NET Institute*

www.NETinst.org

Working Paper #03-14

October 2003

Orchestrating Web Services for Networked Enterprise Collaboration

Ananth Srinivasan and David Sundaram

Center of Digital Enterprise
Department of Management Science and Information Systems
University of Auckland Business School
Auckland, New Zealand

* The Networks, Electronic Commerce, and Telecommunications ("NET") Institute, http://www.NETinst.org, is a non-profit institution devoted to research on network industries, electronic commerce, telecommunications, the Internet, "virtual networks" comprised of computers that share the same technical standard or operating system, and on network issues in general.

ORCHESTRATING WEB SERVICES FOR NETWORKED ENTERPRISE COLLABORATION[1]

Ananth Srinivasan
and
David Sundaram

{a.srinivasan,d.sundaram}@auckland.ac.nz

Center of Digital Enterprise
Department of Information Systems and Operations Management
University of Auckland Business School
Auckland, New Zealand

[1] This project was partially funded by a grant from the Net Institute, New York.

ORCHESTRATING WEB SERVICES FOR NETWORKED ENTERPRISE COLLABORATION

Abstract

Ananth Srinivasan
and
David Sundaram

Center of Digital Enterprise
University of Auckland Business School
Auckland, New Zealand

Internet technologies are widely recognized for their promise as enablers of collaborative computing both within and among organizations. The presence of heterogeneous systems based on different technological platforms in organizations makes the implementation of network collaboration very complex. The approach taken for the most part to deal with this issue has been based on Enterprise Application Integration. The major drawback of this approach is the dependence on proprietary solutions that are not based on open standards. When the need for inter-organizational collaboration arises, such solutions hinder the smooth exchange among the participating organizations due to their complexity and lack of interoperability.

"Web Services" is a new class of internet based, open standards technology that offers the promise of resolving these problems. Web services technologies are offered as the new generation of electronic commerce enablers. What is currently missing is a compelling implementation framework for the deployment of this technology in organizations. A lack of clear understanding about how to deploy Web Services to enable inter-organizational collaboration will impede the uptake of this promising new technology. In this project, our aim is to construct and test a valid implementation framework for Web Services. Such a framework will enable effective inter-organizational network based computing which will have a positive effect on organizational productivity. The emphasis of our work will be to support decision oriented, collaborative business processes.

We propose a conceptual framework for building systems that utilize Web Services technologies to enable networked organizations to automate collaborative processes. We take the view that it is useful to conceptualize organizational service chains in terms of workflows and build a framework for orchestrating such services. We articulate a conceptual framework along these lines and present specific ways by which the implementation of the framework using Web Services technologies could benefit inter-organizational collaboration. Our primary objective of building prototypes that demonstrate the validity of such a framework was successful. Two prototypical systems that demonstrate the validity of such a framework were built. They now offer a platform for ongoing testing and refinement. We believe that such prototypes will serve as useful tools for organizations that are investigating the use of this emerging technology to facilitate networked collaboration to improve productivity.

Key Words: Web Services Orchestration; Inter-organizational systems; Enterprise collaboration

Introduction and Problem Background

Internet technologies are widely recognized for their promise as enablers of collaborative computing both within and among organizations. The presence of heterogeneous systems based on different technological platforms in organizations makes the implementation of network collaboration across organizational boundaries very complex (Stal, 2002). The approach taken for the most part to deal with this issue has been based on Enterprise Application Integration (McKeen and Smith, 2002). The major drawback of this approach is the dependence on proprietary solutions that are not based on open standards (Fremantle, et al 2002). When the need for inter-organizational collaboration arises, such solutions hinder the smooth exchange among the participating organizations due to their complexity and lack of interoperability. "Web Services" is a new class of internet based, open standards technology that offers the promise of resolving these problems (Gottschalk, *et al*, 2002; Johnston, 2002). What is currently missing is a compelling implementation framework for the deployment of this technology in organizations. A lack of clear understanding about how to deploy Web Services to enable inter-organizational collaboration will impede the uptake of this promising new technology. In this project, our aim is to construct and test a valid implementation framework for Web Services. Such a framework will enable effective inter-organizational network based computing which will have a positive effect on organizational productivity. The emphasis of our work will be to support decision oriented business processes.

Web Services: The Essentials

Web Services promise cheap, reliable, flexible and scalable computing. Further, their deployment and maintenance is not very complex. Though much hype still surrounds them most practitioners agree that they offer a significant step forward in leveraging the power of the Internet. Web services have received wide spread support from industry leaders. These organizations have been instrumental in driving the setting of standards and specifications for the development of Web Services and their related technologies. This task is overseen currently by the W3C Working Groups (http://www.w3c.org/2002/ws). Another possible reason for this enthusiasm is summarized well by Johnston (2002), who suggests that while integrating technologies already exist, limitations such as complexity and verbosity have impacted their widespread use. Web Services address these issues by steering a middle road between complexity and verbosity.

A brief definition of Web Services highlights the following: they are applications (operations and tasks) that can be developed in modular, independent fashion and made available through their interfaces across heterogeneous systems using standard internet protocols (Ferris and Farrell, 2003; Gottschalk, et al, 2002) . A Web Service is less a single tangible thing than an interwoven mesh of technologies and standards that work together to achieve their shared goals. A Web Service architecture requires three fundamental operations: publish, find, and bind (Gottschalk, 2000). Service providers publish services to a broker while service requesters find required services using a broker and bind to them (Figure 1).

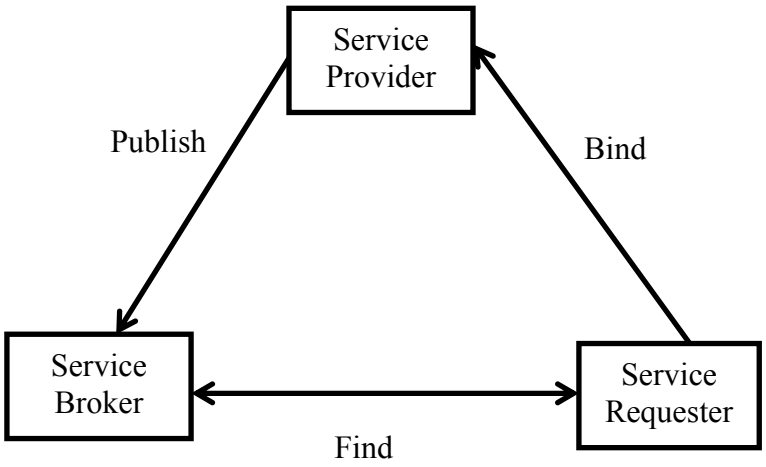


Figure 1: Publish, Find, and Bind (Source: Gottschalk, 2000)

Each of these fundamental operations is enabled through one or more of the several technologies and standards that we encompass in the term "Web Services". At the heart of Web Services lies XML. The Web Service Description Language (WSDL), and the Simple Object Access Protocol (SOAP) are two XML-based technologies that Web Services use. WSDL is used to publish a Web Service and act as an instruction manual for binding to it. SOAP is used to encode and transfer messages to and from Web Services.

"*WSDL offers an XML grammar for describing Web Services as collections of communication endpoints capable of exchanging messages*" (Christensen et al., 2001). WSDL definitions serve as a guide to automate the communication of distributed applications. A WSDL document defines services as collections of network endpoints/ports. In WSDL, a port is defined by associating a network address with a reusable binding. A collection of such ports define a service.

SOAP, like WSDL, is also based on XML. SOAP has been designed as a lightweight protocol for exchanging information in a distributed environment (Box et al., 2000). Web Services use SOAP messages to communicate. A SOAP message consists of three parts 1) an envelope that defines a framework for describing what is in a message and how to process it 2) a set of encoding rules for expressing instances of application-defined data types and 3) a convention for representing remote procedure calls and responses. The goal of SOAP is to break down the barriers between distributed computing platforms. It accomplishes this through simplicity, flexibility, platform neutrality, and finally by being text-based. Thus SOAP allows distributed computing communications over the Web to be standardized.

Examples of Web Services Applications

To illustrate important dimensions of the use of web services, we describe the deployment of the technology in a variety of organizational settings. These examples help us better understand the full range of issues that arise in the use of the technology and the specific organization processes that are impacted by them. They also help us begin to articulate a sensible framework that captures the various dimensions in a manner that fosters systematic development of ideas for web service implementation.

General Motors spends about $4 billion a year on acquiring new software (Business Week Online, June 24, 2003). In an effort to curtail expenditure on the acquisition of new software, GM is deploying web services to upgrade existing software and integrate applications across its many systems. The technology is being used to act as a bridge between a diverse set of systems within the organization. With increasing amounts of software placed in

vehicles themselves, the objective in the future is to use web services to upgrade software that is resident in these vehicles. The technology can help cut the cost associated with upgrading such software. Another useful application that GM sees for the technology is to connect its parts and inventory data in flexible ways with its suppliers. Once issues related to security are resolved satisfactorily, the push will be toward tying together data from a diverse set of systems that span organizational boundaries.

Putnam Lovell Securities, an investment bank, uses web services to automate the process of customizing the content of their research department to suit the particular needs of their client base (Nghiem, 2002). The emphasis on the use of web services was on integrating a variety of applications. Both information about customers and research content of interest to their clients are applications that reside outside the organization and are accessible through the internet infrastructure. The leverage gained by web services technologies is the automation of the collection of research content distributed by multiple sources and the matching of the content to the specific requirements of the investment clientele. Real value is added by conforming to the specific formatting requirements of various clients whether they are investors or distributors of research content. For example, individual clients can receive tailored information by email; a distributor can get the same information in a portable document format. Web services has helped the organization reduce the costs, increased the level of personalization, and improved the speed with which information gathered from multiple sources external to the organization are distributed to clients.

Pantechnik International is a European based organization that has implemented web services technologies in the logistics and transportation industries (Nghiem, 2002; http://www.diffuse.org/Presentations/PeterNicholls.ppt). The company offers logistics information services to connect large suppliers or distributors with their carriers and customers. The company offers relevant information that is accessible in a transparent manner to any of the parties in the supply chain in a manner that is tied to the internal business processes of these organizations. Providing relevant information seamlessly across organizational boundaries that helps organizations connect their systems with each other is the value added by Pantechnik. The business processes that are defined within its platform are at an appropriate level of detail to enable participant organizations to create relevant data by matching and integrating a set of automated processes. The combination of integrating data from diverse systems in a manner that allows clients to orchestrate multiple processes to reflect transactions are hallmarks of the effective use of web services technologies by Pantechnik.

Talaris Corporation (Nghiem, 2002; http://www.talaris.com) is an excellent example of the use of web services technologies to deploy a services based procurement platform for client organizations. Web services technologies are used to maintain a private registry of service providers and detailed profiles of individual clients. When a client organization requires a business service such as a meeting facility or an airline reservation, the registry is processed in a way that matches the specific requirements of the client organization and a solution is offered. The strengths of the application in Talaris lie in (a) compliance with specific client organization procurement policies (b) knowledge of individual client profiles to produce a personalized solution and (c) offering services from suppliers who are approved by client organizations.

There are many common threads that run through the various applications described above. First is the ease of integration that is offered by web services technologies. The use of open, non-proprietary standards that can be deployed over the internet is perhaps the prime catalyst that encourages its use. Second, the ability to integrate diverse systems regardless of whether they are within a single organization or span organizational boundaries addresses a problem that has plagued organizations for many decades. Third, the ability to provide personalized solutions to information needs based on a deep understanding of client profiles makes it possible to accurately align specific business processes with deployment of the technology. Perhaps the key mitigating factor lies in issues related to the security aspects of transporting potential sensitive information across organizational boundaries. Currently, many organizations are dealing with this issue with customized solutions until security standards are well established within web services technologies. The focus of this paper however, is on investigating the variety of applications that are enabled by web services technologies by approaching the design problem in a systematic manner.

Web Services: High Level Issues

Web Services exist as a collection of technologies and standards rather than any individual technology or standard. It is useful to consider a range of high level issues that enable us to better understand the design implications of constructing Web Services in organizational contexts. In this section, we integrate several observations that have been made with regard to Web Services applications that provide a useful identification about what these issues are.

The first issue relates to considerations of how Web Services need to be managed with specific regard to collaboration among enterprises. Kreger (2003) mentions the importance of quality of service, security, and management of Web Services as critical infrastructural concerns in the delivery of Web Services. Some of the operational issues that arise as a result of this are discussed here. The binding of a Web Service refers to how loose or tight its coupling with other Web Services is. If Web Services are loosely coupled, we call them ad-hoc. Tightly coupled Web Services are considered negotiated. Web Services may be published via a federated manager (e.g. a UDDI registry), which automates to a large degree their discovery and brokerage. On the other hand they may be independently published to be discovered by explicit notification or chance. The reach of a Web Service refers to its ability to 1)be used by other organizations and 2) use other Web Services.

A second issue relates to the **complexity** and **dynamism** of the nature of the service that is provided. A Web Service may perform a range of tasks that vary from complicated functions that return manipulated results (such as the valuation of options as provided by http://www.xignite.com), to simply acting as 'dumb' data storage (Andrews, 2003). Further, Web Services may provide result updates in response to repeated execution of a particular query, thereby exhibiting its dynamic nature (Berry, Chase, Cohen, Cox, and Vahdat, 1999).

A third issue relates to multiple embedded services to support a single business process. We refer to this issue as the **depth** of a Web Service in terms of the direction that it provides for purposes of alignment with a business process. Is the Web Service an aggregation of other

nested Web Services (Bloomberg, 2002), or does it stand on its own? Often this dimension is transparent to users of the service.

A fourth issue relates to the ability of Web Services to leverage existing functionality in organizations by providing streamlined application **integration** pathways. Web Services may be entirely new pieces of functionality, or they may act as wrappers for existing functionality. When used as wrappers, Web Services expose the API's of existing (legacy) components to give them the advantages that Web Services have to offer without having to recode existing functionality. This aspect is very significant in the context of Enterprise Application Integration (EAI) and seamless interoperability of software from diverse systems (Almeida, Pires, Sinderen, 2003).

Two final issues that relate to Web Services applications and/or environments that have been developed so far are **versatility** and **usability**. Versatility is the ability of a product to be used for a variety of applications, support different paradigms, operate in multiple application domains, etc. Usability is the ability of a product to be effectively used by a community that possesses a broad range of skills. High usability is usually achieved at the expense of versatility but this could have the advantage of the user/decision maker being shielded from the complexity of the system. In Figure 2 below we have illustrated the trade-offs involved in usability vs. versatility. Most Web Services applications and Orchestration Environments lie on this curve. In this paper we explore two prototypes that were developed as part of our research:

- Prototype 1: A spreadsheet based Web Services Application that integrates a number of web-services to support a complex decision situation

- Prototype 2: A workflow oriented Web Services Orchestration Environment that dynamically allows us to integrate Web Services to create scenarios that meet changing needs. While shielding naïve users from problem complexity the prototype provides versatility, usability, and customisability

In these prototypes we explore the various dimensions discussed above as well as their ability to satisfy the versatility and usability dimensions (Figure 2).

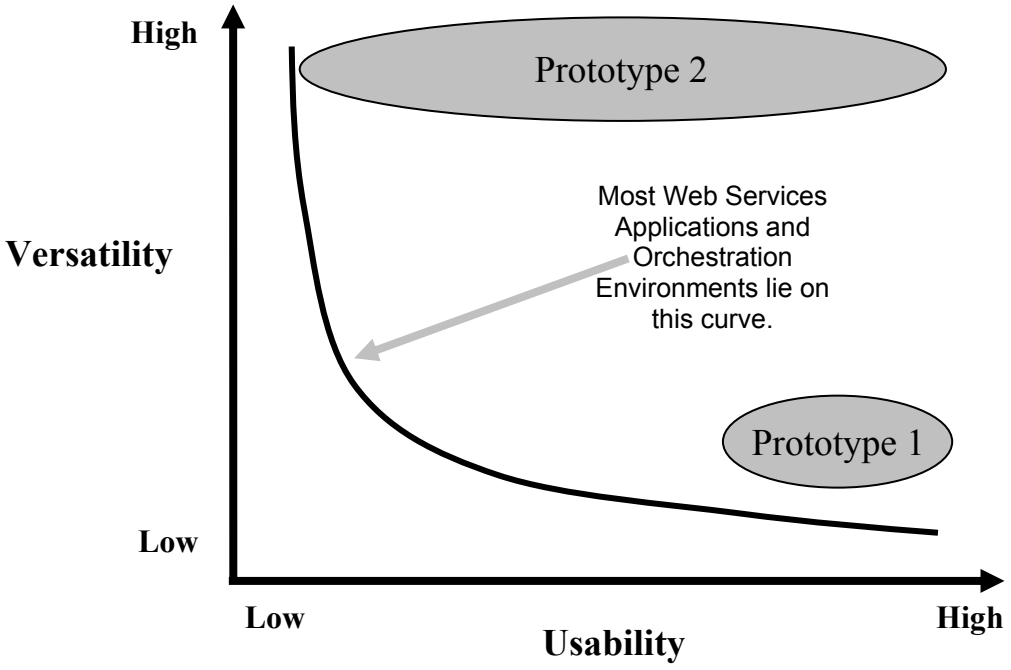


Figure 2: Trade-offs between Usability and Versatility

In many cases, Web Services will not exist at the extremities of any one of the above dimensions. However, the dimensions provide a useful set of ideas for the consideration of how organizations can usefully implement Web Services to fit existing business processes.

Service Chain Orchestration

Recent work by Fremantle, et al. (2002) puts forward the idea of a service oriented vision for enterprise architecture, where components described in WSDL are registered and called upon by other components when they are required. In the service oriented vision, components are choreographed to form a chain of components, performing various enterprise tasks. We call these integrated components service chains.

Actually, Web Services can be wrappers for service chains. A service chain is composed of many linked Web Services, but it can be represented by a single Web Service. A service chain that has its start and end points exposed allows itself to be conveniently wrapped for reuse elsewhere via a Web Service. For this reason, the dimensions of the taxonomy described above apply equally to service chains as well as individual Web Services, because they may in some cases be one and the same thing.

The service oriented vision is driven largely by two themes. The first is e-Business, in which organizations are increasingly exposing and integrating their processes. The second is business process automation, which is the systematic integration of everyday business processes by integrating core systems. The service oriented vision has many benefits including the ability to expose existing functionality through WSDL, moderating heterogeneity, providing interoperability and supporting business processes.

Two key initiatives that support this vision are the Web Services Choreography Interface (Arkin et al., 2002) and the Business Process Execution Language for Web Services (Curbera, et

al., 2002). The Web Services Choreography Interface (WSCI) initiative is spearheaded by BEA Systems, Intalio, SAP, and Sun Microsystems. It is an XML-based language that describes and provides a global picture of the messages exchanged by interacting Web Services. The Business Process Execution Language for Web Services (BPEL4WS) is spearheaded by IBM and Microsoft. It combines IBM's Web Services Flow Language (WSFL) technologies and Microsoft's Xlang and defines a notation for describing the behaviour of business processes that are based on Web Services.

One common application of e-Business is inter-organizational value chain building, where a business process spans across systems of multiple organizations, adding value at each node. While current architectures like CORBA and COM+ perform these tasks well, they are tightly coupled and expensive. Web Services, while less sophisticated than these technologies, are natural candidates to replace them because of the simplicity and modularity they offer.

A construct that outlines how service chains can support e-Business and the service oriented vision is the service chain complexity grid (Figure 3). The grid focuses on the dimensions of complexity and reach of chains of Web Services. We have singled out these dimensions because we believe they are the critical enablers of e-Business. However, many of these dimensions are inter-related in particular ways.

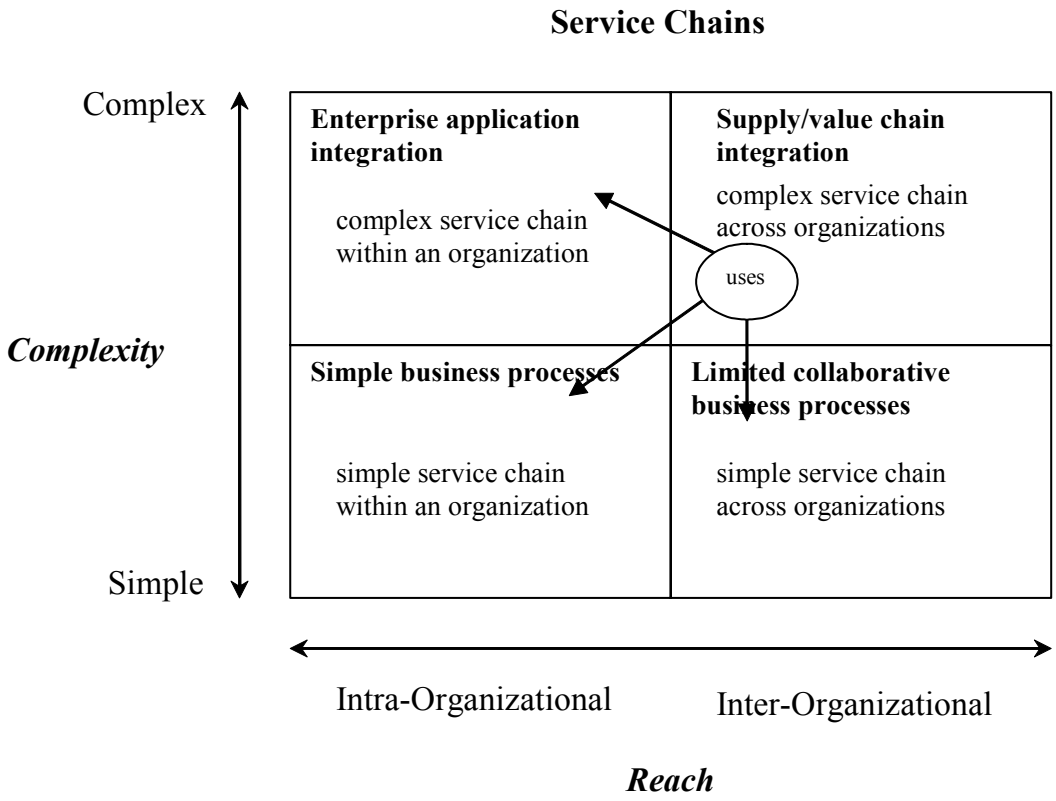


Figure 3: The e-Business service chain grid

For example, if a Web Service is dynamic rather than static, nested or composite rather than flat, or versatile rather than specific, it is also more like to be complex, rather than simple. Essentially, if a Web Service performs a simple task, it is probably a fairly simple Web Service. Conversely, complex tasks are performed by more complex Web Services. The reach of a Web Service is another critical measure of its usefulness in e-Business activities. Reach refers to the Web Service's proclivity to communicate across numerous distinct systems that may belong to a single organization (see the description of the application at GM mentioned earlier) or span organizational boundaries.

All e-Business activities, typically would fall somewhere in the quadrants of this grid. From operational activities and processes at the bottom left of the grid, to more strategic processes at the top right. The first quadrant (bottom left) describes simple business processes. These processes are the building blocks of e-Business, used to perform simple operational tasks. Quadrant two (on the bottom right) of the grid represents business processes which are simplistic but beginning to show a collaborative theme as they span organizational boundaries. E-Business activities are not always conducted across organizational boundaries, however. Often there are intra-organizational processes which leverage e-Business models. Usually these cases involve inter-departmental processes or communication across internal system boundaries. Enterprise application integration is an example of this type of activity, which is found in quadrant three (top left). Quadrant four of the grid is the most sophisticated area of activity, with complex inter-organizational processes being employed to provide strategic collaboration between business partners.

Dynamic collaborations between business processes often require peer to peer interaction. Figure 4 illustrates the peer to peer service model, and is a good example of the collaborations found in the supply/value chain integration quadrant of the grid.

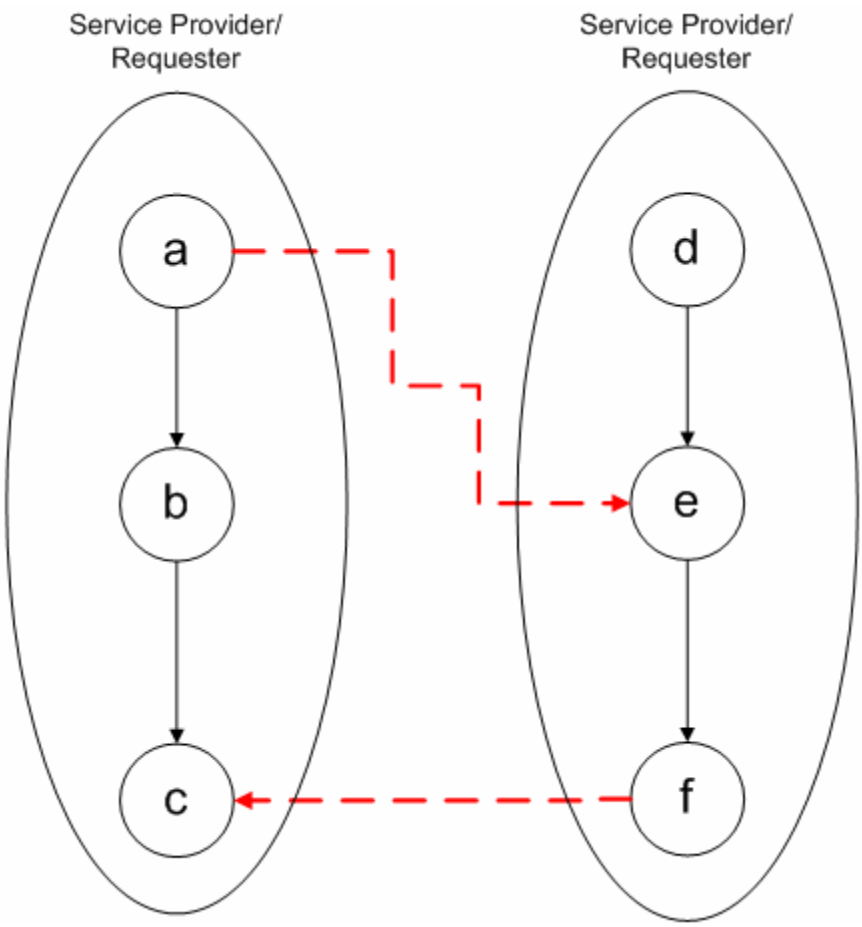


Figure 4: The peer to peer structure of two interacting business processes (Leymann et al., 2002)

Being able to orchestrate diverse Web Services to build service chains would be difficult without a graphical representation of some kind. Allowing users to drag and drop icons onto a screen to build a process dynamically offers many benefits over attempting to code such a process without a visual aid. Increased understanding, better architecture, better manageability, and higher productivity are but a few examples of the benefits that may be derived. Existing workflow architectures do support these features but are generally limited to intra-organizational components. A workflow approach to managing Web Services is one way to overcome this problem. Such an approach will be a critical enabler of the service oriented vision.

Workflow-based Web Services Orchestration

A system that orchestrates diverse Web Services over a distributed environment needs to be able to find them, and bind to them together somehow in a meaningful fashion. Figure 5 illustrates the broad vision of what such a system will achieve.

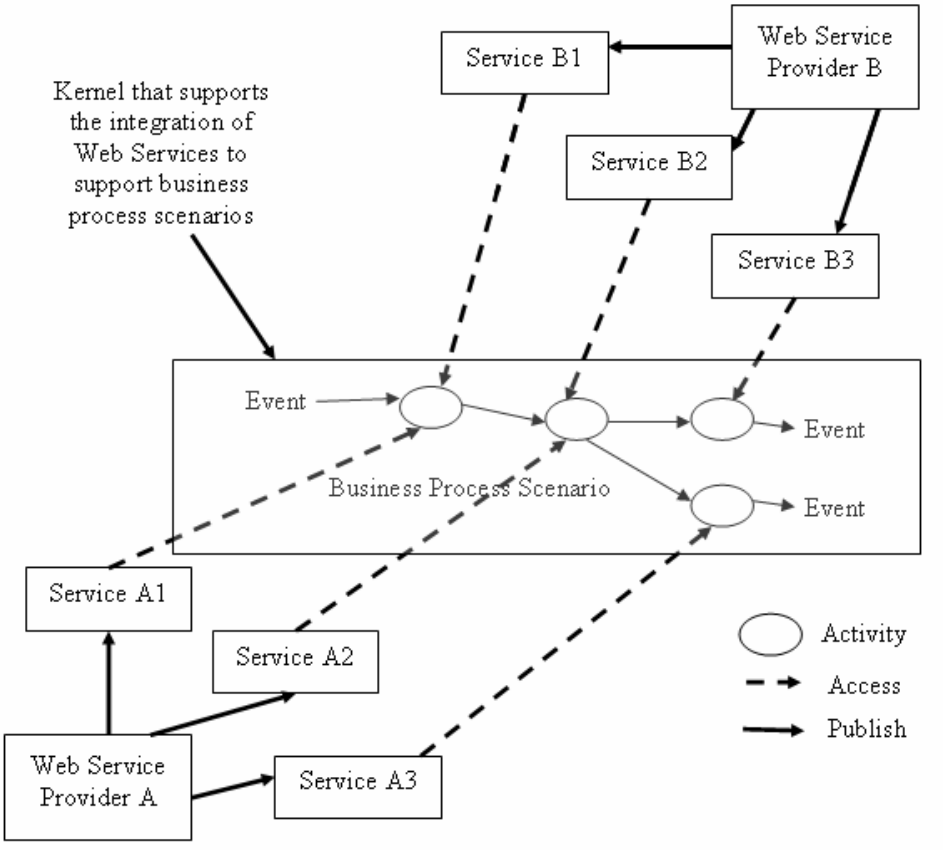


Figure 5: The Service Integration Vision

At the core of the system is a kernel that supports the plugging in of any combination of services to create a service chain which supports the execution of a business process scenario. The *scenario* is intended to allow for the construction of a useful business process on an *ad-hoc* basis. Stable and well recognized business processes would exist in an organization and these have been the focus of much of the existing work. The combination of such stable with *ad-hoc*

processes needs to be explicitly acknowledged in a true service oriented framework. We think of a scenario in this case as an instance of a process. Services may come from within an organization, another organization, or independent service providers. These providers could either be service development specialists or other application vendors or developers exposing useful functionality of existing systems. The aforementioned two prototypes implement this vision in very different ways. Prototype 1 uses a traditional Web Services application oriented approach whereas Prototype 2 uses a Web Services Orchestration and scenario generator approach.

Prototype 1: A Specific Application that integrates multiple Web Services to support a complex decision situation

In the following sections we explore the architecture and implementation of a specific Web Services application that integrates multiple Web Services that reflect some of the dimensions that we explored earlier. This section describes the client side application delivered through a familiar spreadsheet environment. The application provides for a user to get stock quotes and option values based on the Black-Scholes model. It also provides charts of historical data for the stocks. The data can be viewed on an individual or comparative basis of all the stocks in a spreadsheet environment. All this is achieved through the consumption of Web Services of varying degrees of complexity.

Architecture

A simplified view of the Web Services application is illustrated in Figure 6. The application accesses, manipulates, and integrates results from multiple Web Services spread across the internet, intranet, and local databases to support a complex decision process.

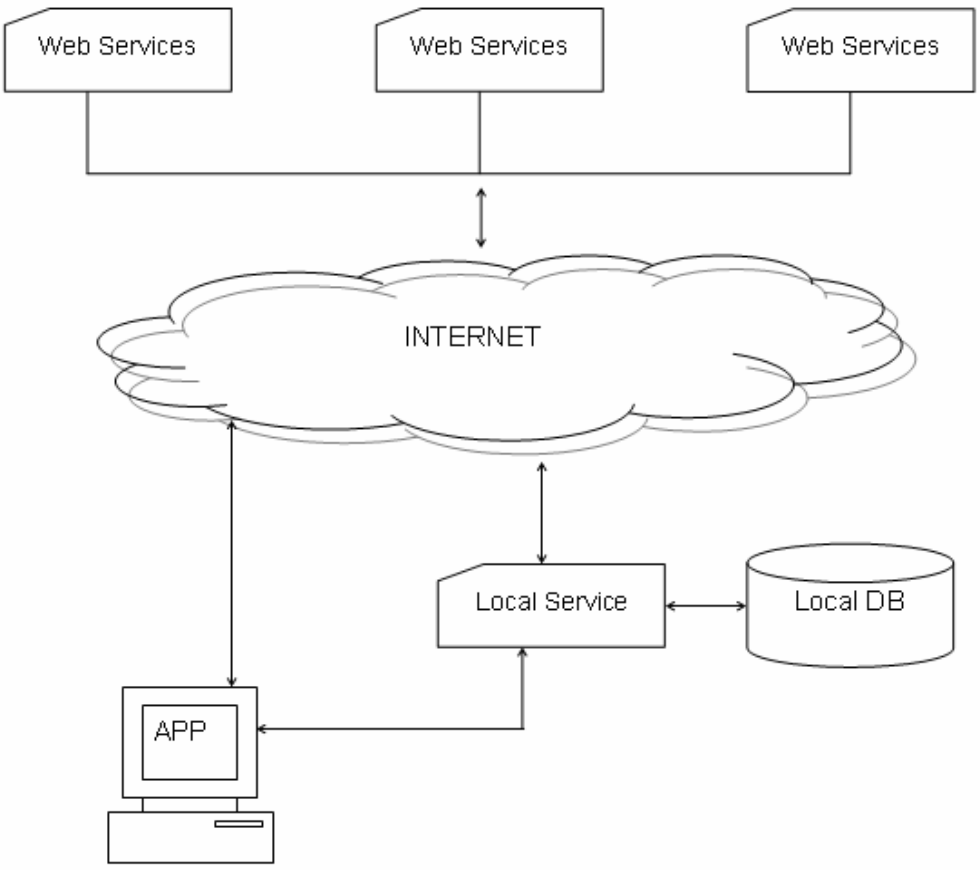


Figure 6: Simplified View of the Web Services Application

The spreadsheet prototype is based on client server architecture. The prototype itself is the client application and the Web Services it consumes act as the server components. The client application can use a number of Web Services to achieve its goal. These Web Services can either be external to the client environment and hosted on the web (internet) or can be internal to the client environment (intranet), or a combination of the two.

Figure 7 describes the architecture in some detail. We refer to the externally accessed Web Services as the Web server layer. The spreadsheet prototype uses 6 different methods provided by 2 different external Web Services (server layer). It also uses 2 different Web Services hosted internally in the client environment.

The first tier shown in the client layer of the architecture is the presentation tier. This tier presents the prototype as a single integrated entity that provides related functionality. The fact that this functionality is derived from different sources is transparent to the user.

The second tier in the client layer is the proxy tier. A proxy is a person (or object) authorized to represent and act for another. The proxy components tier contains localized 'proxy' components. These objects provide a client-side representation of the actual components located at the server. The client application uses these proxies to access both the internal and external Web Services. The proxies themselves are mostly generated by the development environment. The next tier that deals with mapping, interpretation and messaging, along with the generation of the HTTP requests and handling the responses is also dealt with automatically by the development environment. However certain kinds of mapping are still done explicitly by the client application. This sort of mapping deals with linking different Web Services together based on their input and output parameters.

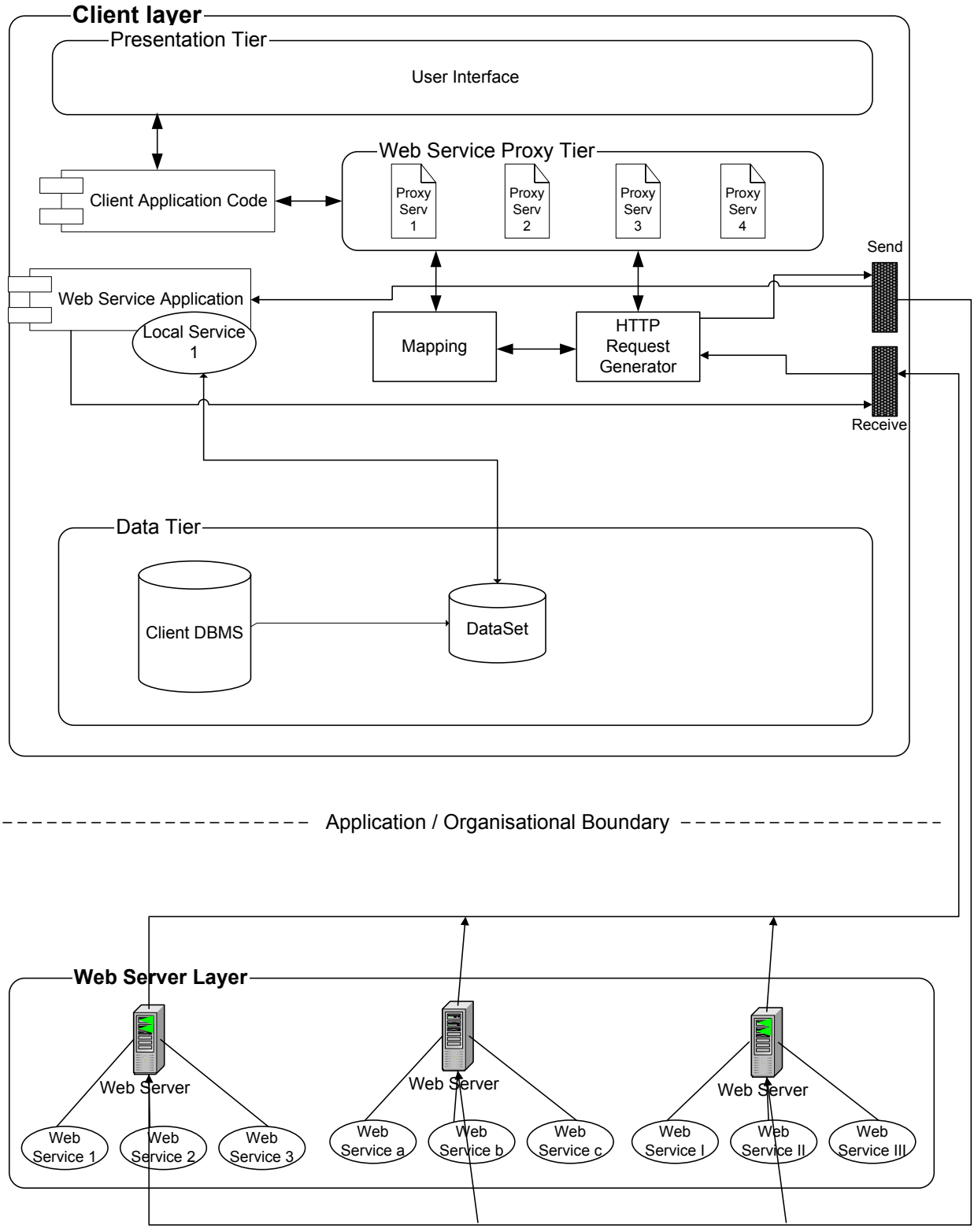


Figure 7: The Web Services Application Architecture

The final tier of the architecture is the data tier, which allows access to locally stored data. The client DBMS is used to store data retrieved from the various external Web Services and also to store data generated locally by the internal Web Services and the application itself. Externally generated data is stored if this historical data needs to be cached locally. If the requested data exists locally then this is used or else it is pulled from the external Web Services. In the case of the spreadsheet prototype, external data in the form of historical stock quote data is stored locally every time the historical quotes Web Service is invoked. This way a check is done to see if the requested data already exists in the database before the Web Service is invoked again. An example of locally generated data stored in the database would be the financial scenario's generated by the prototype. These scenarios are stored as XML spreadsheet objects so that they can be accessed by other external applications through a locally hosted Web Service. These existing scenarios can also be loaded on to the application and further analyzed and modified.

Once all of the necessary relevant services have been integrated at step 1 in Figure 8, the user is able to execute the service chain. When the application is executed, at step 2 in Figure 8, a number of actions occur. First, the application finds all of the relevant services in the service chain, and instantiates all the parameters of the data services. Next, the application retrieves the data from the parameters, and sends it to the Web Service. This is accomplished at step 3 in Figure 8.

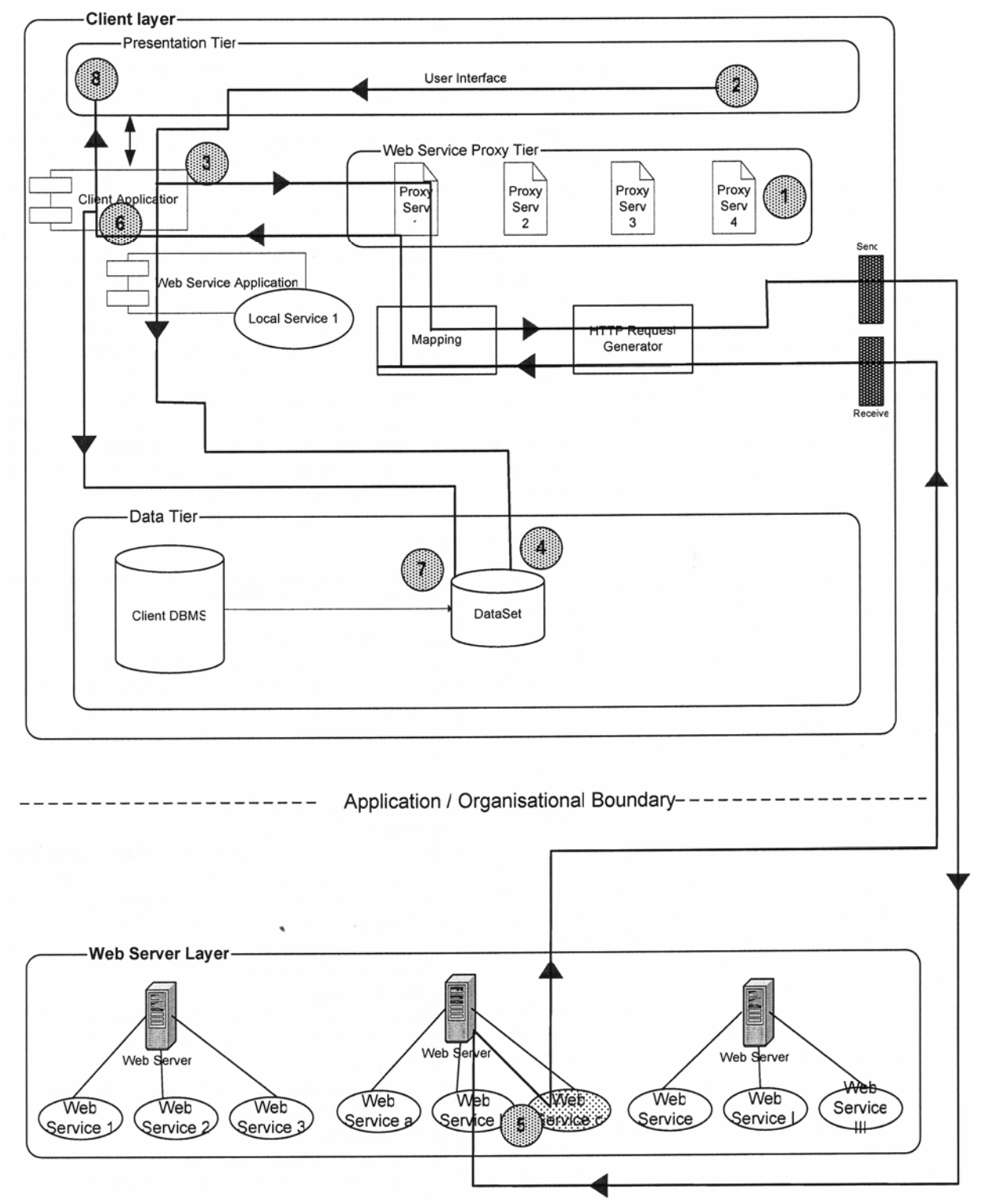


Figure 8: Executing the Service Chain

At this stage a check is done to see if the service is an in-house service or one that is externally accessed on the web server layer. If the service is an in-house one, then it is locally accessed and data is retrieved from the client DBMS itself (step 4, Figure 8). If the service is being provided by an external web service, this web service is accessed at the web server layer. Regardless of the service being external or internal, the application updates the client data service with the data returned by the Web Service. At step 6 Figure 8, the application updates the database that the data service is linked to, with the data returned by the Web Service. The output is also displayed to the user at the presentation layer of the prototype (step 8, Figure 8). This entire process amounts to a series of push and pull operations, where data is pulled from one service and pushed to another. The order of this pushing and pulling defines the business process we are building.

Implementation

The external web services were all sourced from Xignite (www.xignite.com). Xignite is a company that helps financial institutions deliver wealth management services to investors. They provide consulting services to this end and offer two complementary solutions:

- xPortal: A flexible wealth management portal that acts as integration layer deployed on top of the different resources available in the firm.
- xServices: The worlds largest collection of Wealth Management services in the form of web services.

These services are listed in the following UDDI registries:

http://www.xmethods.com/
http://www.salcentral.com/salnet/webserviceswsdl.asp
http://www.remotemethods.com/
http://www.bindingpoint.com/

Details of the Xignite web services that we used and their respective methods are as shown in Figure 9.

Web Service: XigniteOptions

getBlackScholesValue
This method calculates the value of an option using the Black-Scholes model. The method takes in parameters such as the type of the option, stock price, years to maturity, the strike price, the risk free rate and the volatility.

Web Service: XigniteQuotes

getQuote
This method returns a 20 minute delayed stock quote. The return value contains a host of information such as the 52 week high, the 52 week low, Market Capitalization. This method is used to get the stock price and the company name of the stock symbols filled in the spreadsheet.

getTopGainers
Returns quote information about the top gaining equities from NYSE, NASDAQ and AMEX.

getTopLosers
Returns quote information about the top losing equities from NYSE, NASDAQ and AMEX.

getTopMovers
Returns quote information about the top moving equities from NYSE, NASDAQ and AMEX.

getQuotesHistorical
Returns the stock quotes for a given stock for the whole month.

Figure 9: Xignite Web Service and their Methods that were used by the Application

The user interface fundamentally follows the spreadsheet paradigm with some additional functionality provided by buttons on top of the sheet (Figure 10).

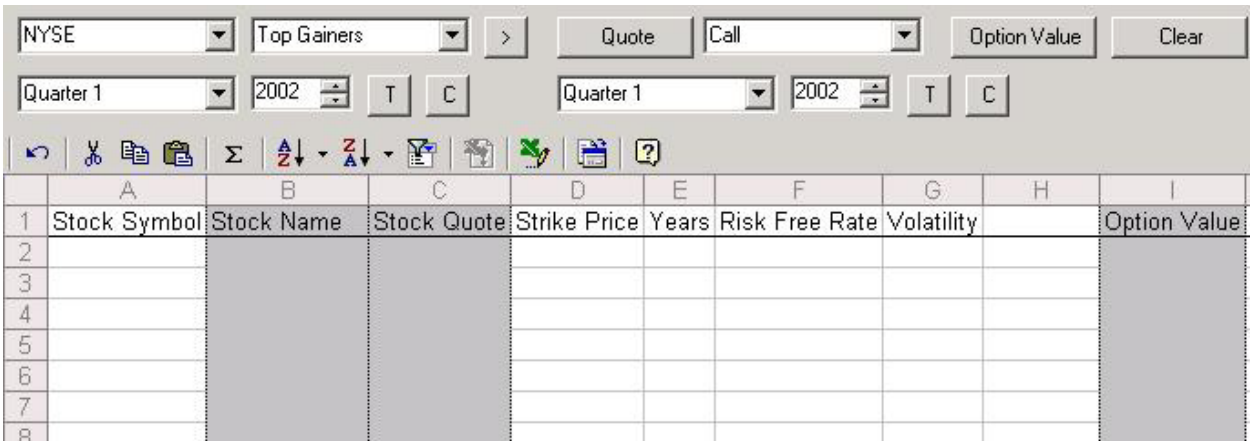


Figure 10: Web Services Application Interface

This is the opening screen. The user can either enter the symbols in the 'Stock Symbol' column or use a section of 'Top Gainers', 'Top Losers' or 'Top movers' from the NYSE, NASDAQ or AMEX stock exchanges. Using this option would give the user a pop up window (Figure 11) from where selection can be made.

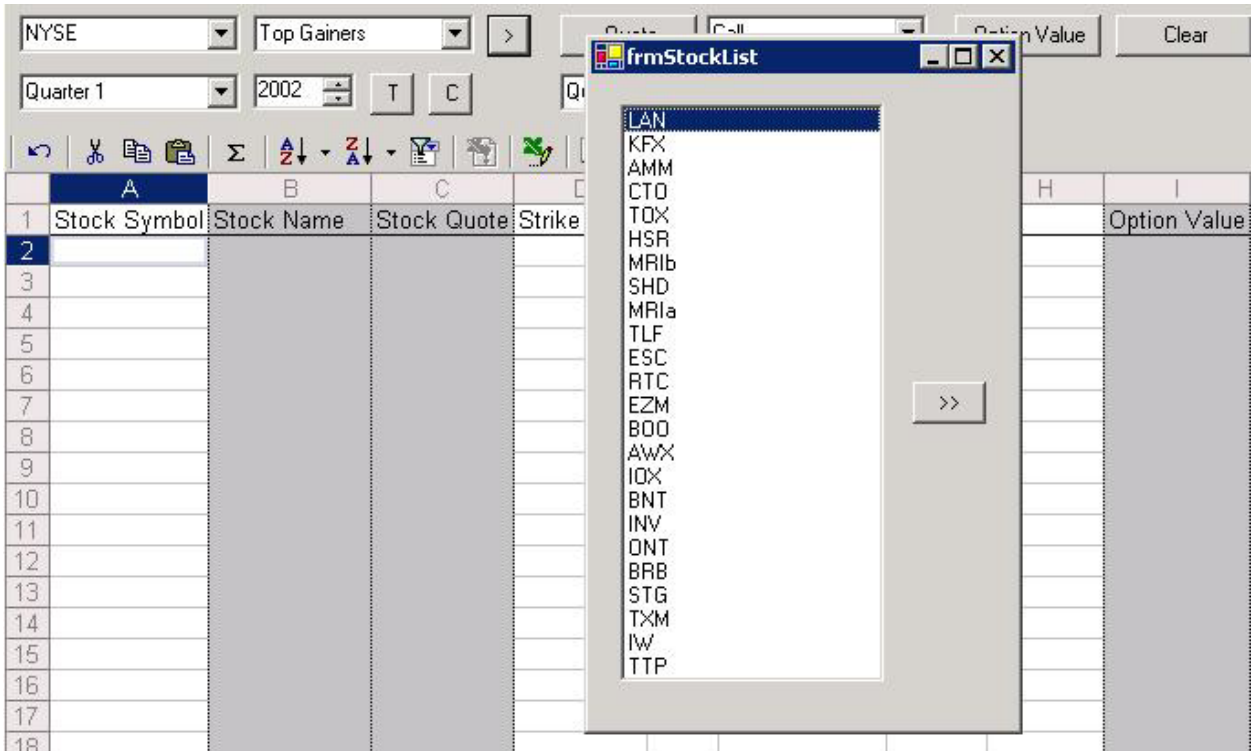


Figure 11: Stock Listing of Top Gainers, Losers, or Movers from NYSE, NASDAQ or AMEX

The user can select the stock symbols from this list which will be added to the existing list in the 'Stock Symbol' column (Figure 12).

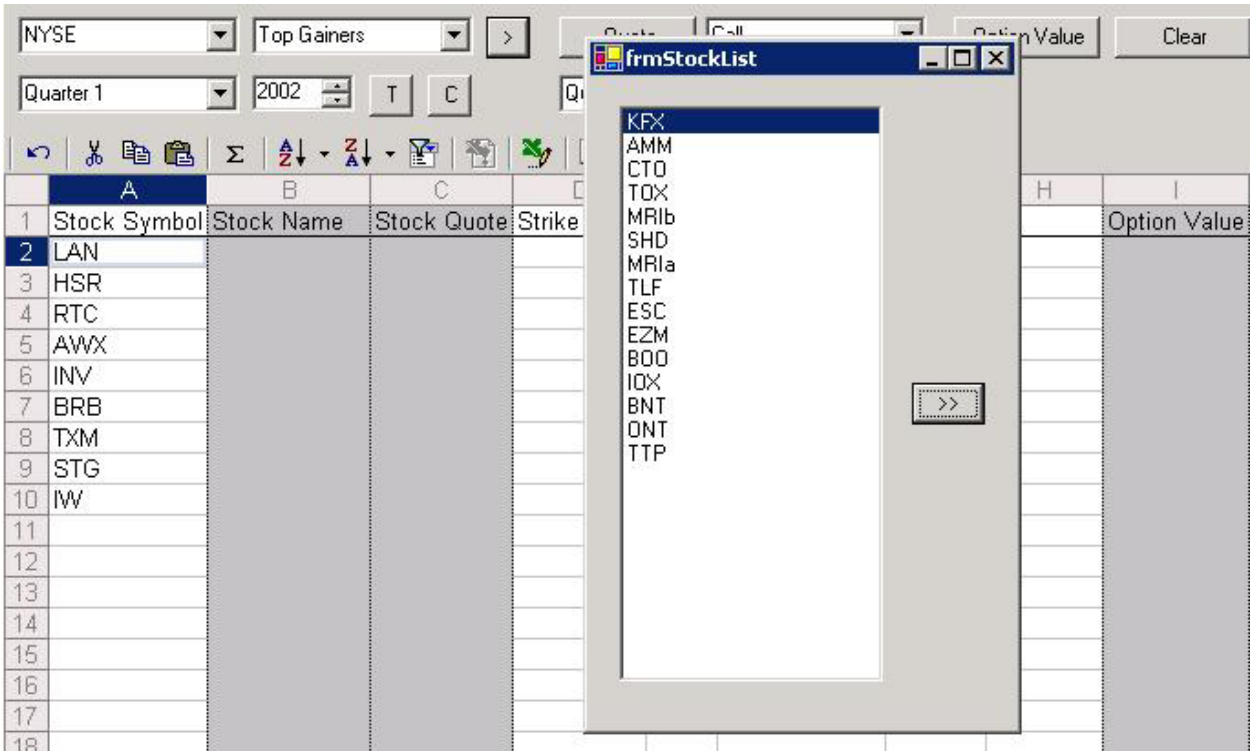
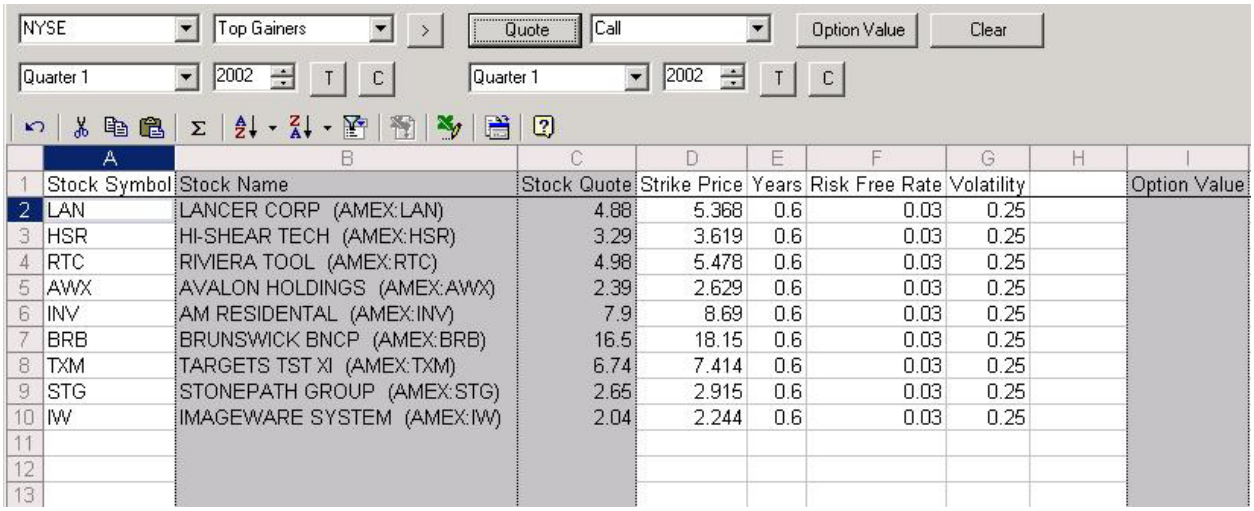


Figure 12: Stocks for Consideration

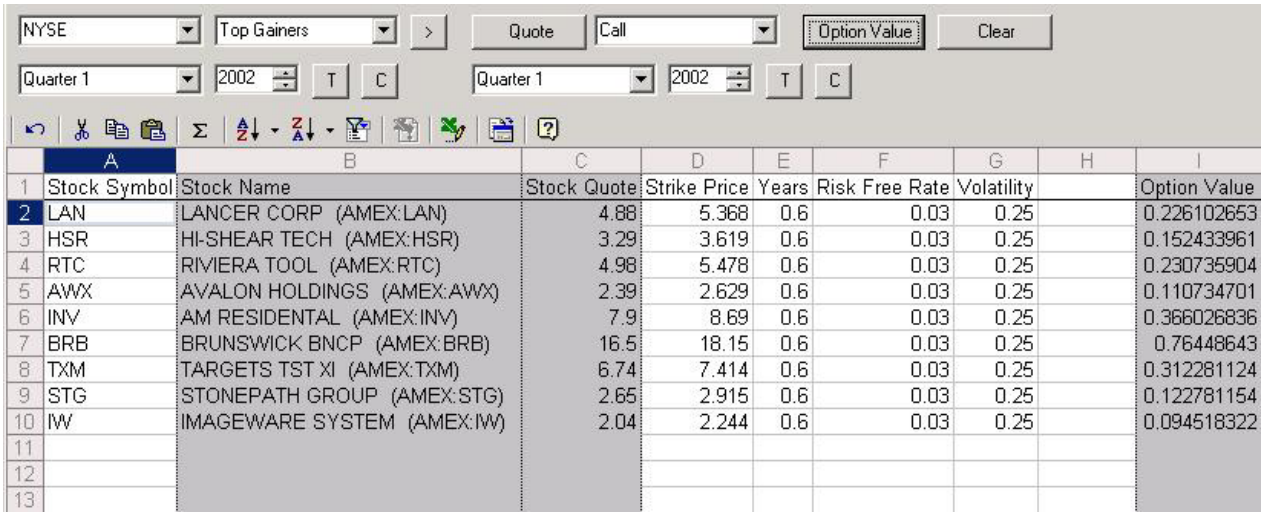
On clicking the quote button Web Services are called to fill in the stock name and the stock quote. A local service will populate the other columns – Strike Price, Years, Risk Free Rate and the Volatility. (Figure 13) These additional columns are populated with default values in calculating the stock option prices based on the stock quotes received. The user can then change these values to calculate the Option price.



	A	B	C	D	E	F	G	H	I
1	Stock Symbol	Stock Name	Stock Quote	Strike Price	Years	Risk Free Rate	Volatility		Option Value
2	LAN	LANCER CORP (AMEX:LAN)	4.88	5.368	0.6	0.03	0.25		
3	HSR	HI-SHEAR TECH (AMEX:HSR)	3.29	3.619	0.6	0.03	0.25		
4	RTC	RIVIERA TOOL (AMEX:RTC)	4.98	5.478	0.6	0.03	0.25		
5	AWX	AVALON HOLDINGS (AMEX:AWX)	2.39	2.629	0.6	0.03	0.25		
6	INV	AM RESIDENTAL (AMEX:INV)	7.9	8.69	0.6	0.03	0.25		
7	BRB	BRUNSWICK BNCP (AMEX:BRB)	16.5	18.15	0.6	0.03	0.25		
8	TXM	TARGETS TST XI (AMEX:TXM)	6.74	7.414	0.6	0.03	0.25		
9	STG	STONEPATH GROUP (AMEX:STG)	2.65	2.915	0.6	0.03	0.25		
10	IW	IMAGEWARE SYSTEM (AMEX:IW)	2.04	2.244	0.6	0.03	0.25		
11									
12									
13									

Figure 13: Local Web Service populates Strike Price, Years, Risk Free Rate and Volatility

The option value is fetched from another Web Service with all the values for a particular stock taken in from the excel sheet. This is done so by clicking the *Options* button and by selecting either a 'put' or a 'call' value based on the type of option required to be calculated. This results in the *option values* being populated (Figure 14).



	A	B	C	D	E	F	G	H	I
1	Stock Symbol	Stock Name	Stock Quote	Strike Price	Years	Risk Free Rate	Volatility		Option Value
2	LAN	LANCER CORP (AMEX:LAN)	4.88	5.368	0.6	0.03	0.25		0.226102653
3	HSR	HI-SHEAR TECH (AMEX:HSR)	3.29	3.619	0.6	0.03	0.25		0.152433961
4	RTC	RIVIERA TOOL (AMEX:RTC)	4.98	5.478	0.6	0.03	0.25		0.230735904
5	AWX	AVALON HOLDINGS (AMEX:AWX)	2.39	2.629	0.6	0.03	0.25		0.110734701
6	INV	AM RESIDENTAL (AMEX:INV)	7.9	8.69	0.6	0.03	0.25		0.366026836
7	BRB	BRUNSWICK BNCP (AMEX:BRB)	16.5	18.15	0.6	0.03	0.25		0.76448643
8	TXM	TARGETS TST XI (AMEX:TXM)	6.74	7.414	0.6	0.03	0.25		0.312281124
9	STG	STONEPATH GROUP (AMEX:STG)	2.65	2.915	0.6	0.03	0.25		0.122781154
10	IW	IMAGEWARE SYSTEM (AMEX:IW)	2.04	2.244	0.6	0.03	0.25		0.094518322
11									
12									
13									

Figure 14: Execution of the Option Price Web Service

The user can view a chart of historical data based on the type of view s/he would like to see. S/he could either see a chart of a single stock (Figure 15) or for all the stocks in the spreadsheet (Figure 16).

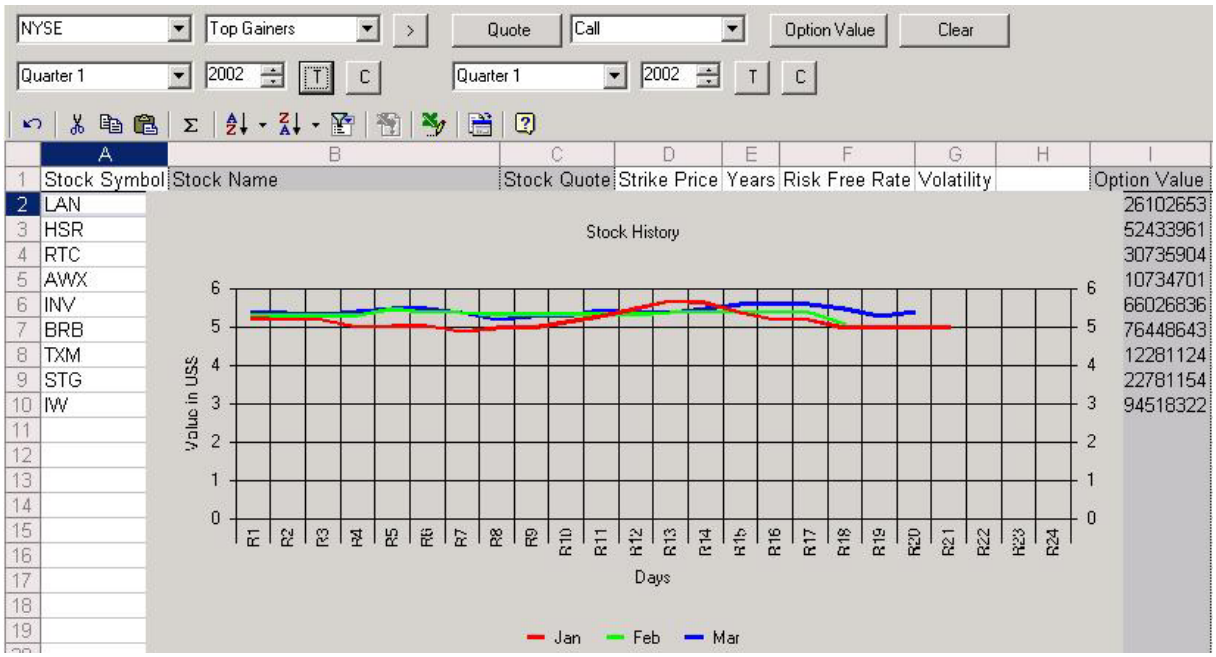


Figure 15: Graph of historical data for the selected stock

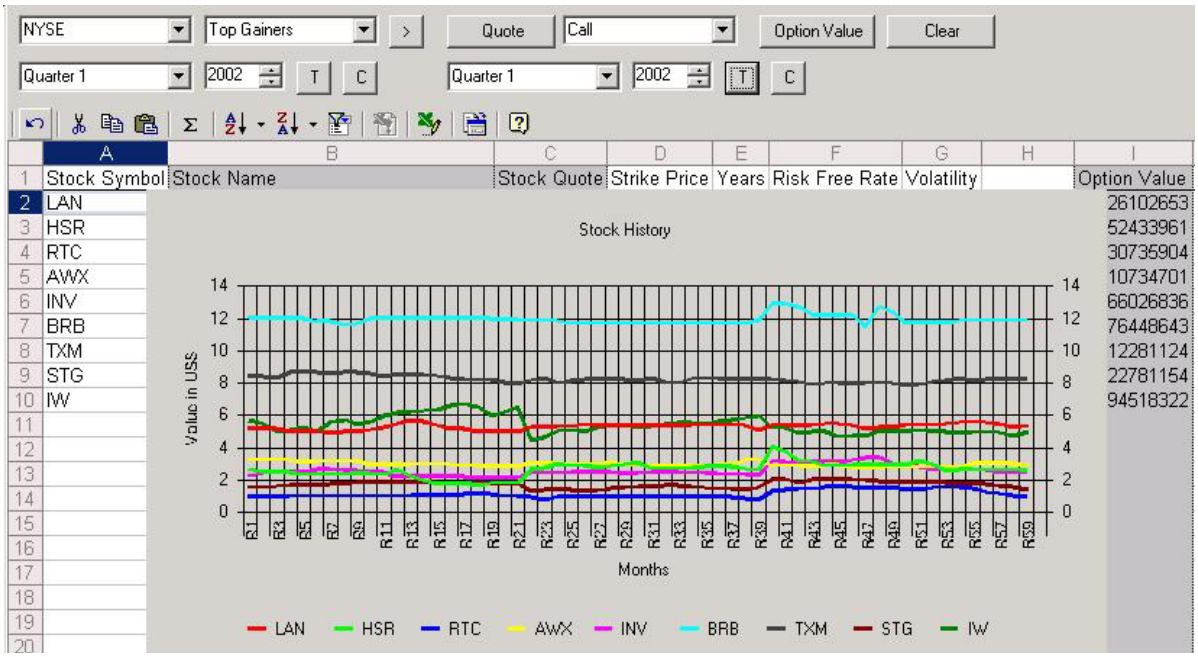


Figure 16: Graph of historical data for all the stock symbols in the sheet

The Web Services that are required to fetch historical data for the selected period for the individual symbols are called individually. This could be time consuming if there were a lot of symbols on the list. For this the application makes used of historical data stored in the local cache. If the requested data exists locally then this is used; else it is pulled from the Web Service.

Prototype 2: Web Services Orchestration and Scenario Generation

In the following sections we explore the framework, architecture, and implementation of a Web Services orchestration and scenario generation environment that overcomes some of the problems exhibited by traditional Web Services applications. The concepts and implementation of this particular prototype were initiated by the work of Pienaar (2002).

Framework and Architecture

We have built a framework that will support the activities that we consider to be important. Figure 17 illustrates the workflow-based Web Services orchestration framework. The framework is simplistic in design. We have taken a rather generalized approach because we see value in reusing it across multiple domains, rather than being too specific and limiting its use.

The framework follows the client/server paradigm. The server provides the Web Service(s) we intend to use. Six high level layers of the client have been proposed. The first layer of the framework is the user interface and manipulation layer. This consists of an iconic graphical user interface (GUI) that enables users to access, map, integrates, manipulate, and visualize parts of the system.

The second layer is the proxy components layer. Similar to the first prototype the proxy components layer contains localized 'proxy' components. Providing a proxy service allows the user to access, view, integrate and manipulate the service transparently, without having to communicate with the server where the actual service resides. This reduces the network traffic, time lag and complexity involved in basic service operations.

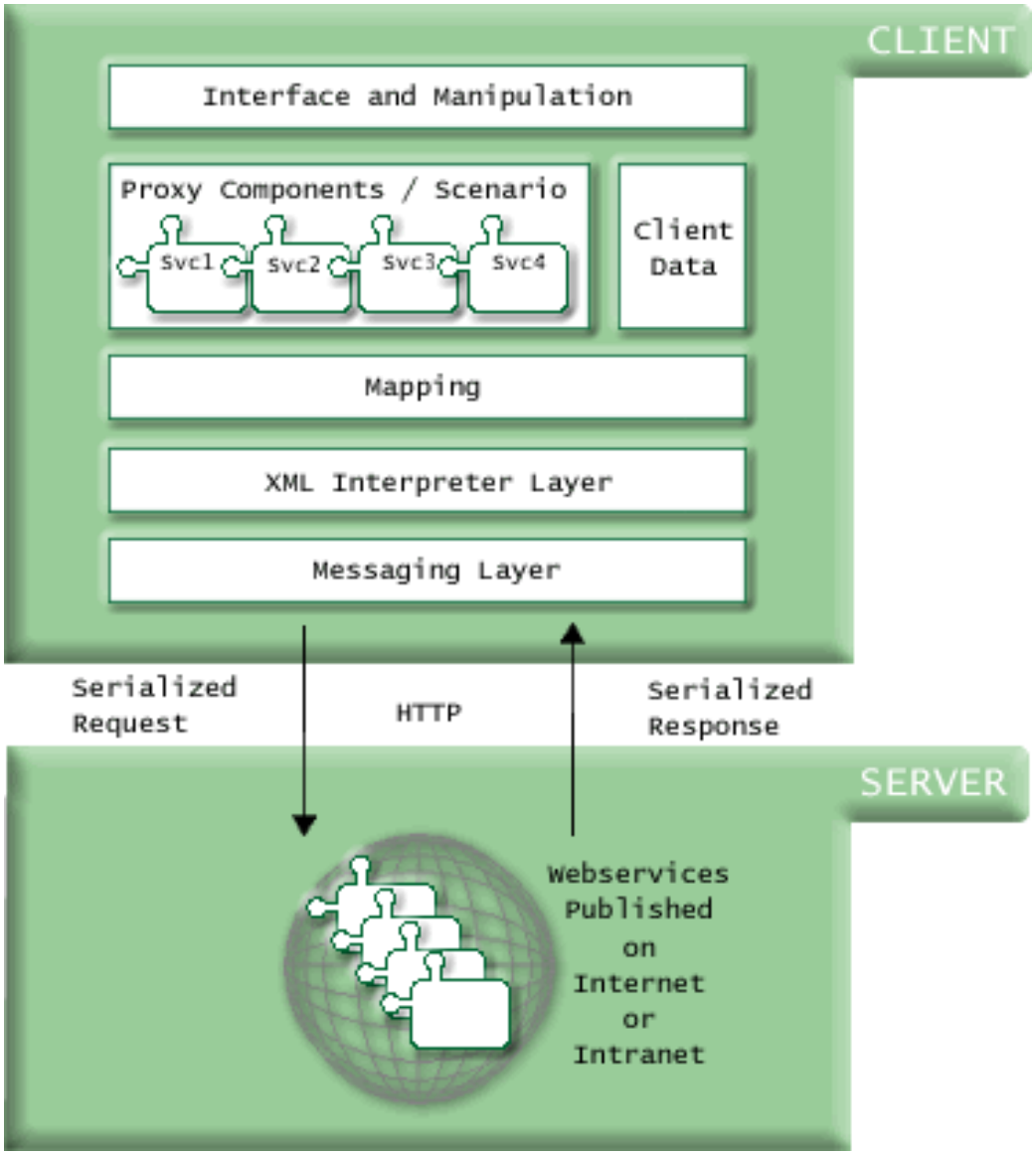


Figure 17: The workflow-based Web Services orchestration framework

The third layer is the client data layer. This layer allows the user to access data stored locally. This may be to retrieve data that could be used to populate a service, or update the data in a database with results obtained by executing a service chain. The client data layer is important in executing many distributed service chains as it is often the beginning of the data flow of the service chain.

The fourth layer of the framework is the mapping layer. It allows the user to link together the services used in the application. It also allows users plug (embed) a specific service into another service. This is implemented by allowing the user to make and maintain mappings (relationships) between parameters of corresponding services.

The fifth layer is the XML interpreter layer. Since the client/server model is a distributed model, a standardized means of communicating between the client and server is required. It must be able to handle services running on different platforms, operating systems, and using different communication software. This layer prepares, encodes (serialization), and decodes (deserialization) XML messages. In our framework, the XML interpreter layer deals mainly with reading and writing SOAP and WSDL.

The last layer of the framework is the messaging layer. The message from the XML interpreter layer is passed on to the messaging layer. The messaging layer then handles the asynchronous sending and receiving of the message and its reply. The messaging layer will send a serialized message via a protocol like HTTP, and then wait for a response. It then passes the response back to the XML interpreter layer for further processing. An example of an architecture

we have created that exploits this framework can be seen in Figure 18. Here the layers of the framework can be seen clearly in the modules of architecture.

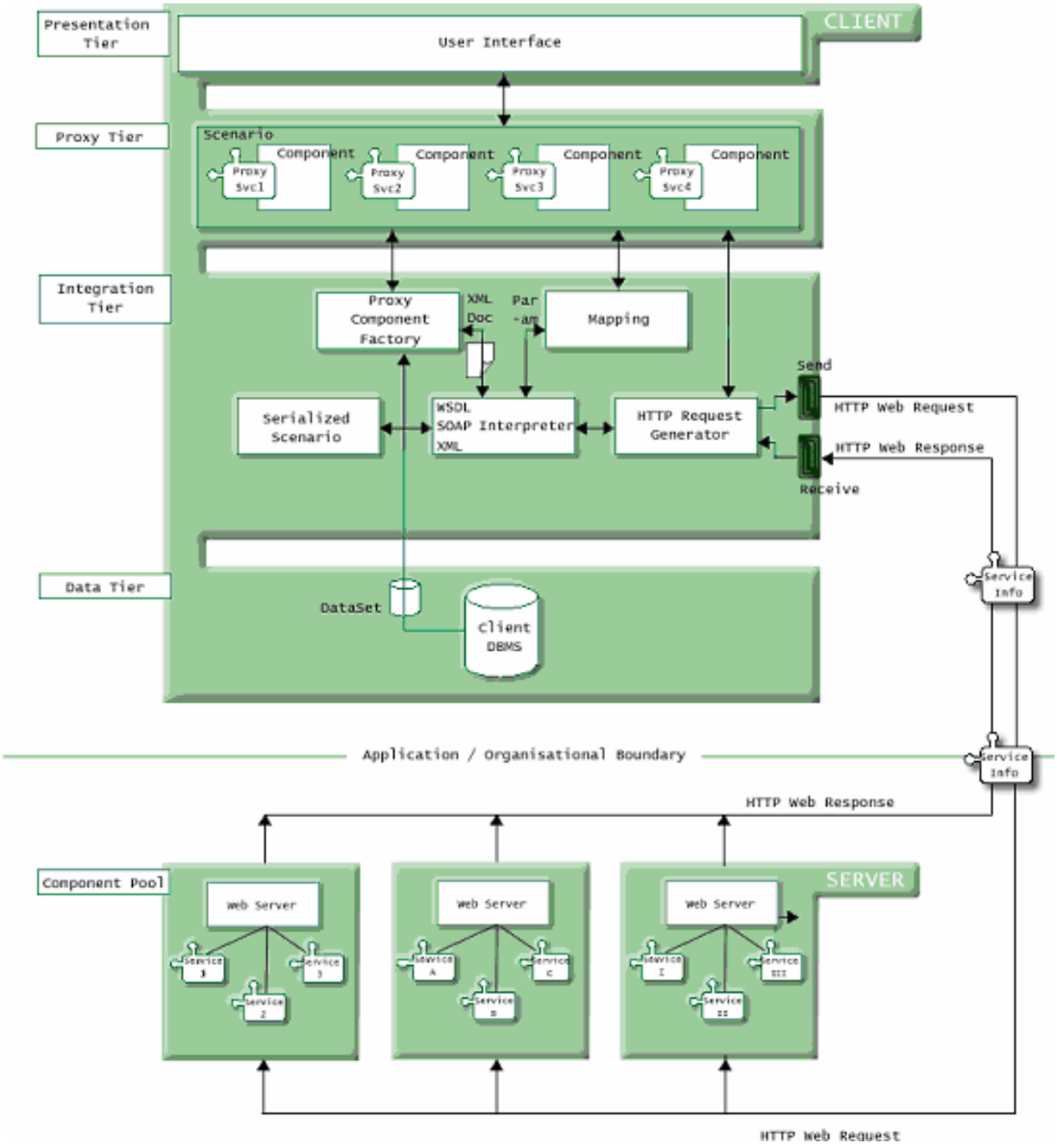


Figure 18: An Architecture for the Web Services orchestration framework

The data tier contains modules for managing the application data, and corresponds to the third (client data) layer of the framework. The integration tier corresponds to the fourth, fifth and sixth layers of the framework (mapping, interpretation and messaging). Most of the functionality in this area deals with creating proxy services through a 'factory' (an OO software development pattern). There is also a strong focus on reading and writing XML through the XML interpreter modules, as well as managing asynchronous messaging in the HTTP request generator module. The proxy tier corresponds to the second layer in the framework (the proxy components layer). This is where most of the functionality to manage proxy services is kept. Finally, the presentation tier relates to the first layer of the framework, the presentation layer. The functionality in this layer provides the workflow tools for the application.

Implementation

A prototype that proves the concepts proposed in the Web Services orchestration framework and architecture was implemented. It was tested against scenarios from a number of different domains with a variety of Web Services. Key phases of Web Service orchestration supported by the prototype are discussed in the following paragraphs.

Generate & manipulate (proxy) components. The first step to using the system is to add services to the canvas (Figure 19). To add a service, the user selects the appropriate service icon in the toolbox, located on the left of the screen, and drops it into the canvas work area. They are (at this point) blank representations of real-world services that will be 'filled in', or instantiated, later. Once the services have been added to the canvas, they may be positioned about the Canvas as required, and operated on using the Editor Panel. Using iconic proxy representations of services

in this fashion shields users from the complexity, location, and implementation of the real Web Services that they represent.

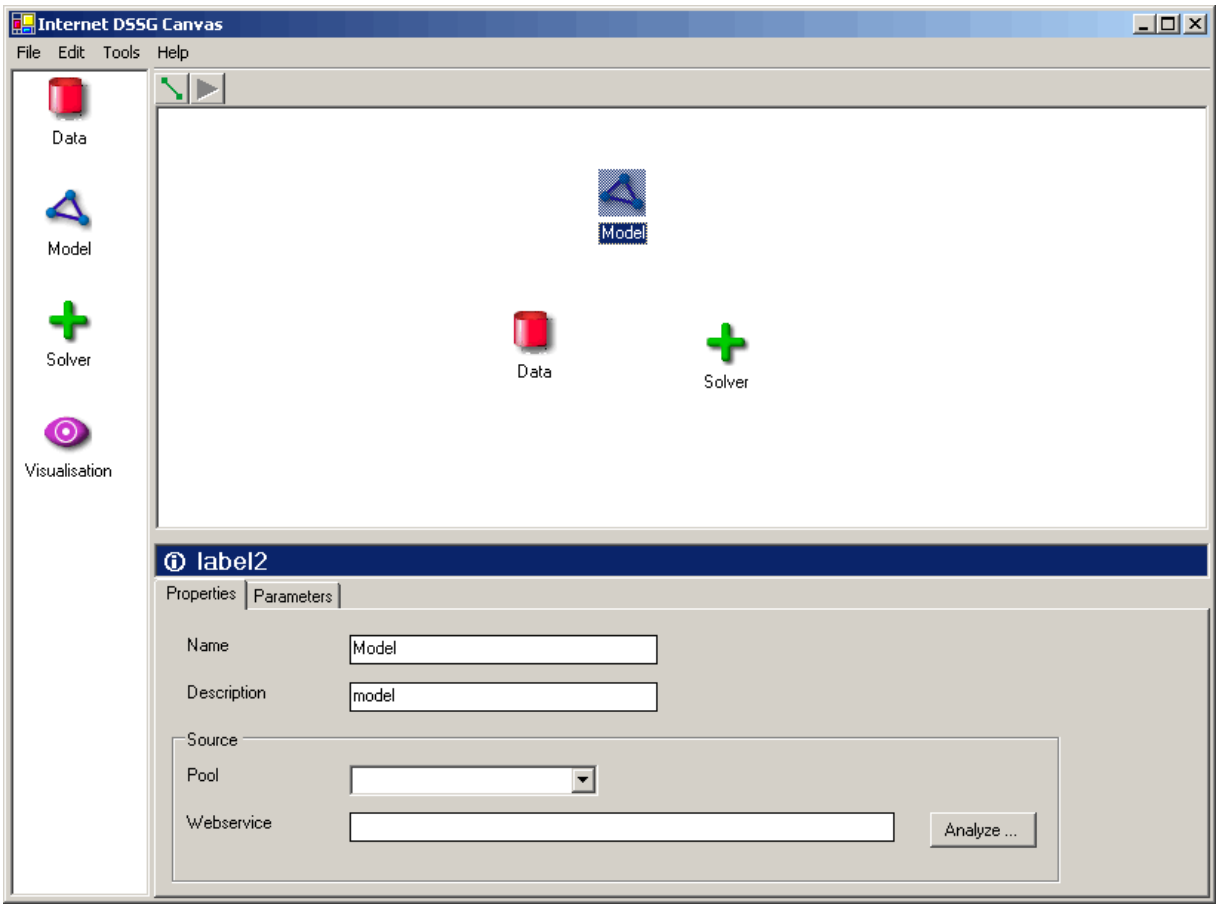


Figure 19: Adding Services to the Canvas

Access heterogeneous and distributed Web Services. The second step is to find and access the real Web Services our icons will represent. Accessing a Web Service is achieved by querying its WSDL interface via HTTP. The user provides a URL for the WSDL file and the application uses that URL to read its description.

Analyse heterogeneous and distributed Web Services to determine their nature and functionality. The third step is to analyse the Web Service via its description to find out what methods it offers. The WSDL is analysed to determine what implementation of SOAP the Web Service uses. Once the implementation type has been discovered, the application reads the WSDL to extract the information about the methods of the Web Service, and what their parameters are (Figure 20).

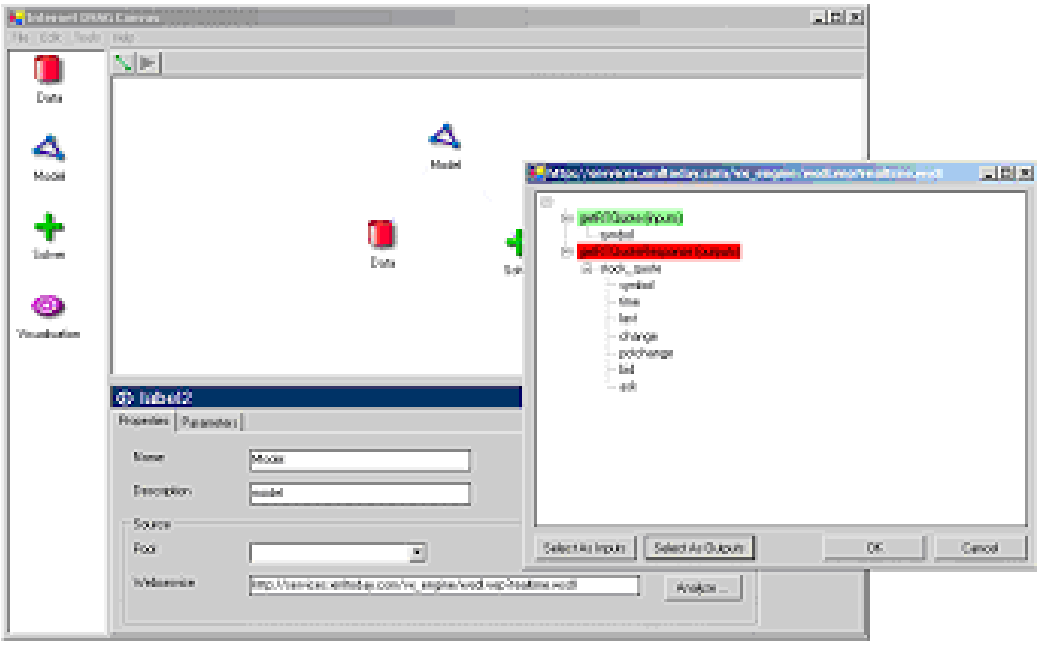


Figure 20: Analysing a Web Service

Select the desired functionality of Web Services based on analysis. The fourth step is to choose which methods of the Web Service to use. The application generates a form that shows the user the methods that are implemented by the Web Service and also the parameters of the methods. Generally, for each Web Service, an input method and its equivalent output method will need to be selected. Selecting the desired methods of the Web Service populates the proxy service with the parameters of those methods.

Map heterogeneous and distributed Web Services to form a service chain. Once we have populated the services, the fifth step is to bind them together through a process called mapping. Mapping services together is the way in which we orchestrate them and design their workflow. To map two populated services together, the user selects them on the Canvas, and clicks on the mapping button on the toolbar. The lines between the services' icons in Figure 21 depict their mappings. Each of the three services in this example is provided by a different organisation. The service chain is therefore inter-organisational.

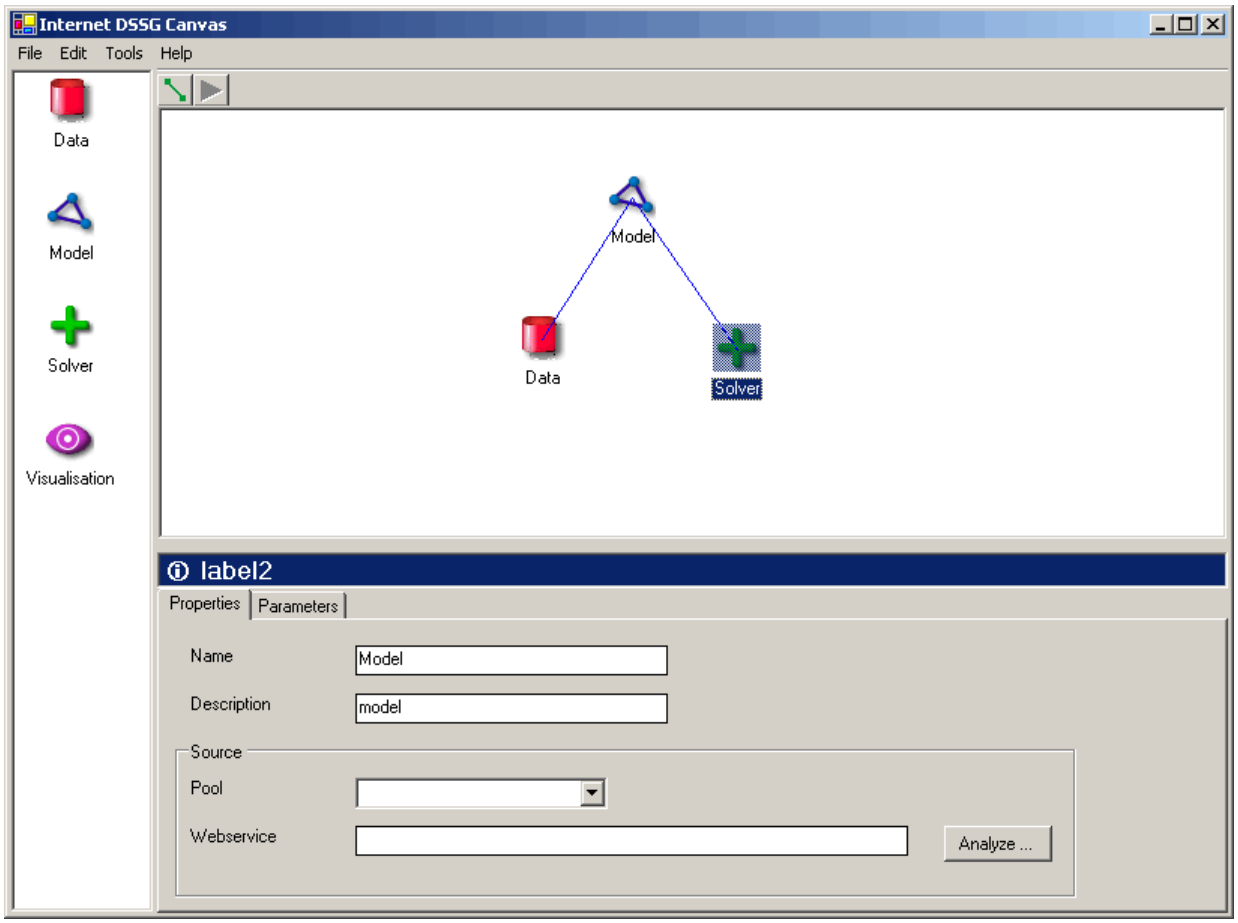


Figure 21: Mapping the Services Together

Map parameters of heterogeneous and distributed Web Services by specifying which parameter of the Web Services should exchange data when a service chain is run. The sixth step involves detailed mappings between the parameters of the services that represent the same data. In the *Parameters* tab of the model service (Figure 22), we can see both of the services to which the model service is mapped: the data service and the solver service. A mapping between two parameters is created by clicking on a parameter of one web service, and dragging and dropping the it onto a parameter of another web service.

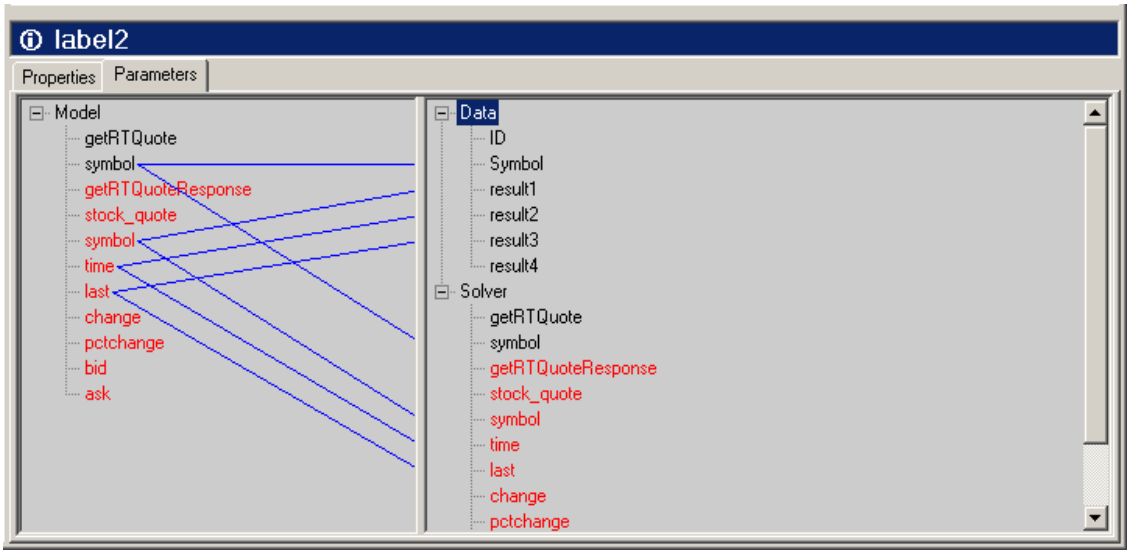


Figure 22: Mapping the Parameters

Execute the distributed service chain by 'pulling' data from distributed Web Services, and 'pushing' data to Web Services. The seventh step is to run the finished service chain. When all of the necessary services have been mapped together, populated, and their parameters mapped together, the user is able to execute the service chain. Figure 23 shows the service chain during execution. A record of the XML messages being sent between the application and the Web Service server is written to a SOAP Log window.

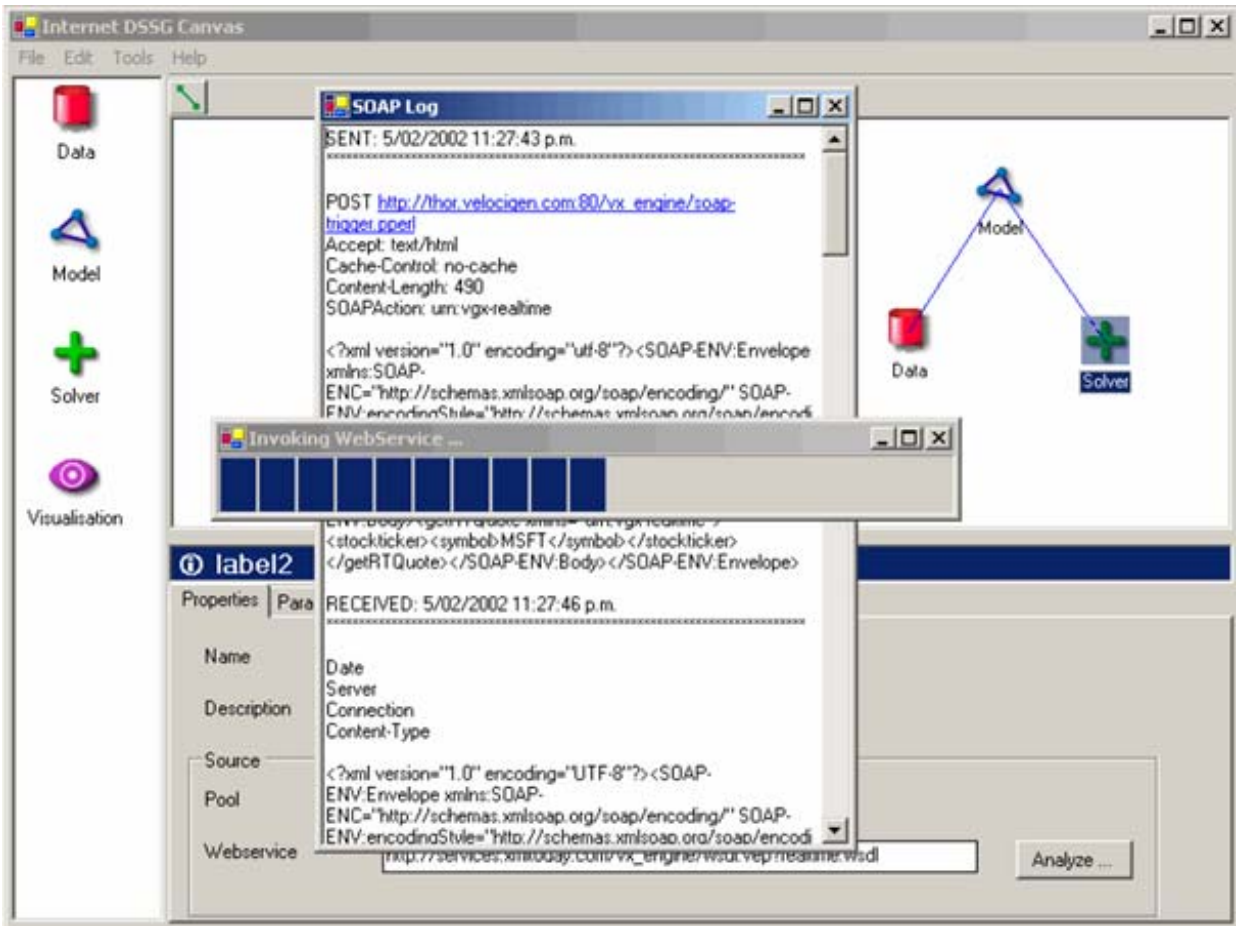


Figure 23: Executing a Service chain

The application executes the solver by creating a SOAP request. This request contains an envelope, a header, and a body. The body in turn contains all the parameters of the Web Service and their current value. This message is then sent to the Web Service. The Web Service interprets this message, executes appropriate methods, and then sends a SOAP response back to the application. Figure 24 is an example of such an interaction while executing the stock value service chain.

```
SENT: 5/02/2002 11:33:22 p.m.
**********************************************************************

POST http://thor.velocigen.com:80/vx_engine/soap-trigger.pperl
Accept: text/html
Cache-Control: no-cache
Content-Length: 490
SOAPAction: urn:vgx-realtime

<?xml version="1.0" encoding="utf-8"?><SOAP-ENV:Envelope xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema"><SOAP-
ENV:Body><getRTQuote xmlns="urn:vgx-
realtime"><stockticker><symbol>MSFT</symbol></stockticker></getRTQuot
e></SOAP-ENV:Body></SOAP-ENV:Envelope>

RECEIVED: 5/02/2002 11:33:23 p.m.
**********************************************************************

Date
Server
Connection
Content-Type

<?xml version="1.0" encoding="UTF-8"?><SOAP-ENV:Envelope xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema"><SOAP-
ENV:Body><namesp166:getRTQuoteResponse
xmlns:namesp166="urn:getRTQuote"><stock_quotes><stock_quote><symbol>M
SFT</symbol><time>Feb  4</time><last>61.12</last><change>-
1.53999996</change><pctchange>-
2.46%</pctchange><bid>61.20</bid><ask>61.22</ask></stock_quote></stoc
k_quotes></namesp166:getRTQuoteResponse></SOAP-ENV:Body></SOAP-
ENV:Envelope>
```

Figure 24: Log of SOAP Requests and Responses

Pressing the run button, on the user interface, results in a sequence of actions. We illustrate the flow of data through the architecture of the application when this occurs through

Figure 25. We instantiate the data sources in the service chain, get the data, execute the solver, set the data, and synchronize the data with the database.

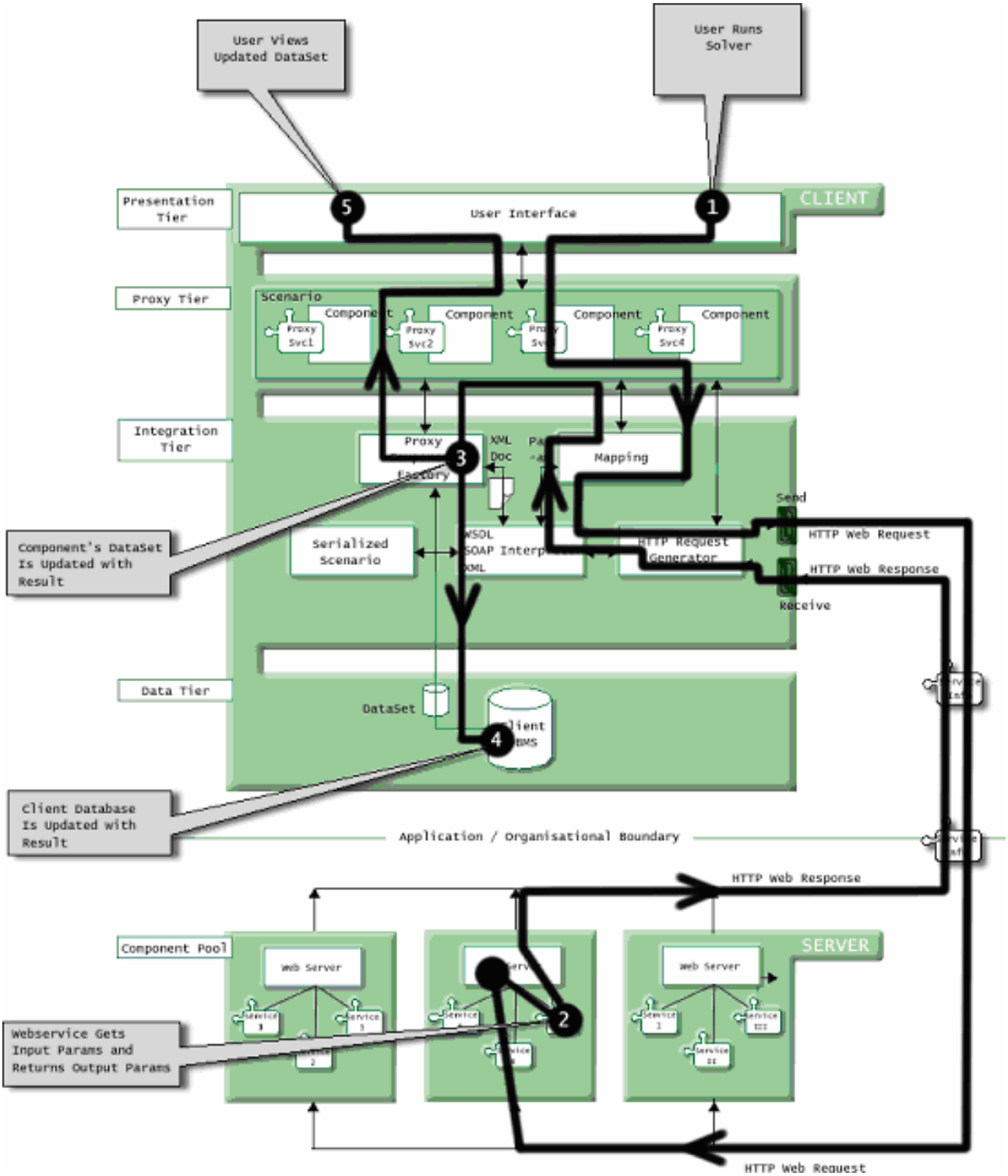


Figure 25: Executing the Service Chain

First, the application finds all of the data services in the service chain, and instantiates all of the mapped parameters of those data services. This prepares them as candidates to provide data when the solver service is looking for data to use. Next, the application retrieves the data from the instantiated parameters, and runs the solver, sending the data from the parameters to the Web Service. This is accomplished at step 2 in Figure 25.

After the solver has been executed, the application updates the client data service with the data returned by the Web Service (step 3 in Figure 25). Finally, at step 4, the application updates the database that the data service is linked to with the data returned by the Web Services.

This entire process amounts to a series of push and pull operations, where data is pulled from one service and pushed to another. The order of this pushing and pulling defines the business process we are building.

Analyse the results of the execution. The eighth step in using the system is getting and analysing the results of executing the service chain. The application allows the user to preview the updated data and thus analyse the results of the service they have executed in the Editor Panel on the main window. Advanced support for flexible analysis of data could easily be achieved through plugging in visualisation Web Services.

Store the distributed service chain including its structure and details of distributed components, for future use. The ninth and final step is to make the service chain persistent. The

system allows the user to save service chains as an XML file, and then load them again whenever they are needed, thereby persisting them. The XML file stores the information required to regenerate the distributed services within a service chain. Once it is saved, the service chain can be loaded again. The ability to retrieve stored service chains allows users to share and incrementally develop them. Thus allowing new functionality to be added or changes to existing functionality. The serialised XML service chain description acts as a language for describing orchestrated Web Services.

Implications of the Framework

The framework we have built supports all of the types of Web Services we defined in the taxonomy. It is general enough to be applied across many domains. This is because we address the fundamental business problem of integrating heterogeneous and loosely coupled components to form a service chain. These building blocks can be the foundation of a huge variety of systems, much as chains of simple proteins in DNA create a huge and complex variety of life.

In order for organizations to evolve, they will need to be able to reap the benefits of the new service model. Quickly and dynamically creating peer to peer interactions will allow them to build inter-organizational value chains which will save time and money. Using workflow principles will give users the tools to harness services in this way.

Our framework provides users with these tools, and provides a set of functionality they can use to quickly and easily knit together diverse services. In so doing, support is provided for all the four quadrants of the e-Business service chain grid we developed. The framework

supports simple and complex service chains, as well as intra and inter-organizational service chains. It does this in a flexible and modular way. The framework therefore supports not only simple operational business processes, but complex strategic service chains as well. This has implications for enterprise application integration, business process management, value chain management, and e-Business as a whole.

Conclusion

Web Services continue to be an important emerging technology. They have important implications for e-Business in particular. We have developed a taxonomy to describe Web Services, as well as a grid to outline how they support e-Business when linked together in service chains. We have also developed a framework and architecture that support the creation and management of service chains. These chains will form the business processes of organizations as they adopt the service vision. They will also increase the ease of inter-organizational integration in the future. Being able to create and orchestrate service chains using a workflow interface will provide many benefits to users, including increased understanding of processes and higher productivity. Support for all kinds of e-Business from operational tasks to strategic ones will be provided by a tool that supports the quadrants of our grid. Our Web Service orchestration prototype is a first attempt at such a tool.

Acknowledgments

We would like to acknowledge the contributions of Schalk Pienaar, Nikhil Ravishankar, and McNeill Mendes toward different aspects of the project.

References

Almeida, J.P.A., Pires, L.F., and M.J. van Sinderen (2003), *Web services and seamless interoperability*, First European Workshop on Object Orientation and Web Services, ECCOP2003, Darmstadt, July 21-25, http://www.cs.ucl.ac.uk/staff/g.piccinelli/eoows/eoows-programme.htm .

Andrews, W. (2003), *Case Studies Illustrate Strategic Use of Web Services*, http://www4.gartner.com/pages/story.php.id.8842.s.8.jsp

Arkin, A., S. Askary, S. Fordin, W. Jekeli, K. Kawaguchi, D. Orchard, S. Pogliani, K. Riemer, S. Struble, P. Takacsi-Nagy, I. Trickovic, S. Zimek, (2002), *Web Service Choreography Interface 1.0*, http://wwws.sun.com/software/xml/developers/wsci/.

Berry, G., Chase, J., Cohen, G., Cox, L., and A. Vahdat (1999), *Toward Automatic State Management for Replicated Dynamic Web Services*, Netstore Symposium, Seattle WA, October 1999.

Bloomberg, J. (2002), *Web Services: Opening Soon for Business*, http://www.therationaledge.com/content/mar_02/f_webServices_jb.html

Box, D., D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, and D. Winer (2000), *Simple Object Access Protocol (SOAP) 1.1*, W3C, http://www.w3.org/TR/SOAP/

Brown, K. (2000), SOAP for Platform Neutral Interoperability, *XMLMag.COM*, http://www.xmlmag.com/upload/free/features/xml/2000/04fal00/kb0004/kb0004.asp

Christensen, E., F. Curbera, G. Meredeth, S. Weerawana (2001), *Web Services Description Language (WSDL) 1.1*, W3C, http://www.w3.org/TR/wsdl.html

Curbera, F., Y. Goland, J. Klein, F. Leymann, D. Roller, S. Thatte, S. Weerawarana, (2002), *Business Process Execution Language for Web Services 1.0*, http://www-106.ibm.com/developerworks/Web Services/library/ws-bpel/.

Ferris, C., and J. Farrell (2003), 'What are Web Services', *Communications of the ACM*, 46(6), 31.

Fremantle, P., S. Weerawarana, and R. Khalaf, 'Enterprise Service', *Communications of the ACM*, 45(10), 77-82.

Gottschalk, K. (2000), Web Services Architecture Overview: The next stage of evolution for e-Business, *IBM developerWorks*, http://www-106.ibm.com/developerworks/library/w-ovr/

Gottschalk, K., S. Graham, H. Kreger, and J. Snell (2002), Introduction to Web services

architecture, *IBM Systems Journal*, 41(2):170-177.

Johnston, S. J. (2002), State of Web Services, *InfoWorld*, http://www.infoworld.com/articles/pl/xml/02/02/04/020204plwebstate.xml

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

Leymann, F., D. Roller, M. Schmidt (2002), Web Services and Business Process Management, *IBM Systems Journal* 41(2): 198-211.

McKeen, J. D., and H. A. Smith (2002), 'New Developments in Practice: Managing the Technology Portfolio,' Communications of the AIS, 9(5), 1-25.

Nghiem, A, (2002), IT Webservices: A Roadmap for the Enterprise, Prentice Hall, USA.

Piennar, S. W., (2002), *The design and implementation of a flexible distributed decision support system generator using web services*, Unpublished MCom thesis, University of Auckland.

Stal, M., (2002), 'Web Services: Beyond Component-based Computing,' *Communications of the ACM*, 45(10), 71-76.