Process Aggregation Using Web Services

Mark Hansen, Stuart Madnick, and Michael Siegel

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

Abstract. This paper examines the opportunities and challenges related to data and process integration architectures in the context of Web Services. A primary goal of most enterprises in today's economic environment is to improve productivity by streamlining and aggregating business processes. This paper illustrates how integration architectures based on Web Services offer new opportunities to improve productivity that are expedient and economical. First, the paper introduces the technical standards associated with Web Services and provides business example for illustration. Abstracting from this example, we introduce a concept we call Process Aggregation that incorporates data aggregation and workflow to improve productivity. We show that Web Services will have a major impact on Process Aggregation, making it both faster and less expensive to implement. Finally, we suggest some research directions relating to the Process Aggregation 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.

1 Introduction

Web Services, a programming paradigm for integrating heterogeneous information systems, offers significant advantages over the currently available set of ad-hoc methods based on proprietary software 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 data integration over corporate intranets once these standards are supported by the information systems underlying a corporation's 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]

Given the recent surge of interest in Web Services within industry, it is appropriate to look at this paradigm from a research standpoint and determine what we can learn by comparing Web Services with other integration paradigms. In particular, we take the position that the integration of heterogeneous information systems using the Web

[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 3.2 defines these acronyms.

Ch. Bussler et al. (Eds.): WES 2002, LNCS 2512, pp. 12–27, 2002.

Services paradigm can be viewed as a form of aggregation, namely Process Aggregation.

Using that analogy, we investigate the challenges researchers have uncovered related to aggregation [1][2][3][4][7][13] and examine them in the Web Services context. Foremost among these challenges are the issues of semantics and context mediation.

2 Example of a Systems Integration Architecture Based on Web Services[3]

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

For example, consider the Order Management System (OMS) required by the corporate headquarters. When a global customer, such as Worldwide Consultants (WC), asks GT to bid on a contract to provide services, GT 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 OMS. The order is communicated to each subsidiary to develop a provisioning plan in their geography. The subsidiaries' plans are sent up to the corporate systems and integrated into a global provisioning plan. Integration of these heterogeneous subsidiary systems with the OMS requires both data and process integration. It also required integration with subsidiary support systems (e.g., Trouble Tickets, Usage Statistics). It is an example of what we call Process Aggregation[4].

2.1 Potential Solutions

GT considered a spectrum of alternatives for building a Process Aggregator for the OMS, summarized in the table below.

Integration Alternative	Description
Single System	This approach involves replacing all the divisional 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.

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

[4] Formal definition in Section 3.1.

GT wanted to implement the Single System alternative because it would standardize processes throughout the organization and reduce the amount of custom code development and maintenance required to interface corporate with divisional systems. However, there were several problems that prevented GT 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.[5] Additionally, GT was acquiring companies and needed a quick way to integrate them with corporate systems.

Considering these challenges, GT decided to implement a five-year plan to standardize divisional systems. In the mean time, GT decided to create custom interfaces between divisional and corporate systems. By building prototype Web Services interfaces for one division, GT determined that this approach leveraged local knowledge to quickly create the interfaces to the OMS. 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 OMS.

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"[3]. Much of this research focuses on the semantic and contextual challenges of aggregation [6][7], and as we will see in Section 5 many of these challenges remain when applying the Web Services paradigm to Process Aggregation.

Before getting into Process Aggregation, however, we should note that Web Services do 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" [5] 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 raises new challenges discussed in Section 5.

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 3.2 for a definition and discussion of these standards.):

---

[5] 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 …"

- 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 Order Management System and the systems in "Division A" are illustrated in Figure 1, such as Provisioning, Trouble Tickets, and Usage Statistics. Similar interfaces would be needed for all the divisions.

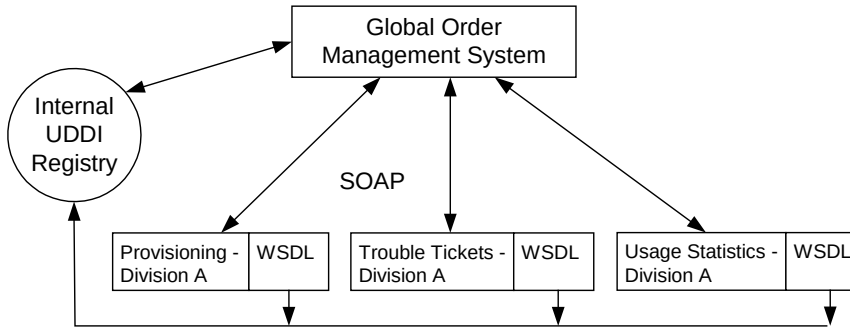

Fig. 1. Global Telecom's Web Services Interfaces

3 Process Aggregation and Web Services

The previous section illustrates how Web Services can be used to facilitate data integration and aggregation. However, to take Web Services a step further and enable Process Aggregation, we need to have workflow capabilities layered on top of Web Services interfaces. In this manner, Web Services plus workflow enable the aggregation of business processes. That is, creating a new business process by linking together existing business process components in a manner that is orchestrated by a workflow manager.

To begin exploring the challenges posed by the Web Services paradigm for integration, we introduce a concept we call Process Aggregation.

3.1 Process Aggregator Definition

A Process Aggregator is an entity that:

- Transparently collects and analyzes information from different data sources;
- Resolves the semantic and contextual differences in the information and services;
- Provides a single point of contact for managing a business process that requires coordination across a variety of services / information sources. (e.g., a multi-step workflow process)

It should be noted that almost any aggregator that accesses a source web site with a CGI (or similar) program behind it generating HTML could be thought of as aggregating processes. For example, Yodlee (www.yodlee.com) accesses account balance lookup processes at the source sites of its members. However, we define Process Aggregation to be the creation of a new business process through the aggregation of component sub processes that comprise a multiple step workflow.

GT's Order Management System, illustrated in Figure 1, is a good example of a Process Aggregator. Below, we describe the workflow aspects that distinguish it as an example of Process Aggregation.

3.2 Web Services Definition

The Web Services paradigm provides a new set of standards and technologies that facilitate an organization's ability to integrate internal heterogeneous systems (e.g., Enterprise Application Integration (EAI)) or integrate with business partners (e.g., Supply Chain Management and other Business-to-Business (B2B) type applications). These types of systems are Process 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[6]) documents are used for data input and output.
- HTTP (Hypertext Transfer Protocol[7]) or a Message Oriented Middleware (MOM) product (e.g., IBM's MQ Series) is the application protocol.
- SOAP (Simple Object Access Protocol[8]) is the standard specifying how XML documents are exchanged over HTTP or MOM.
- WSDL (Web Services Description Language[9]) is used to provide a meta-data description of the input and output parameters for the interface.
- UDDI (Universal Description, Discovery and Integration[10]) is used to register the Web Service.

3.3 Process Aggregation Using Web Services

Figure 2 illustrates a generic example of how Web Services standards are employed for Process Aggregation. This is a generic version of Figure 1 where the box labeled "Process Aggregator" is Global Telecom's Order Management System. The programmers developing this system need to integrate the Order Management Systems from 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.

---

[6] www.w3.org/XML
[7] www.w3.org/Protocols
[8] www.w3.org/2000/xp
[9] www.w3.org/TR/wsdl
[10] www.uddi.org

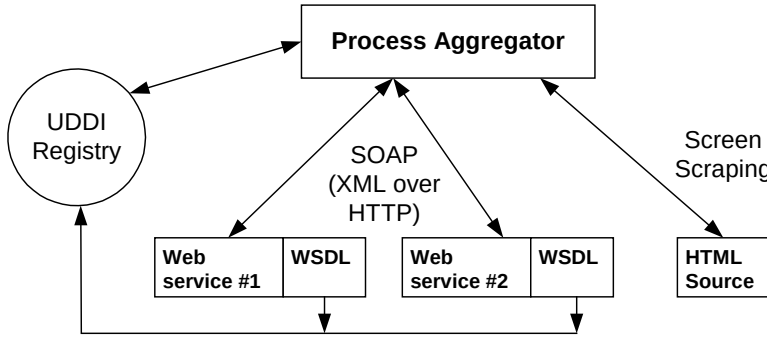

Fig. 2. Process Aggregation with Web Services

Within each division, programmers develop a Web Service that can receive an Order and return a Provisioning document. The interface for each division's Web Service is published using WSDL and registered in a UDDI Registry. The programmers working on the Global Order Management 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 Order Management Systems may be simple enough that instead of implementing a Web Service interface, basic screen scraping off an existing HTML interface is used.

3.4 Process Aggregator Architecture

In addition to data integration, a Process Aggregator combines services from a variety of sources to create and manage a new business process. A standard technical platform architecture is emerging for creating Process Aggregators, as illustrated in Figure 3. This platform architecture, with some variations from vendor to vendor, is used by a wide range of commercial products including Microsoft BizTalk Server[11], webMethods Integration Platform[12], TIBCO ActiveEnterprise[13], and IBM's WebSphere Business Integrator[14].

The Process Aggregation application built on such a platform is referred to as EAI if it involves aggregating internal processes (as in our GT example) or B2B if it involves aggregating business processes from different companies.

3.4.1 Process Manager

A Process Manager component sits on top of the technology stack and manages the business process that is created by aggregating a variety of sub-processes. This component handles events (e.g., request for bid), workflow (e.g., forwards bids to man-

[11] www.microsoft.com/biztalk/default.asp
[12] www.webmethods.com/content/1,1107,webMethodsIntegrationPlatform,FF.html
[13] www.tibco.com/products/enterprise.html
[14] http://www-3.ibm.com/software/webservers/btobintegrator/

agement for approval), and transactions (e.g., issues purchase orders for services based on bids). Note that this requires an embedded "workflow manager" as illustrated in Figure 4.

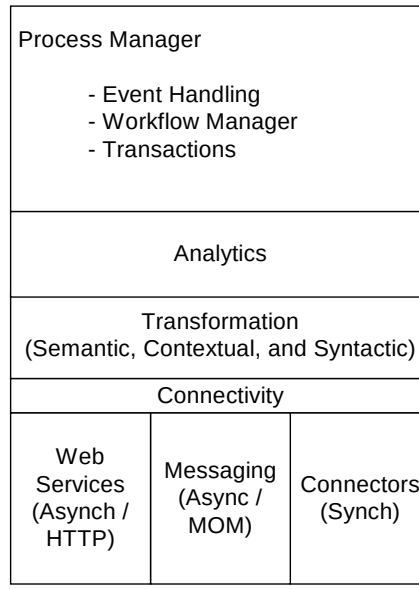

Fig. 3. Process Aggregator Platform

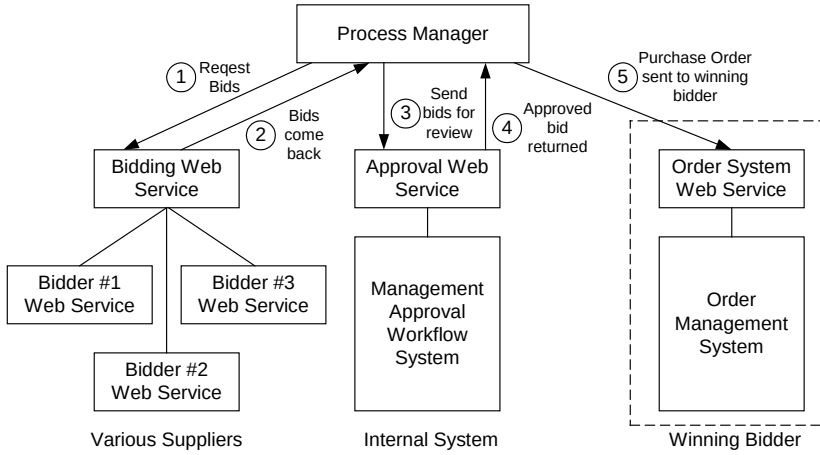

Fig. 4. Process Management Workflow

The steps in this workflow are:

1. Process Manager sends a SOAP message to the Bidding Web Service containing a widget order. The Bidding Web Service could be an independent Internet marketplace or an internal electronic marketplace that communicates with the corporation's widget suppliers.
2. The Bidding Web Services sends a SOAP message back to the Process Manager containing the bids from each supplier who decided to place a bid.
3. The Process Manager then send these bids, as a SOAP message via the Approval Web Service, to a workflow system that presents the bids to management and enables management to electronically approve the winning bid.
4. The winning bid is then sent back to the Process Manager, as a SOAP message via the Approval Web Service.
5. The Process Manager generates a Purchase Order and sends it to the winning bidder as a SOAP message to their Order Management System's Web Service.

3.4.2 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 Process Management component for managing the business process. Analytics also performs analysis that may be useful to decision making that is part of the business process. For example, the Analytics component might run a model of projected end customer usage of that partner's services to get a projected cost for doing business with that partner.

3.4.3 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. dollars using a predetermined exchange rate.

3.4.4 Connectivity

The Connectivity component handles the Web Services function calls using the standards discussed above (e.g., SOAP, XML, WSDL) over either HTTP or a MOM infrastructure. In addition, a Process Aggregator would typically provide a synchronous method for exchanging information with the processes being aggregated and where transactions need to be supported (e.g., rollback, commit). Such synchronous capabilities would be provided by a connector interface to the appropriate Enterprise Information System (EIS) (e.g., SAP, PeopleSoft). Connectors may be implemented using standards such as Java's J2EE Connector Architecture[15] or proprietary products.

3.5 Ford Motor Company's e-Hub[16]

One real world example of how this Process Aggregation Architecture is used in practice is Ford's e-Hub initiative. e-Hub will provide Ford with both EAI and B2B

[15] http://java.sun.com/j2ee/connector/index.html

[16] http://biz.yahoo.com/prnews/010726/sfth060.html

integration capabilities. Currently, e-Hub is being used for collaboration with dealers and suppliers, as well as supply chain integration. Ford Motor uses Microsoft BizTalk Server as the technology platform for the e-Hub Process Aggregation architecture.

3.6 A Prominent Systems Development Paradigm of the Future?

Process Aggregation using Web Services may become a prominent systems development framework for large corporations in the near future. The major problem corporations face in using EIS software is that its "one size fits all" approach to business process automation leaves customers with little flexibility to adapt the software to their business processes as the evolve, or automate new business processes for competitive advantage.

Process Aggregation enables corporate business process to be more flexible and respond to changing business needs. This is accomplished by modularizing systems functionality into Web Services and then arranging and re-arranging the workflow between modules to adapt to changing business requirements. To satisfy the need for such Process Aggregation, many major software vendors are now offering technology platforms that can be used to implement the architecture illustrated in Figure 3. Examples include Microsoft's BizTalk[17], IBM's WebSphere Business Integrator[18], and BEA Systems' WebLogic Integrator[19].

4 What Is New about Process Aggregation with Web Services?

Process Aggregation has been going on long before Web Services standards emerged. As mentioned previously, the aggregation of any HTML or XML data that is generated by a program (e.g., CGI), rather than being static, is doing some form of Process Aggregation. In this respect, there is really a continuum from aggregators that are clearly "information aggregators" (e.g., Yahoo), to those that are clearly Process Aggregators (e.g., A B2B system for supply chain management).

What is newly relevant to the Process Aggregation end of the continuum is the advent of universally accepted standards for Web Services. As discussed in Section 3.6, this will have a profound impact on aggregation and on systems development in general.

4.1 Comparison with Electronic Data Interchange (EDI)

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 broader business-to-business interaction than EDI.

---

[17] www.microsoft.com/biztalk/default.asp

[18] http://www-4.ibm.com/software/webservers/btobintegrator/index.html

[19] www.bea.com/products/weblogic/integration

4.2 Comparison with Distributed Object Paradigms (e.g., CORBA)

Distributed object paradigms have also been promoted as a method of easing application integration and promoting "object re-use" (i.e., reusing modules of code). Examples include Common Object Request Broker Architecture (CORBA), Microsoft's Distributed Component Object Model (DCOM) and .NET platform, and Java's J2EE framework.

There are many similarities between these initiatives and Web Services. Most prominent is the reliance on standards to facilitate communication between applications. However, the problem here is that most organizations have very heterogeneous sets of applications that don't adhere to one distributed object paradigm. Applications can be "wrapped" with a distributed object interface, but this is often costly and time consuming. Additionally, for B2B integration across enterprises, these models are less useful because (i) the communication protocols don't work through firewalls; and (ii) different enterprises use different object models.

Web Services represent a step forward because, at least for the moment, the software industry seems to be supporting the same set of standards. Secondly, the SOAP protocol for exchanging XML messages is not blocked by firewalls. Thirdly, Web Services are easier to implement than building distributed object wrappers around existing applications. In fact, most software vendors are planning to provide Web Services interfaces into their products out of the box, along with tool kits for further development of Web Services tailored to a particular customer's needs.

Both the .NET and J2EE paradigms now include extensive functionality to support Web Services.

4.3 New Software Usage Paradigms

Web Services has the potential to change the manner in which software is most commonly purchased and used today. One example of this would be the potential to pay for software on a per-use basis. For example, consider a Web Service for credit card authorization. Such functionality could be offered on a subscription or per-use basis by organizations operating web sites that need to process credit card transactions and don't want to build or buy software for that purpose.

Another example would be the ability of corporations to more easily implement a "best of breed" strategy when implementing EIS solutions. Since Siebel and SAP now have Web Services interfaces, perhaps a company could easily integrate Order Processing from SAP with Customer Service from Siebel and pay each vendor only for the functionality that they use.

Of course, this would require the software vendors to modularize their products so that they would work interchangeably with modules from other vendors. This is not likely considering that large EIS vendors, like SAP, want to sell a complete package. However, Web Services may provide an opportunity for new, third party, software vendors to provide such modularized products that work well with the existing monolithic EIS systems.

Finally, we may reach a point where potential software users are able to search a UDDI directory for Web Services components and assemble their own custom software tools from the aggregation of existing functionality. A Logistics Management

System, for example, might be assembled by aggregating route information from one Web Service with a route optimization algorithm from another Web Service.

5 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 architecture requirements of many Process Aggregators. 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 does one select the 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?

5.1 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. Assum-

ing 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 GT 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 GT has been doing business with local branches and subsidiaries of WC for years using a variety of customer numbers. Importantly, even XML schema standardization efforts like ebXML do not solve this Corporate Household problem.

5.2 Context Mediation

One solution may be to introduce Context Mediation into the Web Services paradigm. In the GT 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). Alternatively, Context Mediation metadata for a web service could be stored using UDDI tModels (www.uddi.org/pubs/DataStructure-V2.00-Open-20010608.pdf). Currently, tModels are used primarily for storing taxonomy data (e.g., NAICS industry codes), but the specification is flexible enough to be used for storing the rich metadata required for context mediation. Of course, the usefulness of storing context mediation metadata in tModels would depend on the development of standards for the metadata itself.

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 Process 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.

5.3 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 GT, 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 WC directly, but to roll-up local usage from WC to the Enterprise Billing System.

5.4 Security and Trusted Intermediaries

Publishers of Web Services on the Internet will need a security mechanism to control who 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. Such a Web Service might be used by doctors to confirm diagnoses, insurance companies to validate treatments prescribed by doctors, and individual patients themselves. 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.

5.5 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.

5.6 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[20].

5.7 Development Tools for Process Aggregation

To build a system using Process 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 [14][15] 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 example, suppose that each division of GT 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

[20] www.passport.com

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.

6 Conclusion

The infrastructure is falling in place to enable great efficiencies in the integration and aggregation of business processes, 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 that promises to enable Process Aggregation. However, it is important to remember that many challenges lie ahead. 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.
3. 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].
4. 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.
5. Hackathorn, R (1999). *Web Farming for the Data Warehouse*, Morgan Kaufmann Publishers.
6. Goh, C. (1996). *Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Systems*, PhD Thesis, MIT, June 1996.
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].