MIT Sloan School of Management

Working Paper 4406-02
CISL 2002-14
May 2002

Data Integration using Web Services

Mark Hansen, Stuart Madnick, Michael Siegel

Data Integration using Web Services

Mark Hansen[1], Stuart Madnick[2], Michael Siegel[2]

[1] MIT Sloan School of Management, E53-321, 30 Wadsworth St, Cambridge, MA 021239
khookguy@yahoo.com
[2] MIT Sloan School of Management, E53-321, 30 Wadsworth St, Cambridge, MA 021239
{smadnick, msiegel}@mit.edu

Abstract. In this paper we examine the opportunities for data integration in the context of the emerging Web Services systems development paradigm. The paper introduces the programming standards associated with Web Services and provides an example of how Web Services can be used to unlock heterogeneous business systems to extract and integrate business data. We provide an introduction to the problems and research issues encountered when applying Web Services to data integration. We provide a formal definition of aggregation (as a type of data integration) and discuss the impact of Web Services on aggregation. We show that Web Services will make the development of systems for aggregation both faster and less expensive to develop. A system architecture for Web Services based aggregation is presented that is representative of products available from software vendors today. Finally, we highlight some of the challenges facing Web Services that are not currently being addressed by standards bodies or software vendors. These include context mediation, trusted intermediaries, quality and source selection, licensing and payment mechanisms, and systems development tools. We suggest some research directions for each of these challenges.

1 Introduction

By providing interface standards, Web Services can be viewed as programming paradigm for extracting and integrating data from heterogeneous information systems. It offers significant advantages over currently available methods and tools. These advantages have been widely discussed in the popular Information Technology press[1]. Because the Web Services paradigm is based on a new set of standards (e.g., XML, SOAP, WSDL, UDDI)[2] it promises to enable the aggregation of multiple data sources once these standards are supported by the information systems underlying each business process. These standards are being widely adopted in industry as evidenced by Microsoft's .NET initiative and Sun's Java APIs for XML (JAX) extensions to the Java 2 Platform, Enterprise Edition (J2EE). [12]

We believe that, from a research standpoint, it is useful to view Web Services as a paradigm for aggregation. Using that analogy, we investigate the challenges researchers

[1] "Vendors Rally Behind Web Services Spec", *InformationWeek*, November 27, 2000; "Web Services Move One Small Step Closer To Reality", *InformationWeek*, February 12, 2001

[2] Section 4 defines these acronyms.

have uncovered related to aggregation. [1][2][3][7][13] and apply these to Web Services. Foremost among these challenges are the issues of semantics and context mediation.

This paper begins with an example illustrating the power of Web Services as a data integration approach in a telecommunications company. It goes on to illustrate how such an application of Web Services is really a form of aggregation. We provide a working definition of aggregation and examine the application of existing aggregation research to Web Services.

We then briefly explore industry support for Web Services and the technology architecture being adopted by most software vendors for applying Web Services to data integration problems. Lastly, we identify some potential challenges facing Web Services, propose additional infrastructure that will be necessary, and point to some promising research that may be applied to create that infrastructure.

2 Example of a Data Integration Architecture based on Web Services[3]

International Communications (IC) is a worldwide provider of voice and data (Internet) communications services to global corporations. IC has grown by acquisition and has a variety of information systems in different parts of the world that need to be integrated to provide a global view of available services to their global enterprise customers

For example, consider the Global Provisioning System (GPS) required by the corporate headquarters. When a global customer, such as Worldwide Widgets (WW), asks IC to bid on a contract to provide services, IC must turn to its various global subsidiaries to provision the circuits to fulfill this order. The process starts by creating a master order in the corporate GPS. Being able to create a master order implies that the provisioning data from all subsidiaries has been aggregated together into a master data source. It also requires integration with subsidiary support systems (e.g., Trouble Tickets, Usage Statistics). It is an example of intra-organizational aggregation (i.e., aggregating data within an organization).

Once completed, the master order is communicated to each subsidiary to derive the local provisioning plan in their geography.

2.1 Potential Solutions

IC considered a spectrum of alternatives for building an Aggregator for the GPS, summarized in the table below.

[3] Although the details are fictitious, this example is based on real examples of Aggregation challenges faced in the telecommunications industry.

Integration Alternative	Description
Single System	This approach involves replacing all the divisional provisioning components with a single, integrated, system.
Component Interfaces	This approach involves modifying all the divisional components to provide a Web Services interface.
Web Process Wrappers	This approach involves wrapping the existing divisional components with a thin layer of code to provide a Web Service interface.

IC wanted to implement the Single System alternative because it would standardize meta-data throughout the organization and reduce the amount of custom code development and maintenance required to aggregate divisional data up to corporate. However, there were several problems that prevented IC from pursuing this option. First, replacing all the divisional systems would be a multi-year, hugely expensive, project that would require complete retraining the existing divisional Information Technology (IT) employees and end users. Expensive consultants would be needed to assist with installation, configuration, and extensive retraining.[4] Additionally, IC was acquiring companies and needed a quick way to integrate them with corporate systems.

Considering these challenges, IC decided to implement a five-year plan to standardize divisional systems. In the mean time, IC decided to create custom interfaces between divisional and corporate systems. By building prototype Web Services interfaces for one division, IC determined that this approach leveraged local knowledge to quickly create useful interfaces to the GPS. Some divisional systems had interfaces where the fast and simple task of building Web Process Wrappers was sufficient. In other cases, more work was required to modify a divisional system to create a Component Interface supplying Web Services to the GPS.

2.2 Implementing Web Services Interfaces

Implementing the integration architecture using the Web Services paradigm implied using the following standards for systems integration (See Section 4 for more discussion of these standards.):

- Data would be communicated between systems in a standard XML format.
- SOAP would be used to send and receive XML documents.
- Aggregation interfaces specifications would be defined with WSDL.
- A registry of all system interfaces would be published using the UDDI.

The Web Services interfaces between the Global Provisioning System and the systems in "Division A" are illustrated below (Figure 1) such as Provisioning, Trouble Tickets, and Usage Statistics. Similar interfaces would be needed for all the divisions.

---

[4] Lisa Vaas, "Keeping Air Force Flying High," *eWeek*, 22 October 2001, available at http://www.eweek.com/print_article/0,3668,a%253D16944,00.asp / Excerpt: "…The outcome wasn't good. After three painstaking years and a substantial investment — Dittmer declined to quote a cost — a mere 27 percent of the original code's functionality had been reproduced. Originally, Dittmer said, they had expected to retrieve 60 percent of functionality. Eventually, the Air Force killed the project. … Rewriting the systems from scratch would have eaten up an impermissibly large chunk of the Air Force's budget. 'We don't have the money to go out and say, 'OK, let's wholesale replace everything,' Jones said …"

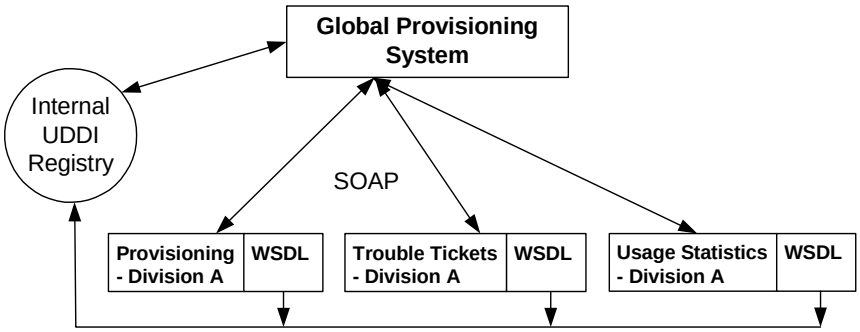

Figure 1. International Communications' Web Services Interfaces

3 Web Services and Aggregation

Research on information aggregation has been going on for a long time, but with the advent of the Internet there has been a new focus on the entities that aggregate information from heterogeneous web sites – often referred to as "Aggregators"[1]. Much of this research focuses on the semantic and contextual challenges of aggregation [5][7], and as we will see in Section 0 many of these challenges remain in the Web Services paradigm.

Web Services do, however, solve a number of the technical challenges faced by early Internet Aggregators. These Aggregators had to overcome technical challenges related to integration of data source sites that were not originally developed with the intent of supporting aggregation. Screen scraping and "web farming" [4] techniques were developed where the Aggregator accessed the source site as if it were a user and parsed the resulting Hyper Text Markup Language (HTML) to extract the information being aggregated.

The Web Services paradigm solves some of the technical integration challenges by standardizing the infrastructure for data exchange. However, the Web Services paradigm also assumes that application components are designed with the intention of being aggregated. This assumption, that disparate data sources are going to be designed and implemented with the intention of being aggregated, raises a whole new set of challenges that we discuss in Section 7.

To begin exploring the challenges posed by the Web Services paradigm for aggregation, we propose the following definition that encompasses both information and processes aggregation.

An Aggregator is an entity that:

- Transparently collects and analyzes information from different data sources;
- Resolves the semantic and contextual differences in the information;
- Addresses one or more of the following aggregation purposes / capabilities:
  - Content Aggregation
  - Comparison Aggregation
  - Relationship Aggregation
  - Process Aggregation

3.1 Aggregation Purposes / Capabilities

From this definition, we see that not every system designed to integrate data can be called an Aggregator. To be an Aggregator, a system must provide certain capabilities, as summarized here.

Aggregation Capability	Definition	Example
Content Aggregation	Pulls together information related to a specific topic (e.g., IBM Corporation) and provides value-added analytics based on relationships across multiple data sources.	Employee Benefits Portals where an employee can get access to all his benefits information (e.g., health plan, 401K, etc.)
Comparison Aggregation	Within a particular business domain identifies the optimal transaction based on criteria supplied by the user (e.g., price, time).	Shopbots that compare product prices (e.g., www.mysimon.com, www.dealtime.com).
Relationship Aggregation	Provides a single point of contact between a user and several business services / information sources with which the user has a business relationship.	Aggregation of all your frequent flyer programs (e.g., www.maxmiles.com) or financial accounts (e.g., www.yodlee.com).
Process Aggregation	Provides a single point of contact for managing a business process that requires coordination across a variety of services / information sources.	B2B and EAI tools that provide rule-based workflow and data aggregation to link multiple business processes together (e.g., WebMethods, BizTalk)

3.2 Aggregation Setting

Aggregation types get applied in different settings and have more or less relevance depending on the setting. Three common settings where aggregation is employed are:

- Intra-Organizational – to integrate systems and data within an organization. Process Aggregation is particularly important here where it is often referred to as Enterprise Application Integration (EAI).
- Inter-Organizational – to integrate systems and data across multiple organizations. All aggregation capabilities are important in this context. Process Aggregation is used in many forms of Business-to-Business (B2B) communication such as Supply Chain Management. Many of the Business to Consumer (B2C) Aggregators employ Content Management (e.g., MyYahoo[5]), Comparison (e.g., MySimon[6]), and Relationship (e.g., Yodlee[7]) capabilities.
- Market/Exchange – to create an independent organization and systems to facilitate commerce among members. Process Aggregation, Content Management, and Comparison capabilities are particularly important in this context. A good example is The World Chemical Exchange (www.chemconnect.com) where you can solicit bids from vendors (Comparison), browse and learn about trading partners (Content Management) and buy, sell, and integrate your supply chain with other vendors (Process Aggregation).

The International Communications example represents an Intra-Organizational setting.

[5] See www.my.yahoo.com
[6] See www.mysimon.com
[7] See www.yodlee.com

4 Web Services Standards – Current State

The Web Services paradigm provides a new set of standards and technologies that facilitate an organization's ability to integrate data from internal heterogeneous systems (e.g., Enterprise Application Integration (EAI)) or integrate data from business partners (e.g., Supply Chain Management and other Business-to-Business (B2B) type applications). These types of systems can be characterized as various types of Aggregators.

For our purposes, we define a Web Service as an application interface that conforms to specific standards in order to enable other applications to communicate with it through that interface regardless of programming language, hardware platform, or operating system. A Web Service interface complies with the following standards:

- XML (eXtensible Markup Language[8]) documents are used for data input and output.
- HTTP (Hypertext Transfer Protocol[9]) or a Message Oriented Middleware (MOM) product (e.g., IBM's MQ Series) is the application protocol.
- SOAP (Simple Object Access Protocol[10]) is the standard specifying how XML documents are exchanged over HTTP or MOM.
- WSDL (Web Services Description Language[11]) is used to provide a meta-data description of the input and output parameters for the interface.
- UDDI (Universal Description, Discovery and Integration[12]) is used to register the Web Service.

Although there is no single standard for XML document structure, many Web Services that are designed to work together will standardize on a particular set of tags or document structure. Various industry groups and standards bodies are publishing XML standards for use in particular contexts. One example that is building support among technology vendors is ebXML (www.ebxml.org).

4.1 How Standards are used for Aggregation

Figure 2 illustrates a generic example of how Web Services standards are employed for Aggregation. This is a generic version of Figure 1 where the box labeled "Aggregator" replaces IC's Global Provisioning System. The programmers developing this system need to integrate the provisioning data provided by systems in various divisions. They accomplish this task by defining standard XML document types as needed (e.g., Order, Provisioning). These documents make use of standard tags for data such as price and bandwidth.

[8] See www.w3.org/XML
[9] See www.w3.org/Protocols
[10] See www.w3.org/2000/xp
[11] See www.w3.org/TR/wsdl
[12] See www.uddi.org

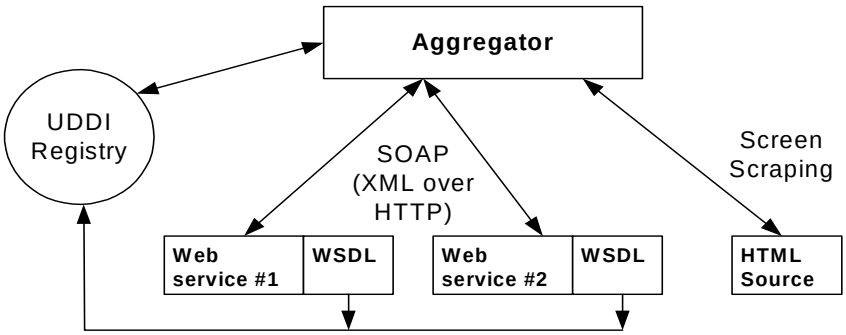

Figure 2. Aggregation with Web Services

Within each division, programmers develop a Web Service that can receive and process a query about the network provisioning available (e.g., what bandwidth frame relay connections are available between points A and B?). The interface for each division's Web Service is published using WSDL and registered in a UDDI Registry. The programmers working on the Global Provisioning System can use the UDDI Registry to look up the Web Services that the divisions have made available. From there, they can access the WSDL for each web service that specifies its inputs and outputs.

Some of the divisional Provisioning Systems may be simple enough that instead of implementing a Web Service interface, basic screen scraping off an existing HTML interface is used.

5 Aggregator Architecture

An Aggregator combines data from a variety of sources to create and maintain a new data source supporting new business processes. A standard technical architecture is emerging for creating Aggregators, and is illustrated in Figure 3. Many commercial products are based on such an architecture.

The Reporting and GUI Access components of this architecture enables the aggregated data to be treated as a single data source and provides tools for querying it as such (e.g., SQL). The Event Handling and Workflow functionality provided by such platforms provides Process Aggregation that is referred to as Enterprise Application Integration (EAI) if it involves data sources (as in our IC example) or B2B integration if it involves data from different companies (e.g., supply chain integration). All the components below this are designed to leverage Web Services standards for data aggregation.

The Global Provisioning System would use a system architecture like that illustrated in Figure 3. When IC needs to provision a global order, the order is translated into an XML document that represents a query against the "Aggregated Data Access" layer – a virtual or physical (e.g., data warehouse) aggregation of all provisioning data. The resulting

provisioning plan is passed down to each local system to create a local image of the provisioning plan for fulfilling the order in the local geography.

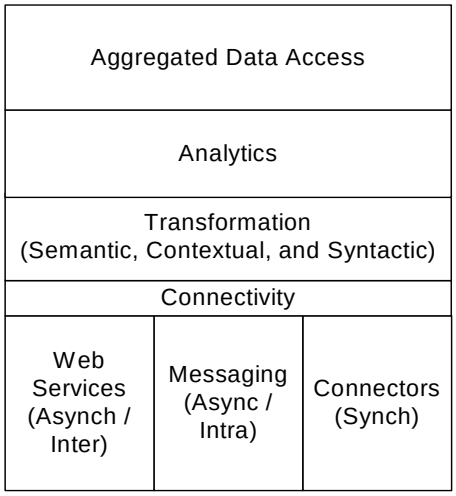

Figure 3. Aggregation Platform

In this scenario, the Aggregation Platform builds an aggregated image of the underlying data sources that can be accessed and queried through the "Aggregated Data Access" layer. Other layers in the technology stack perform the following functions.

The Analytics component assembles divisional provisioning plans into a coherent whole – removing data redundancy, resolving conflicts, and optimizing the resulting network structure.

The Transformation component handles standardizing the context and semantics of the information contained in the XML provisioning documents received from local systems. For example, one system may represent bandwidth in bits per second, while another may use megabits per second. This transformation process is one component of business process aggregation that has not been standardized within the Web Services paradigm and is often one of the most difficult integration challenges to overcome.

For example, IC has no standard customer number for WW. Each local system that has been providing network services to local divisions of WW has their own customer number and other information (e.g., address, spelling of name). This is a challenge because the Billing System, for example, needs to aggregate usage data across all of WW and has no standard context (e.g., customer number) for accomplishing that. Often called the Corporate Household or Corporate Family Structure problem [16][17], the issue is that IC has been doing business with local branches and subsidiaries of WC for years under may different names (e.g., Worldwide Consultants, Inc., WC Tokyo Corp., etc.).

We will discuss this problem further in Section 7 when we discuss additional infrastructure that is needed to realize the full potential of Web Services.

The Connectivity Component uses the appropriate method for transmitting data between the divisional components and corporate system. In the IC example, this connectivity is accomplished with the Web Services paradigm using asynchronous exchange of XML documents (perhaps over a corporate message queuing system such as IBM's MQ Series).

5.1 Analytics

The Analytics component extracts data elements from the XML documents exchanged with the Web Services and puts them into a data structure (e.g., relational database) that can be accessed by the Aggregated Data Access module – perhaps as a type of data warehouse. Analytics also performs analysis that may be useful to decision making that is part of the business process. For example, a bid from one of IC's partners typically contains volume discounts and different pricing for different times of day. The Analytics component will run a model of projected end customer usage of that partner's services to get a projected cost for doing business with that partner. In this manner, it is used by the Event Handling and Workflow module to manage a new business process.

5.2 Transformation

The Transformation component transforms the incoming XML into a standard format with a shared semantics and syntax. For example, if bids come in local currencies, the Transformation component will standardize on U.S. using a pre-determined exchange rate.

5.3 Connectivity

The Connectivity component handles the Web Services function calls using the standards discussed above (e.g., SOAP, XML, WSDL). In addition, an Aggregation architecture would typically provide two other methods for exchanging information with the sources being aggregated. One would be a messaging interface, employing something like IBM's MQ Series for asynchronous communication that is intra-organizational. The other would be a connector interface that provides synchronous connections with intra-organization enterprise computing platforms (e.g., SAP, PeopleSoft). Such connectors may be implemented using standards such as Java's J2EE Connector Architecture.

6 What is New About Aggregation using Web Services?

Aggregation has been going on long before Web Services standards emerged. What is new is the advent of universally accepted standards for accessing information from heterogeneous sources. These standards will have a profound impact on aggregation and on systems development in general. While early successful Aggregators like Yodlee focused primarily on aggregating data, the next generation of Aggregators will be able to aggregate business processes to create new business models faster and more cost effectively than ever before.

6.1 Standards Make Developing Aggregation Solutions Easier

The standards discussed in Section 4 make aggregation easier because they provide programmers with a common set of productivity tools to work with. Such tools allow developers to spend less time resolving data syntax issues and more time on semantic challenges. For example:

- Fast, easy to use XML parsers are available (e.g., Apache's Xerces and Xalan[13]) for a wide range of programming languages.
- HTTP has become a nearly universally available transport protocol. Where higher fault tolerance is required, standards, such as Java Message Service[14] (JMS), are now available as a common interface to most MOM products.
- SOAP eliminates the need for developers to learn and work with vendors' proprietary data transport protocols. As it matures, most commercial MOM and data integration products are supporting SOAP. SOAP's primary drawback is that, as a text-based protocol, it requires higher bandwidth than the proprietary binary protocols used in many vendor solutions. This issue comes up primarily in high volume, transaction oriented messaging systems and is not as important for typical data aggregation solutions.
- WSDL provides a standard form of documentation to developers who need to write code accessing multiple web services. This reduces the learning curve usually associated when dealing with APIs or other kinds of interfaces to multiple systems.

Lastly, a standard architecture, as show in Figure 3, enables the aggregation problem to be broken down into component parts that can then be "plugged in" to the architecture. For example, one could develop an analytics engine specifically designed for solving semantic issues in financial data aggregation and plug it in to one of the commercial products designed around this architecture. Again, this frees up developers to focus energy on the semantic challenges of financial information aggregation rather than the systems integration challenges of building a custom data aggregation architecture.

6.2 Comparison with Electronic Data Interchange (EDI)

What have enabled this change are the ubiquity of the Internet and the standardization of syntax (XML) and protocols for exchanging information. As a forerunner to Web Services, EDI provided standard protocols and syntax, but required the installation and maintenance of a network linking buyers and suppliers. Today, nearly all businesses have Internet access, and Web Services standards promise to enable much richer business-to-business interaction than EDI.

6.3 Implications of Web Services for Aggregation

There are some key differences between Aggregation via Web Services and traditional approaches:

[13] See http://xml.apache.org
[14] See http://java.sun.com/products/jms/

- *Ease of use* – a great deal of early aggregation (e.g., www.maxmiles.com for aggregating frequent flyer information) was accomplished through screen scraping of web sites that were not designed to be aggregated. Since individual web services will design themselves to be aggregated, process aggregation should be much easier.
- *Standards Based* – aggregation will be facilitated by the acceptance of standards for the exchange of information (e.g., SOAP, XML, WSDL) unlike the early data Aggregators that relied primarily on screen scraping approaches. Also syntax standards like ebXML will reduce custom coding for translation.
- *Products* – early aggregation systems were custom developed by the Aggregators. Because aggregation is now recognized as a potentially large market (e.g., Supply Chain Integration), major software vendors are releasing products that are specifically designed to support aggregation/integration via Web Services. Microsoft's BizTalk[15], IBM's WebSphere Business Integrator[16], and BEA Systems' WebLogic Integrator[17] are examples of such products.

6.4 Cooperative Business Models

Process aggregation will focus less on disinter mediation (e.g., MaxMiles) and more on cooperative models for working with the aggregated enterprises. This is a direct result of the need for permission to access an aggregated enterprise's Web Services in most cases. Hence, like in the IC example given above, we are most likely to see process aggregation used in areas like Supply Chain Management, where multiple organizations need to coordinate business processes.

In addition, we are likely to see a number of "open" free Web Services that can be accessed by anyone over the Internet. Although it is not clear what business model would support such services, a number already exist. Some of these are curiosities (e.g., a prime number tester at http://spheon-jsoap.sourceforge.net/webservices.php#isPrimeNumber). Others have more real business benefit (e.g., a credit card authorization service at http://sal006.salnetwork.com:83/userman/ccard/ccard.htm). See www.xmethods.com for a list of free Web Services.

Finally, we are seeing that major software vendors like SAP are beginning to offer Web Services interfaces to their products free of charge. In most cases, these Web Services are simple wrappers around the product's existing APIs.

6.5 Better Return on Investment (ROI)

Process aggregation will improve the ROI of systems integration, business process re-engineering, and B2B applications development. That is because the Web Services standards and the vendor products supporting them via the aggregation architecture illustrated above will make it much faster and easier to aggregate business processes. Many of theses complex integration tasks can now be reduced to defining XML interfaces

[15] www.microsoft.com/biztalk/default.asp
[16] http://www-4.ibm.com/software/webservers/btobintegrator/index.html
[17] www.bea.com/products/weblogic/integration

between an aggregator and an aggregated system. In addition, reusability will be improved, because once a Web Services interface is developed, it can be used by multiple integrators.

7 Challenges and Potential Research Directions

Today's Web Services standards specify common protocols for the exchange of information between systems. Other efforts, like ebXML (www.ebxml.com), target the standardization of syntax and protocols to standardize common business transactions (e.g., invoicing). However, there are still many significant challenges that remain in order for the Web Services paradigm to meet the integration requirements of many aggregation challenges. These challenges are summarized in the table below and explained in the following sub-sections.

Challenge	Brief Description
Semantics	Different Web Services will have different meanings attached to data values that may have the same, standard, name in each service. The challenge is to mediate between these different contexts.
Modularization of Business Processes	Existing EIS solutions (e.g., SAP) are monolithic and not easy to break into modular pieces of functionality to facilitate "best of breed" computing.
Security and Trusted Intermediaries	What methods will be most effective for ensuring that only authorized users can access a Web Service? Conversely, how does a user ensure that a Web Service does not misuse information that is exchanged during interaction?
Quality and Source Selection	The challenge is to ensure that a Web Service is providing accurate, complete, consistent, and correct information. Given the potential for multiple Web Services providing similar capabilities, how select most appropriate source?
Licensing and Payment Mechanisms	How will users pay for access to Web Services?
Development Tools	What kind of tools (e.g., modeling, programming, search) will be needed to make Web Services development efficient?

7.1 Semantics

Web Services Description Language (WSDL) is used to specify the XML syntax required to communicate with a Web Service. However, problems can still arise related to inconsistent meanings, or semantics.

Consider, for example, a Web Service provided by each of Global Telecom's divisions to return bandwidth data when queried about a particular customer's network connection between two points. One division's Web Service may represent bandwidth in bits per second, while another may use megabits per second. This transformation process has not been standardized within the Web Services paradigm and is often one of the most difficult integration challenges to overcome.

The bandwidth problem can be solved by defining a new type, called "mbs" for "megabits per second," and then using this type for the variable Bandwidth. Assuming that the programmers writing this Web Service in each division implement the WSDL specification correctly, then each would convert their units for bandwidth into megabits per second.

Some semantic problems, like the bandwidth units, can be overcome by specifying unique types. However, this is not always possible or practical. Consider a Web Service provided by each division that requires a "customer number" to retrieve local usage information for corporate billing purposes.

Commonly, organizations like IC do not have standard customer numbers for their clients. For example, each local system that has been providing network services to local divisions of WC probably has its own customer number and other information (e.g., address, spelling of name). This is a challenge because the Billing System, for example, needs to aggregate usage data across all of WC and has no standard context (e.g., customer number) for accomplishing that. Often called the Corporate Household or Corporate Family Structure problem[16][17], the issue is that IC has been doing business with local branches and subsidiaries of WW for years using a variety of customer numbers. Importantly, even XML schema standardization efforts like ebXML do not solve this Corporate Household problem.

7.1.1. Context Mediation

One solution may be to introduce Context Mediation into the Web Services paradigm [6][7]. In the IC example, a Context Mediation Service would identify and resolve potential semantic conflicts between the user and provider of a Web Service.

An example of such a Context Mediation framework is provided by MIT's COntext INterchange (COIN) project [2][7][8][9][10][11]. Following the COIN model, with the Web Services framework there would be standards to supply:

- A Domain Model to define rich types (e.g., customer number).
- Elevation Axioms to apply the Domain Model to each Web Service and define integrity constraints specifying general properties of the Web Service.
- Context Definitions to define the different interpretations of types in each Web Service (e.g., CustomerName might be "division level" or "corporate level").

The W3C is doing similar work in the context of its "Semantic Web" initiatives (www.w3.org/2001/sw/) that could be leveraged to provide standards for this type of Context Mediation. For example, a Domain Model standard could be defined as a subset of XML Schema (www.w3.org/XML/Schema).

Another approach to adapting the COIN model for Context Mediation to Web Services is suggested by the work being done on RuleML [14][15]. RuleML is XML syntax for rule knowledge representation. It is designed to be inter-operable with commercially important families of rule systems such as SQL, Prolog, Production rules, and Event-Condition-Action rules (ECA). For example, the Elevation Axioms used by COIN to mediate different contexts could be stored in RuleML in a Web Service's WSDL, or in a local UDDI directory.

If there were clear standards for these components of Context Mediation, then the vendors providing Aggregation tools, with architectures like that exhibited in Figure 3, could build Context Mediation capabilities into their products just as they have built in support for Web Services standards like SOAP, WSDL, and UDDI.

7.2 Modularization of Business Processes

It may prove very difficult to modularize the business processes, as automated in EIS packages like SAP and Siebel. Apart from the programming challenges related to adding Web Services features to these products, there are ontological challenges to modularization.

For example, at IC, many of the divisions have Order Management Systems that automatically generate a new customer in the local Billing System each time a new order is provisioned. The databases behind these Order Management Systems often enforce referential integrity between orders and the customer database in the Billing System. So, to avoid rewriting a lot of code in order to aggregate these local systems, the Enterprise Order Management System will need to add customer information to each of the local Billing Systems. But this customer information will also reside in the Enterprise Billing System, so we now need to maintain consistency across all these systems, and modify the local Billing System to not bill the local division of WW directly, but to roll-up local usage from WC to the Enterprise Billing System.

7.3 Security and Trusted Intermediaries

Publishers of Web Services on the Internet will need a security mechanism to control that is able to access their services. For example, access to a person's credit history should only be available to those with the legal right to obtain that information.

There are several ways that standards could be created, and infrastructure developed to build security into the Web Services paradigm. One possibility is simple password protection. In order to use a particular Web Service one would have to register and receive a user name and password.

Another possibility is to use Public Key Encryption as the basis for a security standard. In this model, anyone would be able to access a Web Service, but the XML documents returned by the service would be encrypted and only authorized users, with the proper key would be able to de-crypt them.

Ensuring the security of a Web Services user is another important consideration. For example, suppose that a company created a Web Service that provided an artificial intelligence based disease diagnosis. For a fee, a customer (or the information systems at a customer's hospital) could supply medical history and symptoms and receive back diagnostic information. Doctors to confirm diagnoses, insurance companies to validate treatments prescribed by doctors, and individual patients themselves, might use such a Web Service. To use such a system, a patient's medical history must be supplied to the Web Service. Clearly, the patient would want to ensure the confidentiality of that information, and also ensure that the company providing the Web Service did not even have access to the information provided.

In this scenario, it might make sense for the user of a Web Service to work through a "trusted intermediary" - an entity that could access Web Services on behalf of the customer and ensure that confidential information is not revealed to the operator of the Web Service.

7.4 Quality and "Source Selection"

Another important issue in the development of the Web Services paradigm is information quality. How does a customer know, for example, that a linear equation solving Web Service is providing correct answers?

Solving this problem (i.e., ensuring the accuracy, consistency, completeness, etc. of results obtained from a Web Service) is difficult. One possibility is the emergence of Web Services auditors that give their seal of approval to individual Web Services much the way that Public Accounting firms audit a company's financial results. Along these lines, the W3C has recently announced the creation of a Quality Assurance (QA) Activity (www.w3.org/QA/). Perhaps some of these issues will be addressed in that forum.

7.5 Licensing and Payment Mechanisms

Suppose you were developing a Financial Advisor site. To offer a complete set of services to customers, you might want to access Web Services for things like stock quotes, yield curve calculations, risk-arbitrage models, etc. One payment scenario would involve you signing licensing agreements with each Web Service – perhaps paying a monthly fee.

Another approach could be a "per use" charge, so that you were charged a small amount each time you accessed the Web Service. The market for Web Services would be helped by the existence of a standard "per use" payment services. If both the Web Services and the Financial Advisor aggregator were members, then the charges would be computed and handled automatically. The service would act as an intermediary, providing monthly statements to the aggregator, collecting fees, and sending payments to the Web Services. One commercial platform that has the potential to become such a service is Microsoft Passport[18].

7.6 Development Tools for Aggregation

To build a system using Aggregation and the Web Services paradigm, developers need tools to locate the Web Services they need to aggregate into their application.

To enable this kind of search, first a language is needed to describe the process that a Web Service is needed for. Perhaps the Unified Modeling Language (UML) could be adapted to this purpose to create a Unified Modeling Language for Web Services (UMLWS).

This is another area where knowledge representation efforts such as RuleML could be helpful. For example, the use of a particular Web Service is probably subject to a number of constraints that may or may not make it suitable for a particular task. Going back to our

[18] See www.passport.com

example, suppose that each division of IC has a "minimum order size" expressed in terms of bandwidth or length of contract. These rules could be expressed as RuleML and stored in the WSDL so that a developer could determine whether or not the Order Management System's Web Service at a particular division can be used for a particular order or not.

Once standards such as UMLWS and RuleML are devised and adopted, then Web Services Search Engines could be developed that take UMLWS and RuleML as input and search a UDDI directory for Web Services that provide the necessary processes.

8 Conclusion

The infrastructure is falling in place to enable great efficiencies of data integration, both internally within an organization (EAI) and externally across organizations (B2B). The ubiquity of the Internet, along with standardization on TCP/IP and HTTP create near universal connectivity. But connectivity is only the first step toward integration. Today, the Web Services paradigm promises to standardize the syntax and protocols used for communication between applications. This is another important step toward facilitating data integration. However, it is important to remember that many challenges lie ahead. A good first step for researchers would be to implement a prototype aggregation system using commercially available software products and the architecture described in this paper. This would provide a concrete demonstration of the degree to which syntax and protocol challenges have been solved.

However, as the problems of syntax and protocols for integration get resolved, we will find ourselves facing the additional challenges of semantics, modularization of business process, security, and other issues discussed in this paper. It will be interesting to see how work that has been done on Context Mediation, the Semantic Web, and other areas can be applied to meet these challenges.

REFERENCES

[1] Madnick, S (1999). "Metadata Jones and the Tower of Babel: The Challenge of Large-Scale Semantic Heterogeneity", *Proc. IEEE Meta-Data Conf.*, April 1999.

[2] Madnick, S. (2001). "The Misguided Silver Bullet: What XML will and will NOT do to help Information Integration", *Proceedings of the Third International Conference on Information Integration and Web-based Applications and Services (IIWAS2001)*, September 2001.Madnick, S., Siegel, M. Frontini, M., Khemka, S., Chan, S., and Pan, H., "Surviving and Thriving in the New World of Web Aggregators", MIT Sloan Working Paper #4138, October 2000 [CISL #00-07].

[3] Bressan, S., Goh, C., Levina, S., Madnick, S., Shah, A., and Siegel, M., "Context Knowledge Representation and Reasoning in the Context Interchange System", *Applied Intelligence* (13:2), Sept. 2000, pp. 165-179.

[4] Hackathorn, R (1999). *Web Farming for the Data Warehouse*, Morgan Kaufmann Publishers.

[5] Goh, C. (1996). *Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Systems*, PhD Thesis, MIT, June 1996.

[6] Tomasic A., Raschid L., and Valduriez P., "Scaling Access to Heterogeneous Databases with DISCO," in IEEE Transactions on Knowledge and Data Engineering, 10(5), 1998

[7] Goh, C., Bressan, S., Madnick, S., and Siegel, M. (1999). "Context Interchange: New Features and Formalisms for the Intelligent Integration of Information," *ACM Transactions on Office Information Systems*, July 1999.

[8] Goh, C., Bressan, S., Levina, S., Madnick, S., Shah, A., and Siegel, M. (2000). "Context Knowledge Representation and Reasoning in the Context of Applied Intelligence," *The International Journal of Artificial Intelligence, Neural Networks, and Complete Problem-Solving Technologies*, Volume 12, Number 2, Sept. 2000, pp. 165-179.

[9] Goh, C., Madnick, S., and Siegel, M. (1994). "Context Interchange: Overcoming the Challenges of Large-Scale Interoperable Database Systems in a Dynamic Environment," *Proceedings of the Third International Conference on Information and Knowledge Management*, pages 337-346, Gaithersburgh MD.

[10] Siegel, M. and Madnick, S. (1991) "Context Interchange: Sharing the Meaning of Data," *SIGMOD RECORD*, Vol. 20, No. 4, December pp. 77-78.

[11] Siegel, M. and Madnick, S. (1991) "A Metadata Approach to Solving Semantic Conflicts," *Proceedings of the 17th International Conference on Very Large Data Bases*, pages 133-145.

[12] Hansen, M., "Changing Terrain: Open middleware standards are redefining EAI and B2B integration", *Intelligent Enterprise*, August 10, 2001.

[13] Moulton, A., Bressan, S., Madnick, S. and Siegel, M., "An Active Conceptual Model for Fixed Income Securities Analysis for Multiple Financial Institutions," *Proc. ER 1998*, pp. 407-420.

[14] Grosof, B. and Labrou, Y., "An Approach to using XML and a Rule-based Content Language with an Agent Communication Language." In Frank Dignum and Mark Greaves, editors, *Issues in Agent Communication*. Springer-Verlag, 2000.

[15] Grosof, B., "Standardizing XML Rules: Preliminary Outline of Invited Talk", *Proceedings of the IJCAI-01 Workshop on E-business and the Intelligent Web*, edited by Alun Preece, August 5, 2001.

[16] Chen, X., Funk, J., Madnick, S., and Wang, R., "Corporate Household Data: Research Directions", *Proceedings of the Americans Conference on Information Systems* (AMCIS, Boston), August 2001 [SWP #4166, CISL WP #01-03, TDQM WP#2001-08].

[17] Madnick, S., Wang, R., Dravis, F., and Chen, X., "Improving the Quality of Corporate Household Data: Current Practices and Research Directions", *Proceedings of the Sixth International Conference on Information Quality* (IQ2001, Cambridge), November 2001, pp. 92-104 [CISL #01-10].