xsl:text/>
               </xsl:for-each>
            </xsl:message>
            <xsl:apply-templates/>
         </xhtml:ul>
      </xsl:element>
   </xsl:template>
</xsl:stylesheet>
```

C.4 wsil-uddi.xsl

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:xs="http://www.w3.org/2001/XMLSchema"
                xmlns:xhtml="http://www.w3.org/1999/xhtml"
                xmlns:wsiluddi="http://schemas.xmlsoap.org/ws/2001/10/inspection/u
ddi/"
                xmlns="http://www.w3.org/2001/XMLSchema"
                xmlns:uddi="urn:uddi-org:api"
                version="2.0">
   <xsl:strip-space elements="*"/>
   <xsl:template match="wsiluddi:discoveryURL">
      <xsl:element name="xhtml:li">
         <xsl:attribute name="class" select="'wsiluddi_discoveryURL'"/>
         <xhtml:span class="element_name">
            <xsl:choose>
               <xsl:when test="./@name">
                  <xsl:value-of select="./@name"/>
                  <xsl:text>(</xsl:text>DiscoveryURL<xsl:text>)</xsl:text>
               </xsl:when>
               <xsl:otherwise>DiscoveryURL</xsl:otherwise>
            </xsl:choose>
         </xhtml:span>
         <xhtml:dl>
```

```
            <xsl:call-template name="AMRA_generate_attribute_defs"/>
            <xsl:call-template name="AMRA_generate_namespace_defs">
               <xsl:with-param name="context" select="."/>
            </xsl:call-template>
         </xhtml:dl>
         <xhtml:ul>
            <xsl:message>
               <xsl:for-each select="child::*">
                  <xsl:variable name="cur_name" select="name()"/>
                  <xsl:text/>
                  <xsl:value-of select="name()"/>
                  <xsl:text>:</xsl:text>
                  <xsl:value-of select="count(../$cur_name)"/>
                  <xsl:text/>
               </xsl:for-each>
            </xsl:message>
            <xsl:apply-templates/>
         </xhtml:ul>
      </xsl:element>
   </xsl:template>
   <xsl:template match="wsiluddi:businessDescription">
      <xsl:element name="xhtml:li">
         <xsl:attribute name="class" select="'wsiluddi_businessDescription'"/>
         <xhtml:span class="element_name">
            <xsl:choose>
               <xsl:when test="./@name">
                  <xsl:value-of select="./@name"/>
                  <xsl:text>(</xsl:text>BusinessDescription<xsl:text>)</xsl:text>
               </xsl:when>
               <xsl:otherwise>BusinessDescription</xsl:otherwise>
            </xsl:choose>
         </xhtml:span>
         <xhtml:dl>
            <xsl:call-template name="AMRA_generate_attribute_defs"/>
            <xsl:call-template name="AMRA_generate_namespace_defs">
               <xsl:with-param name="context" select="."/>
            </xsl:call-template>
         </xhtml:dl>
         <xhtml:ul>
            <xsl:message>
               <xsl:for-each select="child::*">
                  <xsl:variable name="cur_name" select="name()"/>
                  <xsl:text/>
                  <xsl:value-of select="name()"/>
                  <xsl:text>:</xsl:text>
                  <xsl:value-of select="count(../$cur_name)"/>
                  <xsl:text/>
               </xsl:for-each>
            </xsl:message>
            <xsl:apply-templates/>
         </xhtml:ul>
      </xsl:element>
   </xsl:template>
   <xsl:template match="wsiluddi:businessKey">
      <xsl:element name="xhtml:li">
         <xsl:attribute name="class" select="'wsiluddi_businessKey'"/>
         <xhtml:span class="element_name">
            <xsl:choose>
               <xsl:when test="./@name">
                  <xsl:value-of select="./@name"/>
                  <xsl:text>(</xsl:text>BusinessKey<xsl:text>)</xsl:text>
               </xsl:when>
               <xsl:otherwise>BusinessKey</xsl:otherwise>
            </xsl:choose>
         </xhtml:span>
         <xhtml:dl>
            <xsl:call-template name="AMRA_generate_attribute_defs"/>
            <xsl:call-template name="AMRA_generate_namespace_defs">
               <xsl:with-param name="context" select="."/>
            </xsl:call-template>
         </xhtml:dl>
         <xhtml:ul>
            <xsl:message>
               <xsl:for-each select="child::*">
                  <xsl:variable name="cur_name" select="name()"/>
                  <xsl:text/>
                  <xsl:value-of select="name()"/>
                  <xsl:text>:</xsl:text>
                  <xsl:value-of select="count(../$cur_name)"/>
                  <xsl:text/>
               </xsl:for-each>
            </xsl:message>
            <xsl:apply-templates/>
         </xhtml:ul>
      </xsl:element>
   </xsl:template>
   <xsl:template match="wsiluddi:serviceDescription">
      <xsl:element name="xhtml:li">
         <xsl:attribute name="class" select="'wsiluddi_serviceDescription'"/>
         <xhtml:span class="element_name">
            <xsl:choose>
               <xsl:when test="./@name">
                  <xsl:value-of select="./@name"/>
                  <xsl:text>(</xsl:text>ServiceDescription<xsl:text>)</xsl:text>
               </xsl:when>
               <xsl:otherwise>ServiceDescription</xsl:otherwise>
            </xsl:choose>
         </xhtml:span>
         <xhtml:dl>
            <xsl:call-template name="AMRA_generate_attribute_defs"/>
            <xsl:call-template name="AMRA_generate_namespace_defs">
               <xsl:with-param name="context" select="."/>
            </xsl:call-template>
         </xhtml:dl>
         <xhtml:ul>
            <xsl:message>
               <xsl:for-each select="child::*">
                  <xsl:variable name="cur_name" select="name()"/>
                  <xsl:text/>
                  <xsl:value-of select="name()"/>
                  <xsl:text>:</xsl:text>
                  <xsl:value-of select="count(../$cur_name)"/>
                  <xsl:text/>
               </xsl:for-each>
            </xsl:message>
            <xsl:apply-templates/>
         </xhtml:ul>
      </xsl:element>
   </xsl:template>
   <xsl:template match="wsiluddi:serviceKey">
      <xsl:element name="xhtml:li">
         <xsl:attribute name="class" select="'wsiluddi_serviceKey'"/>
         <xhtml:span class="element_name">
            <xsl:choose>
               <xsl:when test="./@name">
                  <xsl:value-of select="./@name"/>
                  <xsl:text>(</xsl:text>ServiceKey<xsl:text>)</xsl:text>
               </xsl:when>
               <xsl:otherwise>ServiceKey</xsl:otherwise>
            </xsl:choose>
         </xhtml:span>
         <xhtml:dl>
            <xsl:call-template name="AMRA_generate_attribute_defs"/>
            <xsl:call-template name="AMRA_generate_namespace_defs">
               <xsl:with-param name="context" select="."/>
            </xsl:call-template>
         </xhtml:dl>
         <xhtml:ul>
```

```
            <xsl:message>
               <xsl:for-each select="child::*">
                  <xsl:variable name="cur_name" select="name()"/>
                  <xsl:text/>
                  <xsl:value-of select="name()"/>
                  <xsl:text>:</xsl:text>
                  <xsl:value-of select="count(../$cur_name)"/>
                  <xsl:text/>
               </xsl:for-each>
            </xsl:message>
            <xsl:apply-templates/>
         </xhtml:ul>
      </xsl:element>
   </xsl:template>
</xsl:stylesheet>
```

C.5 wsil-wsdl.xsl

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:xs="http://www.w3.org/2001/XMLSchema"
                xmlns:xhtml="http://www.w3.org/1999/xhtml"
                xmlns:wsilwsdl="http://schemas.xmlsoap.org/wsil/wsdl/"
                xmlns="http://www.w3.org/2001/XMLSchema"
                version="2.0">
   <xsl:strip-space elements="*"/>
   <xsl:template match="wsilwsdl:referencedService">
      <xsl:element name="xhtml:li">
         <xsl:attribute name="class" select="'wsilwsdl_referencedService'"/>
         <xhtml:span class="element_name">
            <xsl:choose>
               <xsl:when test="./@name">
                  <xsl:value-of select="./@name"/>
                  <xsl:text>(</xsl:text>ReferencedService<xsl:text>)</xsl:text>
               </xsl:when>
               <xsl:otherwise>ReferencedService</xsl:otherwise>
            </xsl:choose>
         </xhtml:span>
         <xhtml:dl>
            <xsl:call-template name="AMRA_generate_attribute_defs"/>
            <xsl:call-template name="AMRA_generate_namespace_defs">
               <xsl:with-param name="context" select="."/>
            </xsl:call-template>
         </xhtml:dl>
         <xhtml:ul>
            <xsl:message>
               <xsl:for-each select="child::*">
                  <xsl:variable name="cur_name" select="name()"/>
                  <xsl:text/>
                  <xsl:value-of select="name()"/>
                  <xsl:text>:</xsl:text>
                  <xsl:value-of select="count(../$cur_name)"/>
                  <xsl:text/>
               </xsl:for-each>
            </xsl:message>
            <xsl:apply-templates/>
         </xhtml:ul>
      </xsl:element>
   </xsl:template>
   <xsl:template match="wsilwsdl:implementedBinding">
      <xsl:element name="xhtml:li">
         <xsl:attribute name="class" select="'wsilwsdl_implementedBinding'"/>
         <xhtml:span class="element_name">
            <xsl:choose>
               <xsl:when test="./@name">
                  <xsl:value-of select="./@name"/>
                  <xsl:text>(</xsl:text>ImplementedBinding<xsl:text>)</xsl:text>
               </xsl:when>
```

```
                <xsl:otherwise>ImplementedBinding</xsl:otherwise>
             </xsl:choose>
          </xhtml:span>
          <xhtml:dl>
             <xsl:call-template name="AMRA_generate_attribute_defs"/>
             <xsl:call-template name="AMRA_generate_namespace_defs">
                <xsl:with-param name="context" select="."/>
             </xsl:call-template>
          </xhtml:dl>
          <xhtml:ul>
             <xsl:message>
                <xsl:for-each select="child::*">
                   <xsl:variable name="cur_name" select="name()"/>
                   <xsl:text/>
                   <xsl:value-of select="name()"/>
                   <xsl:text>:</xsl:text>
                   <xsl:value-of select="count(../$cur_name)"/>
                   <xsl:text/>
                </xsl:for-each>
             </xsl:message>
             <xsl:apply-templates/>
          </xhtml:ul>
       </xsl:element>
    </xsl:template>
    <xsl:template match="wsilwsdl:reference">
       <xsl:element name="xhtml:li">
          <xsl:attribute name="class" select="'wsilwsdl_reference'"/>
          <xhtml:span class="element_name">
             <xsl:choose>
                <xsl:when test="./@name">
                   <xsl:value-of select="./@name"/>
                   <xsl:text>(</xsl:text>Reference<xsl:text>)</xsl:text>
                </xsl:when>
                <xsl:otherwise>Reference</xsl:otherwise>
             </xsl:choose>
          </xhtml:span>
          <xhtml:dl>
             <xsl:call-template name="AMRA_generate_attribute_defs"/>
             <xsl:call-template name="AMRA_generate_namespace_defs">
                <xsl:with-param name="context" select="."/>
             </xsl:call-template>
          </xhtml:dl>
          <xhtml:ul>
             <xsl:message>
                <xsl:for-each select="child::*">
                   <xsl:variable name="cur_name" select="name()"/>
                   <xsl:text/>
                   <xsl:value-of select="name()"/>
                   <xsl:text>:</xsl:text>
                   <xsl:value-of select="count(../$cur_name)"/>
                   <xsl:text/>
                </xsl:for-each>
             </xsl:message>
             <xsl:apply-templates/>
          </xhtml:ul>
       </xsl:element>
    </xsl:template>
 </xsl:stylesheet>
```

Appendix D Converter

This appendix presents the XSLT code for the converter for mWSIL. For brevity, the XSLT for the UDDI and XML Schema schemas have been omitted.

D.1 mwsil.xsl

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:xs="http://www.w3.org/2001/XMLSchema"
                xmlns:xhtml="http://www.w3.org/1999/xhtml"
                version="2.0">
   <xsl:param name="amra_index" select="1"/>
   <xsl:param name="top_element" select="'amra_wsil_inspection'"/>
   <xsl:include href="amra.xsl"/>
   <xsl:include href="mWSIL/inspection.xsl"/>
   <xsl:include href="mWSIL/uddi.xsl"/>
   <xsl:include href="mWSIL/wsil-uddi.xsl"/>
   <xsl:include href="mWSIL/wsil-wsdl.xsl"/>
   <xsl:include href="mWSIL/XMLSchema.xsl"/>
   <xsl:strip-space elements="*"/>
   <xsl:output method="xml" indent="yes"/>
</xsl:stylesheet>
```

D.2 amra.xsl

```
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xhtml="http://www.w3.org/1999/xhtml">

  <!-- AMRA Specific Templates -->
  <xsl:template match="/">
    <xsl:message>Applying at root</xsl:message>
    <xsl:apply-templates select="/xhtml:html/xhtml:body" />
  </xsl:template>
  <xsl:template match="/xhtml:html/xhtml:body">
    <xsl:message>Applying in body</xsl:message>
    <xsl:apply-templates
      select=".//xhtml:ul[contains(concat(' ',@class,' '),' xoxo ')]" />
    <xsl:apply-templates
      select=".//xhtml:ol[contains(concat(' ',@class,' '),' xoxo ')]" />
  </xsl:template>
  <xsl:template match="xhtml:ul[contains(concat(' ',@class,' '),' xoxo ')]">
    <xsl:apply-templates select=".//xhtml:li[contains(concat(' ',@class,'
'),concat(' ',$top_element,' '))][$amra_index]"/>
  </xsl:template>
  <xsl:template match="xhtml:ol[contains(concat(' ',@class,' '),' xoxo ')]">
    <xsl:apply-templates select=".//xhtml:li[contains(concat(' ',@class,'
'),concat(' ',$top_element,' '))][$amra_index]"/>
  </xsl:template>
  <xsl:template match="xhtml:ul">
    <xsl:message>Applying in list</xsl:message>
    <xsl:apply-templates select="./xhtml:li" />
  </xsl:template>
  <xsl:template match="xhtml:ol">
    <xsl:message>Applying in list</xsl:message>
    <xsl:apply-templates select="./xhtml:li" />
  </xsl:template>

  <!-- Ignore any text that might be present (Any thing that we need will still be
used) -->
```

```
          <xsl:template match="xhtml:li/text()"></xsl:template>
          <xsl:template match="xhtml:span/text()"></xsl:template>
          <xsl:template match="xhtml:dd/text()"></xsl:template>
          <xsl:template match="xhtml:dt/text()"></xsl:template>

          <!-- Output text node -->
          <xsl:template match="xhtml:li[contains(@class,'text')]">
            <xsl:value-of select="." />
          </xsl:template>

          <!-- Process namespaces -->
          <xsl:template name="AMRA_process_namespaces">
            <xsl:param name="ns_node" />
            <!--    match="xhtml:dd[contains(concat(' ',normalize-space(@class),' '),'
amrans ')]"> -->
            <xsl:for-each select="$ns_node/*/xhtml:dd[contains(@class,'amrans')]">
              <xsl:variable name="prefix">
                <xsl:variable name="spaced_class"
                  select="concat(' ',normalize-space(@class),' ')" />
                <xsl:value-of
                  select="substring-before(substring-after($spaced_class,' amrans_'),' ')"
/>
              </xsl:variable>
              <xsl:message>
                <xsl:text>Prefix:</xsl:text>
                <xsl:value-of select="$prefix" />
                <xsl:text>:</xsl:text>
                <xsl:value-of select="." />
              </xsl:message>
              <xsl:choose>
                <xsl:when test="'Default' ne $prefix">
                  <xsl:namespace name="{$prefix}" select="." />
                </xsl:when>
                <xsl:otherwise>
                  <xsl:namespace name="" select="." />
                </xsl:otherwise>
              </xsl:choose>
            </xsl:for-each>
          </xsl:template>

          <!-- Process attributes from dds -->
          <xsl:template name="AMRA_process_dd_attr">
            <xsl:for-each select="./xhtml:dl/xhtml:dd[@class]">
              <xsl:choose>
                <xsl:when test="not(contains(@class,'amrans'))">
                  <xsl:variable name="attribute">
                    <xsl:call-template name="convert_class_name">
                      <xsl:with-param name="class_name"
                        select="./@class" />
                    </xsl:call-template>
                  </xsl:variable>
                  <xsl:variable name="namespace">
                    <xsl:variable name="prefix" select="substring-before($attribute,':')"/>
                    <xsl:choose>
                      <xsl:when test="$prefix eq ''">
                        <xsl:value-of select="''"/>
                      </xsl:when>
                      <xsl:otherwise>
                        <xsl:value-of select="namespace-uri-for-prefix($prefix,.)"/>
                      </xsl:otherwise>
                    </xsl:choose>
                  </xsl:variable>
                  <xsl:message><xsl:text>Attribute: </xsl:text><xsl:value-of
select="$attribute"/></xsl:message>
                  <xsl:message><xsl:text>Namespace: </xsl:text><xsl:value-of
select="$namespace"/></xsl:message>
                  <xsl:attribute name="{$attribute}" namespace="{$namespace}">
                    <xsl:value-of select="." />
                  </xsl:attribute>
                </xsl:when>
                <xsl:otherwise>
                  <!-- In this case, we have come to a amrans class, so process it! -->
                  <xsl:call-template name="AMRA_process_namespaces">
                    <xsl:with-param name="ns_node" select="." />
                  </xsl:call-template>
                </xsl:otherwise>
              </xsl:choose>
            </xsl:for-each>
          </xsl:template>

          <xsl:template name="AMRA_dd_attr">
            <xsl:param name="class" />
            <xsl:variable name="attribute">
              <xsl:call-template name="convert_class_name">
                <xsl:with-param name="class_name" select="$class" />
              </xsl:call-template>
            </xsl:variable>
            <xsl:if test="./xhtml:dl/xhtml:dd[contains(@class,$class)]">
              <xsl:attribute name="{$attribute}">
                <xsl:value-of
                  select="./xhtml:dl/xhtml:dd[contains(@class,$class)]" />
              </xsl:attribute>
            </xsl:if>
          </xsl:template>
          <xsl:template name="convert_class_name">
            <xsl:param name="class_name" />
            <!-- First, take off the first part of the attribute name (from amra_ and
earlier) -->
            <xsl:variable name="class_sans_amra"
              select="substring-after($class_name,'amra_')" />
            <!-- Now, get rid of anything afterwards -->
            <xsl:variable name="unconverted_attribute_name">
              <xsl:choose>
                <xsl:when test="contains($class_sans_amra,' ')">
                  <!-- There is a space, so there is another class -->
                  <xsl:value-of
                    select="substring-before($class_sans_amra,' ')" />
                </xsl:when>
                <xsl:otherwise>
                  <!-- If there is no space, then this is it -->
                  <xsl:value-of select="$class_sans_amra" />
                </xsl:otherwise>
              </xsl:choose>
            </xsl:variable>
            <xsl:variable name="prefix"
              select="substring-before($unconverted_attribute_name,'_')" />
            <xsl:variable name="post_prefix">
              <xsl:choose>
                <xsl:when
                  test="contains($unconverted_attribute_name,'_')">
                  <xsl:value-of
                    select="substring-after($unconverted_attribute_name,'_')" />
                </xsl:when>
                <xsl:otherwise>
                  <xsl:value-of select="$unconverted_attribute_name" />
                </xsl:otherwise>
              </xsl:choose>
            </xsl:variable>
            <xsl:choose>
              <xsl:when test="$prefix eq ''">
                <xsl:value-of select="$post_prefix" />
              </xsl:when>
              <xsl:otherwise>
                <xsl:value-of select="concat($prefix,':',$post_prefix)" />
              </xsl:otherwise>
            </xsl:choose>
          </xsl:template>
        </xsl:stylesheet>
```

D.3 inspection.xsl

```
       <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                       xmlns:xs="http://www.w3.org/2001/XMLSchema"
                       xmlns:xhtml="http://www.w3.org/1999/xhtml"
                       xmlns:wsil="http://schemas.xmlsoap.org/ws/2001/10/inspection/"
                       version="2.0">
          <xsl:template match="xhtml:li[contains(concat(' ',normalize-space(@class),'
'),' wsil_abstract ')][$amra_index]">
             <xsl:element name="wsil:abstract"
                          namespace="http://schemas.xmlsoap.org/ws/2001/10/inspection/">
                <xsl:call-template name="AMRA_process_dd_attr"/>
                <xsl:apply-templates/>
             </xsl:element>
          </xsl:template>
          <xsl:template match="xhtml:li[contains(concat(' ',normalize-space(@class),'
'),' wsil_name ')][$amra_index]">
             <xsl:element name="wsil:name"
namespace="http://schemas.xmlsoap.org/ws/2001/10/inspection/">
                <xsl:call-template name="AMRA_process_dd_attr"/>
                <xsl:apply-templates/>
             </xsl:element>
          </xsl:template>
          <xsl:template match="xhtml:li[contains(concat(' ',normalize-space(@class),'
'),' wsil_description ')][$amra_index]">
             <xsl:element name="wsil:description"
                          namespace="http://schemas.xmlsoap.org/ws/2001/10/inspection/">
                <xsl:call-template name="AMRA_process_dd_attr"/>
                <xsl:apply-templates/>
             </xsl:element>
          </xsl:template>
          <xsl:template match="xhtml:li[contains(concat(' ',normalize-space(@class),'
'),' wsil_inspection ')][$amra_index]">
             <xsl:element name="wsil:inspection"
                          namespace="http://schemas.xmlsoap.org/ws/2001/10/inspection/">
                <xsl:call-template name="AMRA_process_dd_attr"/>
                <xsl:apply-templates/>
             </xsl:element>
          </xsl:template>
          <xsl:template match="xhtml:li[contains(concat(' ',normalize-space(@class),'
'),' wsil_service ')][$amra_index]">
             <xsl:element name="wsil:service"
                          namespace="http://schemas.xmlsoap.org/ws/2001/10/inspection/">
                <xsl:call-template name="AMRA_process_dd_attr"/>
                <xsl:apply-templates/>
             </xsl:element>
          </xsl:template>
          <xsl:template match="xhtml:li[contains(concat(' ',normalize-space(@class),'
'),' wsil_link ')][$amra_index]">
             <xsl:element name="wsil:link"
namespace="http://schemas.xmlsoap.org/ws/2001/10/inspection/">
                <xsl:call-template name="AMRA_process_dd_attr"/>
                <xsl:apply-templates/>
             </xsl:element>
          </xsl:template>
       </xsl:stylesheet>
```

D.4 wsil-uddi.xsl

```
       <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                       xmlns:xs="http://www.w3.org/2001/XMLSchema"
                       xmlns:xhtml="http://www.w3.org/1999/xhtml"
                       xmlns:wsiluddi="http://schemas.xmlsoap.org/ws/2001/10/inspection/u
ddi/"
                       version="2.0">
```

```
          <xsl:template match="xhtml:li[contains(concat(' ',normalize-space(@class),'
'),' wsiluddi_discoveryURL ')][$amra_index]">
             <xsl:element name="wsiluddi:discoveryURL"
                          namespace="http://schemas.xmlsoap.org/ws/2001/10/inspection/udd
i/">
                <xsl:call-template name="AMRA_process_dd_attr"/>
                <xsl:apply-templates/>
             </xsl:element>
          </xsl:template>
          <xsl:template match="xhtml:li[contains(concat(' ',normalize-space(@class),'
'),' wsiluddi_businessDescription ')][$amra_index]">
             <xsl:element name="wsiluddi:businessDescription"
                          namespace="http://schemas.xmlsoap.org/ws/2001/10/inspection/udd
i/">
                <xsl:call-template name="AMRA_process_dd_attr"/>
                <xsl:apply-templates/>
             </xsl:element>
          </xsl:template>
          <xsl:template match="xhtml:li[contains(concat(' ',normalize-space(@class),'
'),' wsiluddi_businessKey ')][$amra_index]">
             <xsl:element name="wsiluddi:businessKey"
                          namespace="http://schemas.xmlsoap.org/ws/2001/10/inspection/udd
i/">
                <xsl:call-template name="AMRA_process_dd_attr"/>
                <xsl:apply-templates/>
             </xsl:element>
          </xsl:template>
          <xsl:template match="xhtml:li[contains(concat(' ',normalize-space(@class),'
'),' wsiluddi_serviceDescription ')][$amra_index]">
             <xsl:element name="wsiluddi:serviceDescription"
                          namespace="http://schemas.xmlsoap.org/ws/2001/10/inspection/udd
i/">
                <xsl:call-template name="AMRA_process_dd_attr"/>
                <xsl:apply-templates/>
             </xsl:element>
          </xsl:template>
          <xsl:template match="xhtml:li[contains(concat(' ',normalize-space(@class),'
'),' wsiluddi_serviceKey ')][$amra_index]">
             <xsl:element name="wsiluddi:serviceKey"
                          namespace="http://schemas.xmlsoap.org/ws/2001/10/inspection/udd
i/">
                <xsl:call-template name="AMRA_process_dd_attr"/>
                <xsl:apply-templates/>
             </xsl:element>
          </xsl:template>
       </xsl:stylesheet>
```

D.5 wsil-wsdl.xsl

```
       <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                       xmlns:xs="http://www.w3.org/2001/XMLSchema"
                       xmlns:xhtml="http://www.w3.org/1999/xhtml"
                       xmlns:wsilwsdl="http://schemas.xmlsoap.org/wsil/wsdl/"
                       version="2.0">
          <xsl:template match="xhtml:li[contains(concat(' ',normalize-space(@class),'
'),' wsilwsdl_referencedService ')][$amra_index]">
             <xsl:element name="wsilwsdl:referencedService"
                          namespace="http://schemas.xmlsoap.org/wsil/wsdl/">
                <xsl:call-template name="AMRA_process_dd_attr"/>
                <xsl:apply-templates/>
             </xsl:element>
          </xsl:template>
          <xsl:template match="xhtml:li[contains(concat(' ',normalize-space(@class),'
'),' wsilwsdl_implementedBinding ')][$amra_index]">
             <xsl:element name="wsilwsdl:implementedBinding"
                          namespace="http://schemas.xmlsoap.org/wsil/wsdl/">
                <xsl:call-template name="AMRA_process_dd_attr"/>
                <xsl:apply-templates/>
             </xsl:element>
          </xsl:template>
          <xsl:template match="xhtml:li[contains(concat(' ',normalize-space(@class),'
'),' wsilwsdl_reference ')][$amra_index]">
             <xsl:element name="wsilwsdl:reference"
namespace="http://schemas.xmlsoap.org/wsil/wsdl/">
                <xsl:call-template name="AMRA_process_dd_attr"/>
                <xsl:apply-templates/>
             </xsl:element>
          </xsl:template>
       </xsl:stylesheet>
```

Appendix E Other Software

The registry Web application is implemented using Apache Tomcat 5.5 [Apache Software Foundation]. WSDL and WSIL documents are represented internally using WSDL4J and WSIL4J. The several JSPs involved in the Web application use custom tags to separate, in some degree, the presentation from the business logic. To utilize XSLT 2.0 and XPath 2.0, this application also uses the Saxon B [Kay] open source XSLT engine. In some places hCard support is needed – for this an XSLT transform [Suda], mentioned on the microformats.org hCard site, is used. To provide search capabilities, the Apache Lucene [Apache Software Foundation] Java search engine is used.

Bibliography

Adamopoulos, D., et al. 1999. Distributed Object Platforms in Telecommunications: A Comparison between DCOM and CORBA. *British Telecommunications Engineering Journal 18*, 2, 43-49.

Amazon. Alexa Top Sites. http://aws.amazon.com/alexatopsites.

Apache Software Foundation. Axis User's Guide. http://ws.apache.org/axis/java/user-guide.html.

Apache Software Foundation. Apache Lucene. http://lucene.apache.org/.

Apache Software Foundation. Apache Tomcat. http://tomcat.apache.org/.

Appnel, T. 2002. An Introduction to WSIL. *OnJava.com*. http://www.onjava.com/pub/a/onjava/2002/10/16/wsil.html. Accessed on 1 September 2005.

Ballinger, K, et al. 2001. Web Services Inspection Language (WS-Inspection) 1.0. *IBM DeveloperWorks*. http://www-128.ibm.com/developerworks/library/specification/ws-wsilspec/. Accessed on 1 September 2005.

Bloomberg, J. 2004. UDDI: Straw Man Or Ugly Duckling? *SearchWebServices.com*. http://searchwebservices.techtarget.com/originalContent/0,289142,sid26_gci990488,00.html. Accessed on 1 September 2005.

Campbell, A. T., Coulson, G., and Kounavis, M. E. 1999. Managing Complexity: Middleware Explained. *IT Professional 1*, 5, 22-28.

Çelik, T. 2004. Extensible Open XHTML Outlines. *http://microformats.org/wiki/xoxo*. Microformats.org.

Çelik, T. 2005a. So You Wanna Develop a New Microformat? *Microformats.org*. http://microformats.org/wiki/process. Accessed on 1 September 2005.

Çelik, T., Meyer, E. A., and Mullenweg, M. 2005b. XHTML Meta Data Profiles. *WWW '05: Special Interest Tracks and Posters of the 14th International Conference on World Wide Web* (Chiba, Japan). ACM Press, New York, 994-995.

Çelik, T. 2006. hCard. http://microformats.org/wiki/hcard.

Chinnici, R., Moreau, J., Ryman, A., and Weerawarana, S. 2005. Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. *W3C* . http://www.w3.org/TR/2005/WD-wsdl20-20050803/. Accessed on 13 September 2005.

Clement, L., Hately, A., von Riegen, C., and Rogers, T. 2004. UDDI Version 3.0.2. *OASIS Open*.

Dawson, F., Howes, T. 1998. vCard MIME Directory Profile. *The Internet Society.* http://www.ietf.org/rfc/rfc2426.txt.

Dawson, F., Stenerson, D. 1998. Internet Calendaring and Scheduling Core Object Specification (iCalendar). *The Internet Society.* http://www.ietf.org/rfc/rfc2445.txt.

Gisolfi, D. 2001. Web Services Architect, Part 1: An Introduction to Dynamic e-Business. *IBM DeveloperWorks*. http://www-106.ibm.com/developerworks/webservices/library/ws-arc1/. Accessed 1 September 2005.

Google A. Google Web APIs. http://www.google.com/apis/.

Google B. Google Web Search Features. http://www.google.com/help/features.html.

Hoschek, W. 2002. The Web Service Discovery Architecture. In *Proceedings of the 2002 ACM/IEEE conference on Supercomputing* (Baltimore, MD). IEEE Computer Society Press, Los Alamitos, CA, 1-15.

Invesbot. CompanySearch. http://ws.invesbot.com/companysearch.asmx.

Kay, M. Saxon: The XSLT and XQuery Processor. http://saxon.sourceforge.net/.

Kay, M. 2005. XSL Transformations (XSLT) Version 2.0. *http://www.w3.org/TR/xslt20/*. World Wide Web Consortium.

Kreger, H. 2003. Fulfilling the Web Services Promise. *Communications of the ACM 46*, 6, 29-34.

Liang, Q., Su, S.Y.W., Li, H., and Chung, J.-Y. 2003. A united approach to discover multimedia Web services. In *Proceedings. Fifth International Symposium on Multimedia Software Engineering, 2003.* IEEE, 62-69.

Lemahieu, W. 2001. Web Service Description, Advertising, and Discovery: WSDL and Beyond. *New Directions in Software Engineering (Eds. J. Vandenbulcke and M. Snoeck)*, First ed., Leuven University Press, Leuven, Belgium. 135-152.

microformats.org A. About Microformats. http://microformats.org/about/. Accessed on 1 September 2005.

microformats.org B. Search Results – Example. http://microformats.org/wiki/search-results-example.

microformats.org C. XHTML-REST Brainstorming.

http://microformats.org/wiki/rest-brainstorming.

Miles, S., Papay, J., Dialani, V., Luck, M., Decker, K., Payne, T., and Moreau, L. 2003. Personalised Grid service Discovery. *Software, IEE Proceedings 150*, 4, 252-256 .

Mimoso, M. S. 2004. Insurance that SOA Works. *SearchWebServices.com*. http://searchwebservices.techtarget.com/originalContent/0,289142,sid26_gci1030044,00.html. Accessed 1 September 2005.

Modi, T. 2002. WSIL: Do we need another Web Services Specification? *WebServicesArchitect.com*. http://www.webservicesarchitect.com/content/articles/modi01.asp. Accessed on 10 June 2005.

Nagy, W. and Ballinger, K. 2001. The WS-Inspection and UDDI Relationship. *IBM DeveloperWorks* . http://www-128.ibm.com/developerworks/webservices/library/ws-wsiluddi.html. Accessed on 13 September 2005.

Narayanan, S, and McIlraith, S. 2002. Simulation, Verification, and Automated Composition of Web Services. In *Proceedings of the 11th International conference on World Wide Web* . ACM Press, New York, 77-88.

Peeters, J. 2003. WSDL Tales from the Trenches, Part 1. *O'Reilly webservices.xml.com*. http://webservices.xml.com/pub/a/ws/2003/05/27/wsdl.html. Accessed on 1 September 2005.

Rodgers, K. 2003. UDDI finds a role after all. *LooselyCoupled.com*. http://www.looselycoupled.com/stories/2003/uddi-role-infr0220.html. Accessed on 10 September 2005.

Suda, B. X2V. http://suda.co.uk/projects/X2V/.

Yahoo. Search Services. http://tools.search.yahoo.com/about/forsearchers.html.

Yu, J, and Zhou, G. 2004. Dynamic Web service invocation based on UDDI. In *E-Commerce Technology for Dynamic E-Business, 2004. IEEE International Conference on* . 154-157.