Logical Channels: Using Web Services for Cross-Organizational Workflow

Keywords: *Web Services, Workflow, Cross-Organizational, Integration, User Interface*

Universal cross-organizational workflow is one of the motivating goals behind the web services standards development efforts. Alternative flow languages, which allow web services to be combined and executed across organizations, have been created. Yet the mechanisms for true coordination between trading partners with workflow haven't been explored. We present three typical business scenarios involving multiple organizations in a workflow, and develop sequence diagrams of how cross-organizational workflow will unfold. In the process of doing this we identify logical channels of interaction that are distinct from the initial invocation of the flow – these include a connection between workflow engines in different organizations, as well as connections between monitor subsystems. We discuss the potential use of agents, and revisit the meaning of an interpersonal channel in a highly automated environment. After summarizing our model of logical channels, we look at how they might be realized physically, and conclude with general observations about the challenges of using web services for cross-organizational workflow.

Introduction

The end result of current web services efforts will probably be a set of standards that will permit cross-enterprise integration across a wide range of institutions. The exact form of these standards is contingent on a set of tactical skirmishes between several large software vendors, but the general direction is clear enough that its implications for business can be discussed. Most of the published work related to web services is in standards documents (Christensen, Curbera et al. 2001), (Ankolekar, Huch et al. 2002) (Jim Clark 2001). In a recent journal article, Leymann, Roller et al. (2002), present a detailed view of the intersection point of web services and business processes. Agarwal and Gondha (2002) provide an example of web services research in an academic setting, with links to examples of working services.

In order to motivate the discussion, we define three business scenarios which present problems for any method of cross-organization business process management. We discuss briefly the motivation and design of the basic web services standards. Then we explore the extension of web services into cross-organizational workflow.

In analyzing interactions between trading partners, a general marketing practice is to distinguish between different channels. Channels are meant in the broad sense of a conduit over which a variety of things flow, including goods, services, promotions, transaction information, and status updates (Kotler 2000).

We apply this practice to web services, and find that we can distinguish several logical channels. These distinctions, we think, will be useful to those involved in the design of web services standards and their accompanying software products, as well as those who are more generally interested in the technology.

Motivating Scenarios

Workflow distinguishes itself by seeking to abstract the movement of tasks from the tasks themselves (Aalst and Hee 2002). What we do inside a task changes, and the associated application also changes, but the movement of tasks to other people in an organization follows a more constant pattern that can be best captured at a different level of abstraction. When workflow goes across organizations, the focus on discovery increases – before we do something, we have to find the resource that can accomplish it. And visibility becomes much more of an issue – organizations don't like to reveal their inner workings to their trading partners. We outline here several realistic commerce-related situations.

> ***Scenario 1:*** *A company wants to find a shipper who can handle a heavy engine – a representative in the company initiates a search, which automatically finds a list of 3 suppliers, and chooses one on the combined criteria of familiarity and price. The service is automatically invoked.*

This is the type of scenario web services are envisioned to help solve. Such a business scenario can be manually accomplished through a buyer contacting over the telephone a set of potential bidders, soliciting bids, and negotiating a contract. But the contacting of potential bidders may be so tedious that the buyer will streamline the bidding process in a way that favors familiar, and perhaps more expensive, suppliers. Automation may increase the market power of the buyer by extending the list of possible bidders.

> ***Scenario 2****: The supplier shipping company chosen in the above scenario runs out of capacity and searches for a subcontractor, finds one, and delegates the task to the subcontractor. The customer has requested status messages, and the supplier does not want to reveal the identity of the sub-contractor, so the supplier collects and redistributes a re-branded set of error messages from the subcontractor back to the customer.*

This scenario brings in issues of visibility – the customer needs to know where things are, but the supplier may want to hide the identity of a subcontractor who might become a competitor.

> ***Scenario 3:*** *As in the above scenario, there is a subcontractor arrangement. Part way through the shipment, the customer sends a message to the prime contractor asking how much it will cost to divert the order to a different location. The prime contractor systems negotiate with the subcontractor systems, and present an increase in price. The customer accepts the increase, and the task in progress is modified through messages that flow through the prime to the subcontractor.*

This third scenario is by far the most complex, involving a change of a task already well underway in a multiparty transaction. In many businesses, it is likely that orders will be changed by customers while in process, and, in most cases, these change requests are difficult to manage.

Together, these scenarios present an escalating set of requirements for an automated business process management system that must work across organizations. Now we look at web services, and the way they might be invoked to handle these scenarios.

Invocation

Web services have been developed in reaction to current limitations of electronic commerce over the Internet (Glass 2002). The standards efforts that define web services are focused on allowing programs to take over the work that humans can currently accomplish on the web. On the web, transactions are often accomplished in the manner of figure 1.

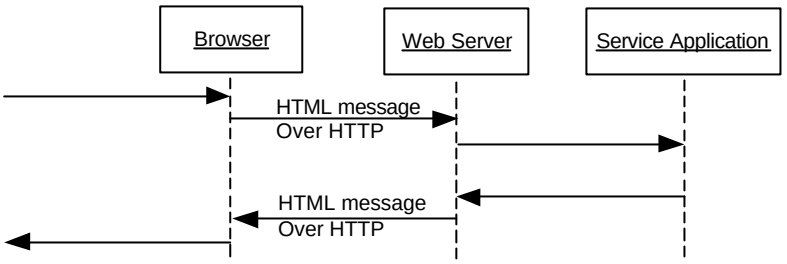

Figure 1. The World Wide Web, shown using sequence diagram conventions, which are defined in Rumbaugh, Jacobson et al. (1999).

The power of this approach comes from ubiquity – most potential customers have browsers, and most companies have web servers. But the primary goal for web services is to automate. Programming the browser to look like an individual is hard to do properly, as the machine can't differentiate between cosmetic and substantive changes to the web site. So the essential idea of a web service is to replace the person/browser combination with a program, to replace HTML with more structured XML (Ibbotson 2001), and to replace the Web Server with a SOAP interface intended to read and translate the XML messages and relay them to applications doing the work (Box, Ehnebuske et al. 2000). Web services structure the interaction so that a machine, rather than an individual, can drive the interaction.

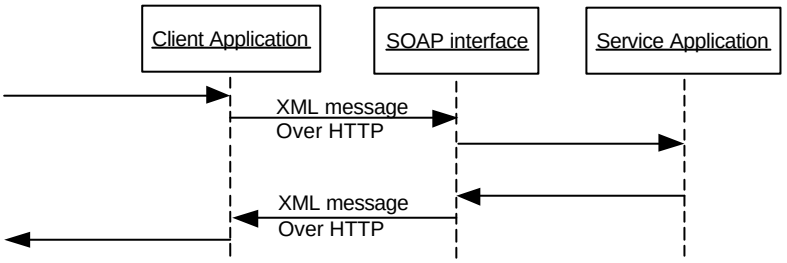

Figure 2. Web service invocation.

Considering our first scenario, it is apparent that before a service is invoked by a machine, it needs to be found. In the human realm, we find out the names of a site we want to visit through marketing channels of the company, including direct sales and advertising. We also use search engines, and, through a process simple for us and difficult for machines, filter the results into a set of candidate companies. In order for a machine to accomplish discovery, companies have to market their services to other machines, through what is essentially a yellow pages for computers.

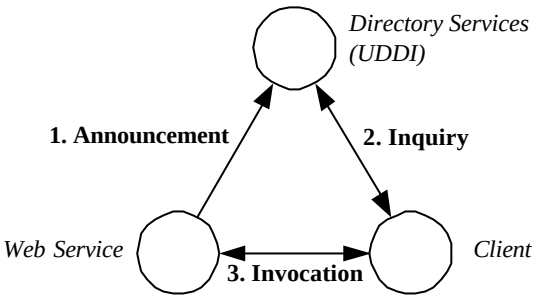

Figure 3. Discovery of a service.

Figure 3 shows this triangular relationship. And figure 4 shows how, assuming that a service has already registered with the directory, a client application can connect.

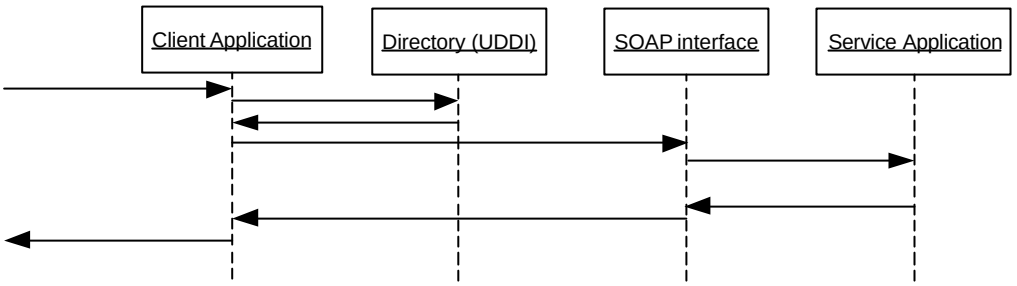

Figure 4. Discovery and invocation.

Someone – or some program, starts the client application, which accesses a directory outside the company, looking for a particular type of web service. It receives from the directory, UDDI (McKee, Ehnebuske et al. 2001), the location and detailed, machine-readable, information about the web service. This machine-readable information is written in Web Service Description Language, abbreviated as WSDL (Christensen, Curbera et al. 2001). The client then sends an XML message containing the parameters or data sets it wants processed to the SOAP interface, which reads the XML, and invokes the actual application computing the service. The result set goes to the SOAP interface, which passes it along in an XML message to the client application.

Flow

Web services, like program functions, can be composed, one after the other, creating a larger business process that can span company lines. Figure 5 shows how web services can be nested – how one service can invoke other services on behalf of a client.

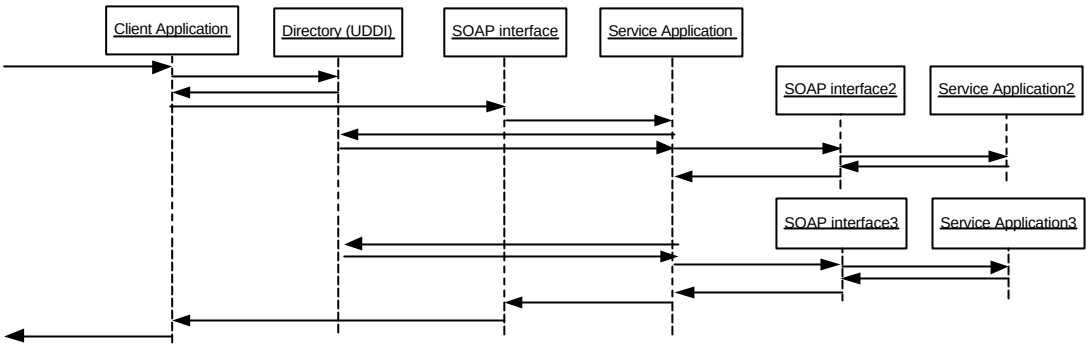

Figure 5. Nested web services.

Looking at the complexity of figure 5, a programmer would be tempted to simplify the entire flow as shown in figure 6.

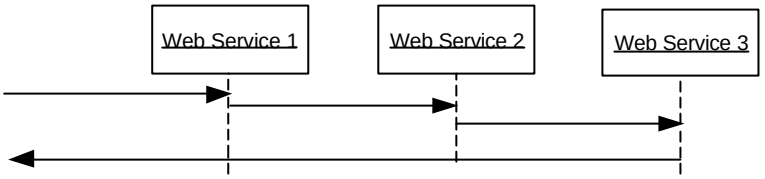

Figure 6. Services linked as functions.

This concatenation is easy to implement, but has a cost – inspecting the progress of an application becomes more difficult. When transactions are sub-second, inspection may not be an issue, but when transactions range over days, the ability to inspect is important. The above form of sequencing also makes it harder to program complex workflow rules, which may run two applications in parallel, and then combine results.

WSDL only describes a service, not a sequence of services, so in order to implement a workflow a new definition language needs to be developed. There are at least two languages defined at this time which can be used to combine web services. Web Services Flow Language, created at IBM is one such proposal (Leymann 2001); XLANG created at Microsoft is another (Thatte 2001). Both presuppose some form of workflow engine to read and run the workflow specification.

In implementations, it is likely that a web service will invoke a workflow engine, sending a business process specification, and the workflow engine will delegate tasks, possibly invoking other web services, as is shown in figure 7.

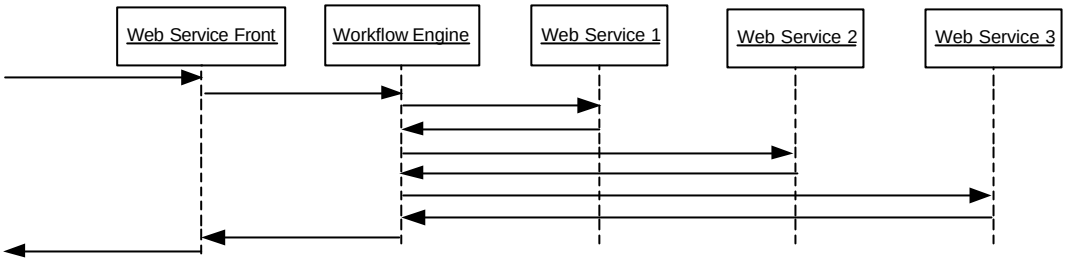

Figure 7. The workflow engine behind a facade.

In our second scenario, there are three organizations involved – a company acting as a customer, another acting as the service provider, and a third acting as a subcontractor to the second. Drawing organizational boundaries yields figure 8.

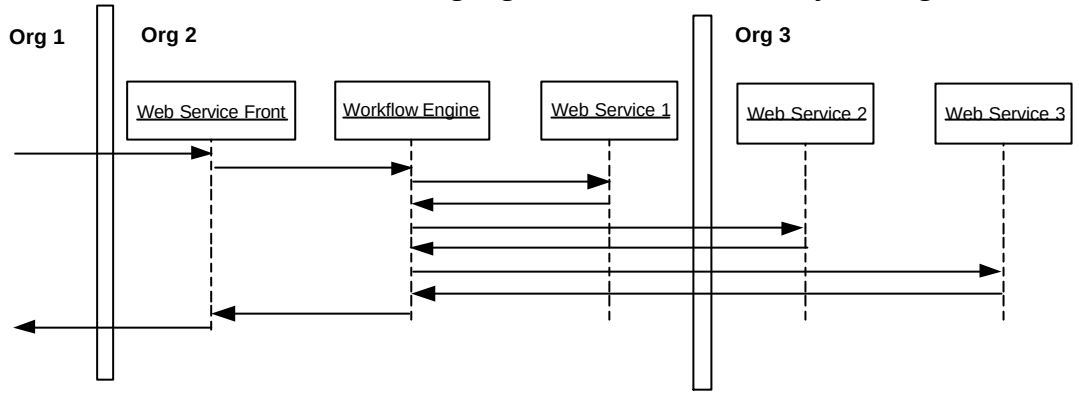

Figure 8. Organizational boundaries in a multi-party flow.

A workflow engine sits in the service provider, organization two, and choreographs the sequence of activities. Considering the role of the subcontractor (organization 3), it becomes apparent that the subcontractor is also likely to run a workflow engine. For the subcontractor will be managing multiple jobs from other customers. Adding into the diagram a workflow engine for the subcontractor, we arrive at figure 9.

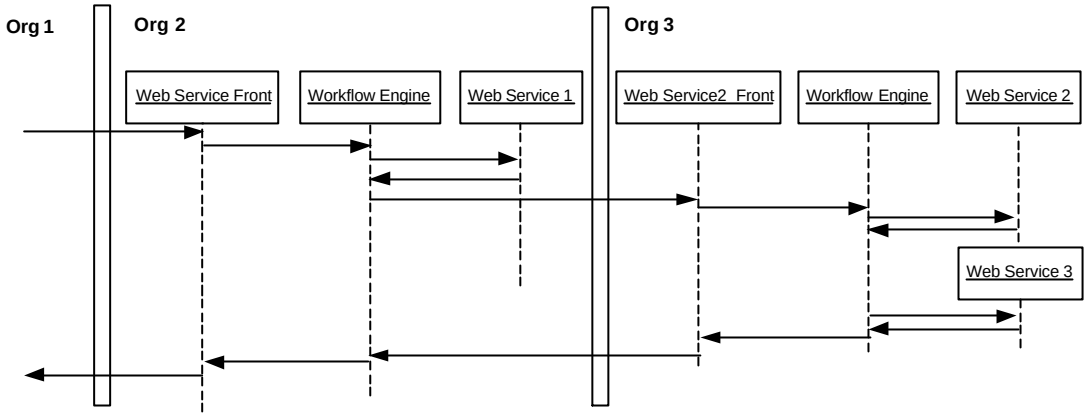

Figure 9. Multiple Workflow Engines.

Now we consider the third scenario, in which the customer wishes to change the order in progress. The workflow has already been invoked, so there is no use in calling the web service. What the customer wants to do is at a higher level of abstraction – to locate and modify a task that is already moving. Within an organization, the flexibility to actively track and intercept a task is one of the features that sell workflow engines. Cross-organizationally, this function is more difficult.

The most likely way to implement such a request would be to link the different workflow systems – both systems, if standardized, would understand a common set of requests regarding location of a task and diversion to a new destination. We show this in figure 10. Effectively, we have introduced a new channel of communication between the different organizations, linking the workflow engines together. We look at a potential third channel next.

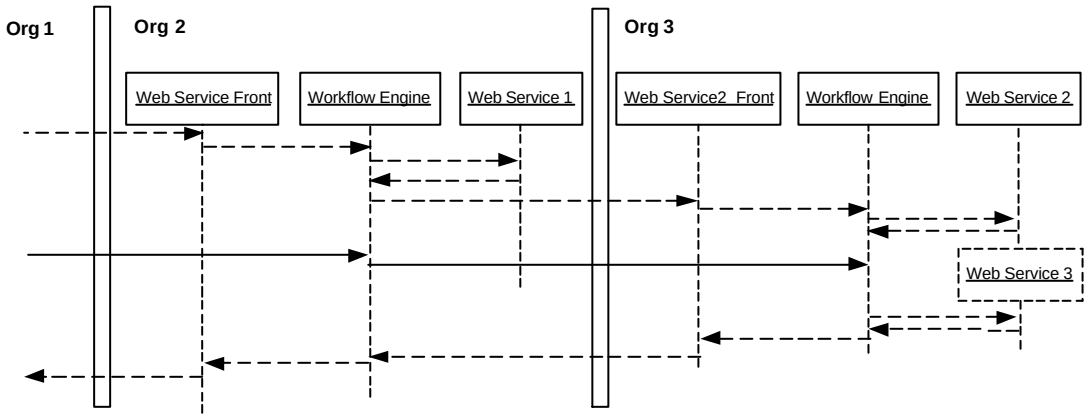

Figure 10. The flow channel – workflow engines communicating directly.

Monitoring

There are a number of business and technical events that a customer of a web service may legitimately want to monitor – in our scenarios, it would not be unreasonable to request updates as the engine moves from city to city. And if there is a quality of service agreement in place, monitoring may have contractual implications. In our scenarios, the supplier is using a subcontractor, and may not want the identity of the contractor revealed. To generalize, at each organizational boundary, there needs to be a subsystem that subscribes to certain events outside the company, and distributes the events inside and outside the company according to some set of visibility criteria designed to answer contractual obligations while protecting secrets. Leymann, Roller et al. (2002) discuss how public views on private flows might be constructed.

We insert into our diagram two monitoring systems, creating figure 11.

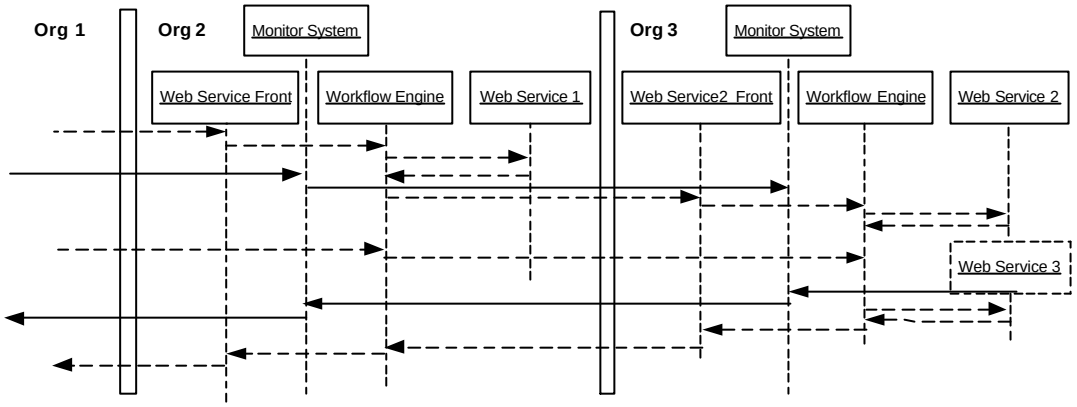

Figure 11. The monitoring channel.

The customer subscribes to events through the supplier's monitoring system, which in turn will subscribe to events in the subcontractor's system. Nickerson (2003) points out that monitoring happens at different organizational levels. It is likely that a manager, supervising a set of users, makes the determination of what will be monitored. It may be that the person running the task is looking at task specific messages, while the manager is looking at aggregated statistics.

Monitoring, logically, is distinct from both the workflow control channel and the original invocation channel. But monitoring, physically, might share underlying technology. Both workflow and monitoring systems can be built on top of information buses (Oki, Pfluegl et al. 1993; Hong, Lee et al. 2000; Cugola, Di Nitto et al. 2001).

Automated Negotiation

Web services are seen as part of a broader move toward building a semantic web (Berners-Lee, Hendler et al. 2001), a web where the symbols that are passed

around are described in a way that machines as well as humans can process. In the move to automate processes, automated negotiation is a current research topic (Singh 1998; Jain, Aparicio et al. 1999; Hendler and Mcguinness 2001).

Negotiation services can be seen as a specialized service utilized as part of a web services interaction.

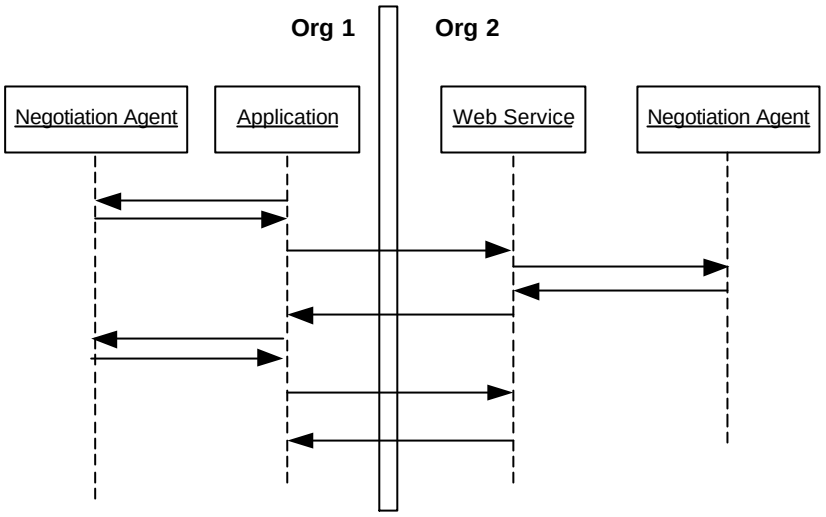

Figure 12. Automated negotiation in the background.

In figure 12, an application, prior to connecting with a web services, asks for guidelines for negotiation, and then invokes the service. The service utilizes its own agent, and returns a counter-offer, which the application checks with the negotiation agent. But there is an alternative – the agents can negotiate directly, and effectively become the interface to the other organization. The web services and applications become back-end functions, as in figure 13.

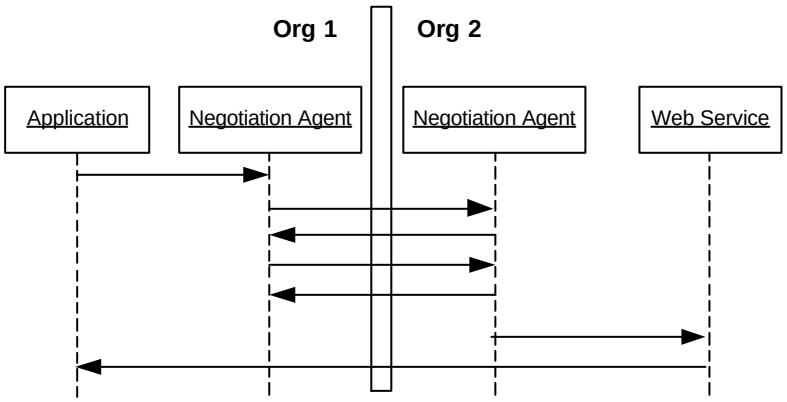

Figure 13. Automated negotiation channel.

This is the point of Singh (2002) – that our model of web services is based on invocation, whereas real business interactions are based on relationship, and, if we want to automate some of these relationships, our software will need to have the qualities of agents, which are persistent, keep histories, and can implement simple negotiation strategies.

Automated negotiation is still a new area. Automated negotiators need to understand the rules under which they operate, and probably need to be connected directly to each other in a reliable fashion in order contractually bound the outcome of the interaction.

The communication between automated agents will be a channel of communication between organizations. Singh (2002) suggests that we replace the invocation channel with this agent channel. For reasons we will explore in our discussion of physical channels, the invocation channel will probably survive.

Interpersonal Interaction

In suggesting we move from the semantic to the pragmatic web, Singh (2002) cites Morris (1938) who created the distinction *syntax, semantics, pragmatics*. But if we go back to Morris's inspiration, Peirce (1931), we will be led to reconsider if an automated web would truly be pragmatic. For in Peirce's universe, signs need to be interpreted, not just processed.

In highly automated environments, there are still people interpreting the results of transactions and setting the strategies for future transactions. So the interpersonal channel will and should still exist between companies. In the same way that integration within companies happens at both the machine and human level (Stohr and Nickerson 2002), so does integration across companies.

It is easy to imagine modifications of the scenarios we discussed in which a person-to-person communication would be necessary to either start something or fix something that has gone awry. In trading partner relationships, sometimes someone needs to pick up a telephone. The monitoring and flow change channels we talked about may trigger the need for direct interpersonal communication.

Besides this direct communication, there is also mediated communication. Nadin (1997) points out our increased automation does not eliminate the human component, but instead mediates the interaction. Even in the case of automated negotiation, there is interpersonal communication happening – through the competition or cooperation of the algorithms, which are designed by programmer analysts, who are effectively involved in a game with economic consequences.

If automated negotiation becomes the chosen method of doing business, those who design the negotiation algorithms become increasingly important – as do the interfaces that allow them to make changes and monitor progress.

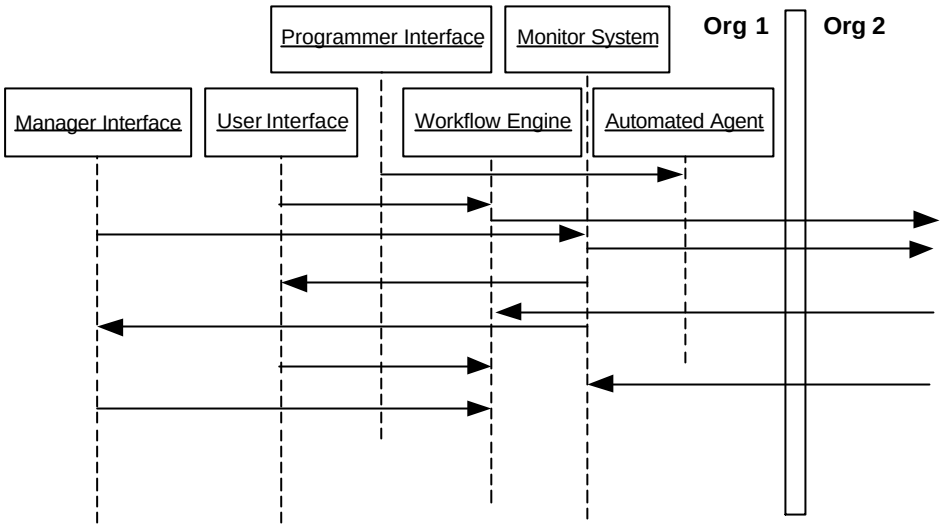

Figure 14. User interfaces

In Figure 14, we show several of the user interfaces that are part of a web services environment. A user interface commanded by someone in charge of a workflow task will initiate a flow in the workflow engine. Even before this, a programmer has updated the strategies in an automated agent. A manager will make requests of a monitoring system for aggregate event data, and the monitor system will communication both with the workflow task user and the manager. In the case of a high level task intervention, it is likely a manager would communicate directly with the workflow engine.

Summarizing the Logical Channels

Most descriptions of web services focus on interactions happening over the invocation channel – the initiating SOAP call and its result. We have built a model with larger set of channels. Our model is summarized in table 1 as a layer diagram, which is intended to be read from the bottom, starting with the invocation channel.

The second channel is flow – each participant in a multi-party transaction will probably be running a workflow engine, and these engines will talk to each other, especially to fulfill a complex request such as a modification to an in-progress flow.

The third channel is monitoring. Events need to be watched, but not all events are public. Monitoring subsystems will communicate directly to each other on top of a shared information bus, with visibility policies screening events before the subsystem propagates them across organizational boundaries.

Logical Channel	Description
Interpersonal Interaction	Unmediated, this a phone call from one partner to another to complete a negotiation or fix a problem. Mediated, it is the cooperation and competition that manifests through the software the trading partners build.
Automated Negotiation	Automated negotiation relies on shared protocols, data, and ontologies.
Monitoring	Monitoring across organizations calls for the forwarding of events based on visibility rules – some things are secret, some are not.
Flow	This is the interaction that workflow engines need across organizational boundaries.
Invocation	The SOAP calls that get a flow started. These SOAP interfaces are public, listed in a UDDI registry, and easily accessible. As part of this stage, the Web Service provider will authorize the client to open up private channels.

Table 1. Logical channels as a stack, starting with *invocation* on the bottom.

Automated agents who engage in negotiation concerning aspects of a contractual service need to share a language and an ontology – as well as a channel.

Regardless of whether automated agents are used, there are going to be humans in companies, and they will interact with each other through interpersonal channels. A number of business scenarios call for visits or phone calls. And, ultimately, even the automated agents are a form of interpersonal interaction, albeit highly mediated. The programmer will play an increasingly important role in defining the algorithms which negotiate. And all the monitoring is certainly geared for human eyes. The user interfaces throughout all of the web service channels are the ways through which highly automated businesses will be steered.

Taking the Logical to the Physical

In considering how cross-organization workflow will become universal, it is fair to ask if all of the logical channels can be realized through one physical channel. For almost all the communication we have discussed is happening through the physical connectivity of the Internet.

However, because of security concerns, almost all institutions have installed firewalls which restrict communication over the internet to several well-known ports. It is possible to open up a firewall in order to allow communication over a different port. But, to use a marketing analogy, it is always easier to use an existing channel than it is to create a new one.

SOAP can be run on any underlying protocol. However, in almost all cases, it is being run on HTTP, because HTTP's port is guaranteed to and from any company doing business on the web.

HTTP is not ideal for the scenarios we have outlined here. It is not reliable, in the sense that if a connection breaks, it is difficult to discover the state of a transaction that is in progress. HTTP is also intended as a synchronous protocol – the client waits while the server processes – which is a disadvantage when engage in long transactions.

Standards committees are proposing solutions to this problem – HTTPR promises reliability on top of HTTP, and BEEP offers a variety of configurable

communication behaviors (Rose 2001; Todd, Parr et al. 2002). It should be noted that not everyone is in favor of creating a more complex protocol – Lessig (2001) points out that we run the risk of ruining the end-to-end, content neutral nature of the net by tinkering in order to achieve noble-sounding goals such as quality of service.

For monitoring and true event-based workflow, an event-based approach, using the publish/subscribe paradigm, is desirable. There are many commercial messaging products which offer publish/subscribe, and there is a standard, Java Message Service, which specifies both publish/subscribe and store and forward capabilities(Monson-Haefel and Chappell 2000). But this approach requires an administrator to change a firewall.

If an entire industry agrees on an infrastructure, then firewalls will be opened up – otherwise, it is a risky proposition for a vendor to assume customers and suppliers will gladly reconfigure their security to let through a new stream of information.

Conceptually, the workflow change channel and the monitor channel could share an information bus. If such a sophisticated messaging channel were in place and universal, all of the logical channels might make use of it.

Most likely, there will be at least two physical channels, two ports, involved in web services. It makes sense to allow the initial discovery and connection between partners to happen over a universally open channel, which today is HTTP. Then, once connections have been made, tokens can be exchanged, and more sophisticated messaging channels can be established, possibly on top of a messaging engine. The two step process will probably persuade companies to open up a message bus, given they can control some form of token-based security which is granted in the initial invocation phase.

Conclusions

Web services can provide a mechanism for cross-organizational workflow. But the current ways of thinking about the services need to be expanded into a consideration of the logical channels of communication. In non-automated commerce, there are many touch points between trading partners, including initial contact, price negotiations, logistical problem solving, and ongoing monitoring of the relationship. Automated commerce also needs to consider multiple logical channels of communication.

In particular, invocation of a process is a different, and much simpler form of communication, from adjustment of a process in progress. Complex workflow engines will need to connect to each other to properly handle this kind of change. Monitoring of transactions across organizations is complicated by issues of corporate visibility and quality of service contractual obligations – the monitoring subsystems will need direct links.

Singh (2002) makes the point that invocation is not the proper model – that automated agents should be the main channel of communication. It is true that interactions between trading partners tend to be repeated, and so software that has a memory for past interactions will have an advantage over software which treats each connection as new. We think that soon automated agents will converse directly over a

separate channel, but that the simple invocation channel will remain the initial starting point for automated organizational interaction.

Finally, the interpersonal channel has a role in highly automated cross-organizational workflow. While the goal of web services is to reduce routine interpersonal interaction, the non-routine will require intervention. And even those firms engaged in automated commerce will still be practicing a form of interpersonal interaction. Their interactions will be mediated through agent software, but the agents will have been designed by a team of programmers and strategists. For this reason, user interfaces, ranging from event monitoring to agent design, deserve attention as an important part of the cross-organizational system that web services promise.

References

Aalst, W. v. and K. v. Hee (2002). Workflow Management: Models, Methods, and Systems. Cambridge, Mass, MIT.

Agarwal, M. and A. Gondha (2002). An Introduction to the Stevens Web Services Laboratory, http://attila.stevens-tech.edu/webservices/labintro.htm

Ankolekar, A., F. Huch, et al. (2002). Concurrent Execution Semantics for DAML-S with Subtypes. The First International Semantic Web Conference (ISWC).

Berners-Lee, T., J. Hendler, et al. (2001). "The Semantic Web." Scientific American(May).

Box, D., D. Ehnebuske, et al. (2000). Simple Object Access Protocol (SOAP) 1.1, www.w3.org/tr/soap

Christensen, E., F. Curbera, et al. (2001). Web Services Description Language (WSDL) 1.1, http://www.w3.org/TR/wsdl

Cugola, G., E. Di Nitto, et al. (2001). "The JEDI event-based infrastructure and its application to the development of the OPSS WFMS." Software Engineering, IEEE Transactions on **27**(9): 827 -850.

Glass, G. (2002). Web Services: Building Blocks for Distributed Systems, Prentice Hall.

Hendler, J. and D. L. Mcguinness (2001). "DARPA Agent Markup Language." IEEE Intelligent Systems: 15(6):, 72-73.

Hong, H.-S., B.-S. Lee, et al. (2000). A Web-based transactional workflow monitoring system. Proceedings of WISE 2000: 1st International Conference on Web Information Systems Engineering.

Ibbotson, J. (2001). XML Protocol Usage Scenarios, http://www.w3.org/TR/2001/WD-xmlp-scenarios-20011217

Jain, A. K., M. Aparicio, et al. (1999). "Agents for Process Coherence in Virtual Enterprises." Communications of the ACM **42**(3): 62-69.

Jim Clark, C. C., Kanaskie, Betty Harvey, Jamie Clark, Neal Smith, John Yunker, Karsten Riemer, (2001). ebXML Business Process Specification Schema, UN/CEFACT and OASIS.

Kotler, P. (2000). Marketing management. Upper Saddle River, N.J., Prentice Hall.

Lessig, L. (2001). The Future of Ideas : the fate of the commons in a connected world. New York, Random House.

Leymann, F. (2001). Web services flow language., http://www-3.ibm.com/software/solutions/webservices/pdf/WSFL.pdf

Leymann, F., D. Roller, et al. (2002). "Web services and business process management." IBM Systems Journal **41**(2).

McKee, B., D. Ehnebuske, et al. (2001). UDDI Version 2.0 API Specification, http://www.uddi.org/pubs/ProgrammersAPI-V2.00-Open-20010608.pdf

Monson-Haefel, B. R. and D. Chappell (2000). Java Message Service, O'Reilly.

Morris, C. W. (1938). Foundations of the Theory of Signs. Chicago University Press, Chicago.

Nadin, M. (1997). The Civilization of Illiteracy. Dresden, Dresden University Press.

Nickerson, J. V. (2003). Event-based Workflow and the Management Interface. HICSS.

Oki, B., M. Pfluegl, et al. (1993). The Information Bus-- an architecture for extensible distributed systems. Proceedings of the Fourteenth ACM Symposium on Operating Systems Principles, ACM.

Peirce, C. S. (1931). Collected papers of Charles Sanders Peirce. Cambridge, Harvard University Press.

Rose, M. (2001). The Blocks Extensible Exchange Protocol Core, http://www.ietf.org/rfc/rfc3080.txt?number=3080

Rumbaugh, J., I. Jacobson, et al. (1999). The Unified Modeling Language Reference Manual. Reading, Mass., Addison-Wesley.

Singh, M. P. (1998). "Agent Communication Languages: Rethinking the Principles." IEEE Computer **31**(12): 40-47.

Singh, M. P. (2002). The Pragmatic Web: Preliminary Thoughts. NSF-OntoWeb Workshop on Database and Information Systems Research for Semantic Web and Enterprises.

Stohr, E. T. and J. V. Nickerson (2002). Enterprise Integration: Methods and Direction. Competing in the Information Age: Align in the Sand. J. Luftman. New York, Oxford University Press.

Thatte, S. (2001). XLANG Web Services for Business Process Design, http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm

Todd, S., F. Parr, et al. (2002). A Primer for HTTPR, http://www-106.ibm.com/developerworks/webservices/library/ws-phtt/