zur Muehlen, M.; Nickerson, J.V.; Swenson, K.D.: Developing Web Services Choreography Standards – The Case of REST vs. SOAP. Decision Support Systems 37 (2004), Elsevier, North Holland. *(to appear)*

Developing Web Services Choreography Standards – The Case of REST vs. SOAP

Michael zur Muehlen[1a], Jeffrey V. Nickerson[a], Keith D. Swenson[b]

[a] *Wesley J. Howe School of Technology Management*
Stevens Institute of Technology
Castle Point on the Hudson
Hoboken, NJ 07030, USA
{mzurmuehlen|jnickerson}@stevens.edu
[b] *Fujitsu Software Corporation*
3055 Orchard Drive
San Jose, CA, 95134, USA
kswenson@fsc.fujitsu.com

Abstract

This paper presents a case study of the development of standards in the area of cross-organizational workflows based on web services. We discuss two opposing types of standards: those based on SOAP, with tightly coupled designs similar to remote procedure calls, and those based on REST, with loosely coupled designs similar to the navigating of web links. We illustrate the standardization process, clarify the technical underpinnings of the conflict, and analyze the interests of stakeholders. The decision criteria for each group of stakeholders are discussed. Finally, we present implications for both the workflow and the wider Internet communities.

Keywords

Workflow, Web Services, Choreography, Interoperability, Standards, Process, Integration, REST, SOAP

[1] Corresponding author, e-mail: mzurmuehlen@stevens.edu

1. Introduction

1.1. Motivation and Outline

Web Services Choreography aims at the coordination of long-running interactions between distributed parties, which use web services to expose their externally accessible operations. The coordination of long-running interactions is necessary in almost any cross-organizational interaction, in areas ranging from electronic publishing to supply chain management. In this article, we will trace the development of the many standards in this area [3, 4, 9, 18, 20, 30, 51].

This is an exploratory case study [60], motivated by two observations. First, that the current interest in web services is directing attention to issues that have a longer history in the workflow community. Second, that the debate over web services choreography standards appears to be deeply influenced by architectural style, understood by relatively few. We started with one question – whether the technical debate has merit and should be better understood by a wider community. We also ask whether the battle is only about the technology. We found answers to both questions.

The study presented makes use of a large set of documents related to the standards process – the standards themselves, as well as the public commentaries in the form of mailing list entries, web sites, articles, and blogs. One of the authors, Keith Swenson, was involved in several of the standardization efforts discussed. And, through email, we were able to gain both factual clarification and elicit opinion from other standardization participants.

Our work should be of interest to others engaged in research on the development of technical standards. For instance, this research might be combined with other cases into a multi-case analysis. Furthermore, our work should also be of interest to the wider community involved in both the creation and adoption of the standards discussed here – the history of the case, the analysis of the technology, and the discussion of the decision process may inform those individuals and companies who must decide for or against a standard in this area.

The case study timeline starts with the creation of workflow standards and ends with two connected events – the consolidation of several vendor-driven initiatives in form of the BPEL4WS (Business Process Execution Language for Web Services) standard, and the incorporation of REST (Representational State Transfer) ideas into the latest SOAP (Simple Object Access Protocol) specification. For the questions we have defined, these milestones provide a natural boundary to the case. The moment of this writing is a natural break in the standards development process – there will undoubtedly be more developments to come, but the current situation is a good time to reflect, as this reflection may have an effect on future decisions of standards makers and adopters.

We classify existing standards proposals into two categories: REST-oriented and SOAP-oriented standards, and discuss the implications of these design philosophies. Readers with background in this area may notice that REST and SOAP aren't necessarily opposites – REST is an architectural style, and SOAP is a general protocol that can be used as an element of many different architectures. Yet these terms are an easy, and often-used, shorthand for an active debate. In section 3 of this article we discuss in detail where the arguments of this debate lie. For now, we suggest thinking of REST as representing a navigational style of design, and thinking of SOAP as representing a procedural style.

First, we explain what process integration standards are. Then we trace the chronology of standards development in this area. Next, we discuss the nature of the technical argument used by the standards groups. Finally, we discuss alternate decision strategies that may explain the actions of the different parties involved in the creation and use of these standards. We distinguish between the choices that users and vendors may hold.

1.2. The Domain Process Integration Standards

Organizations wishing to link their local processes across a network face a number of areas in which they need to achieve agreement about which formats and protocols to use during the interaction. Figure 1 shows a generic scenario for inter-organizational process integration. The individual participants have internal operations in form of private processes (A, B, and E), which are not exposed to outside trading

partners. As a façade, public processes (C and D) are created, and their activities (C_1 through C_3 and D_1 through D_3) serve as contact points for the other party. In order to automate this interaction, the two parties need to agree on a number of standards. First of all, they need to specify the semantics of the public process operations, and whether these operations receive or send data to the other party. Furthermore, the structure and content of the messages that are to be exchanged need to be defined, for example using an industry-specific format. In addition, the sequence of messages needs to be defined, including ways to detect messages that are lost or out of sync. Finally, the partners need to agree on access, addressing and authentication mechanisms that allow one party to send a message through the network gateway of the opposite party.

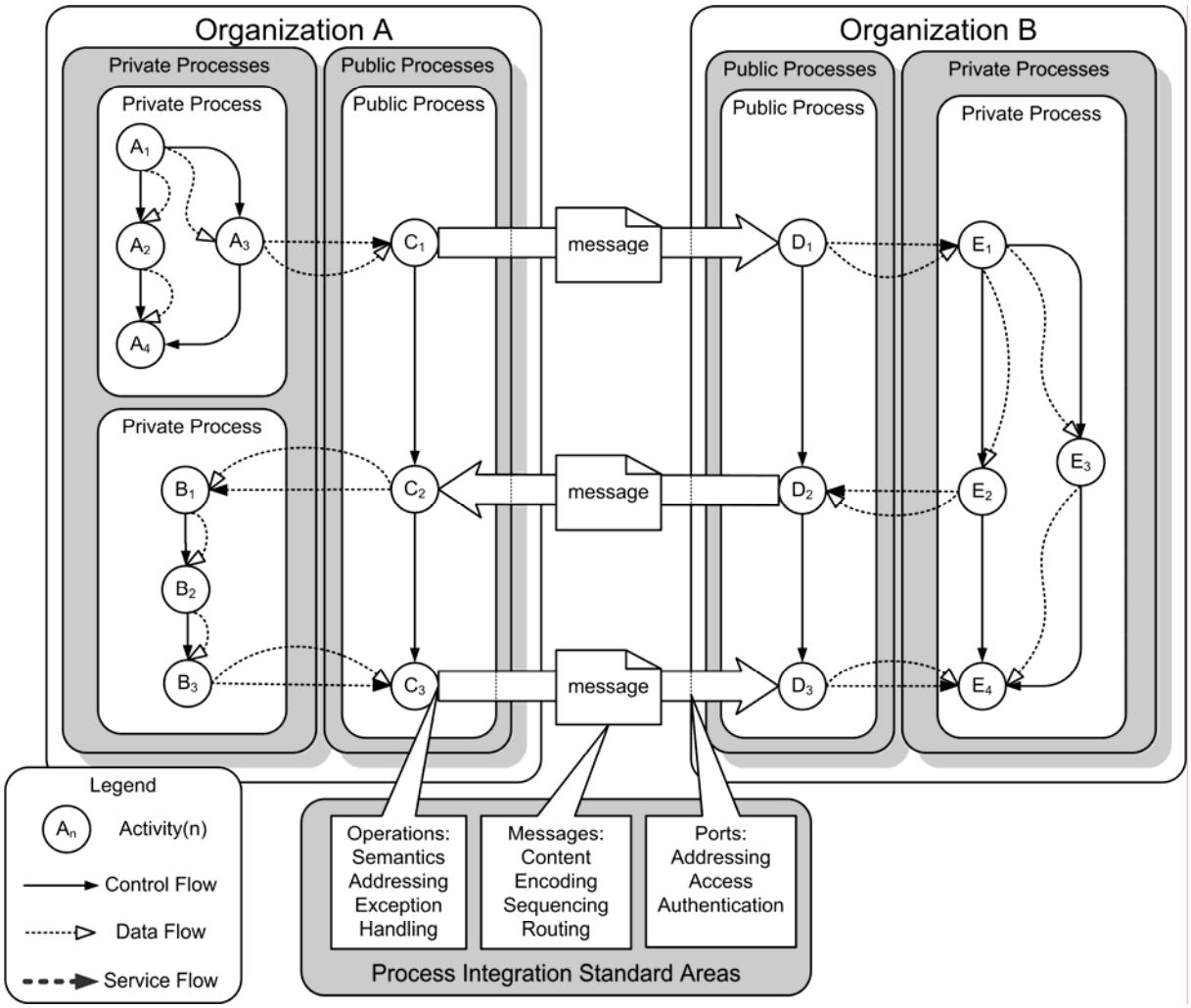

Figure 1: Scope of Process Integration Standards

With an increasing number of trading partners the individual negotiation of these aspects becomes cumbersome as the number of interfaces in the network increases geometrically. The use of general standards for process integration purposes becomes economically desirable.

2. A Timeline of Workflow and Web Services Choreography Standards

Several standards bodies and software vendors have proposed solutions in this area, but we need to go back further in order to explain how these proposals emerged. While the World Wide Web Consortium (W3C) first used the term web services in 2001, the underlying idea of electronic coordination for long-running business-to-business transactions has a long history. Workflow interoperability standards have been developed since the middle of the 1990s, and some ideas of these standards have found their way into the current web services choreography submissions. Standards for electronic data interchange have an even longer history, dating back to the 1970s.

Figure 2 shows the evolution of process interoperability standards over time. In the following section, we describe the evolution of the standards depicted in Figure 2 and discuss the relationships between the individual standardization groups in terms of cooperation and competition.

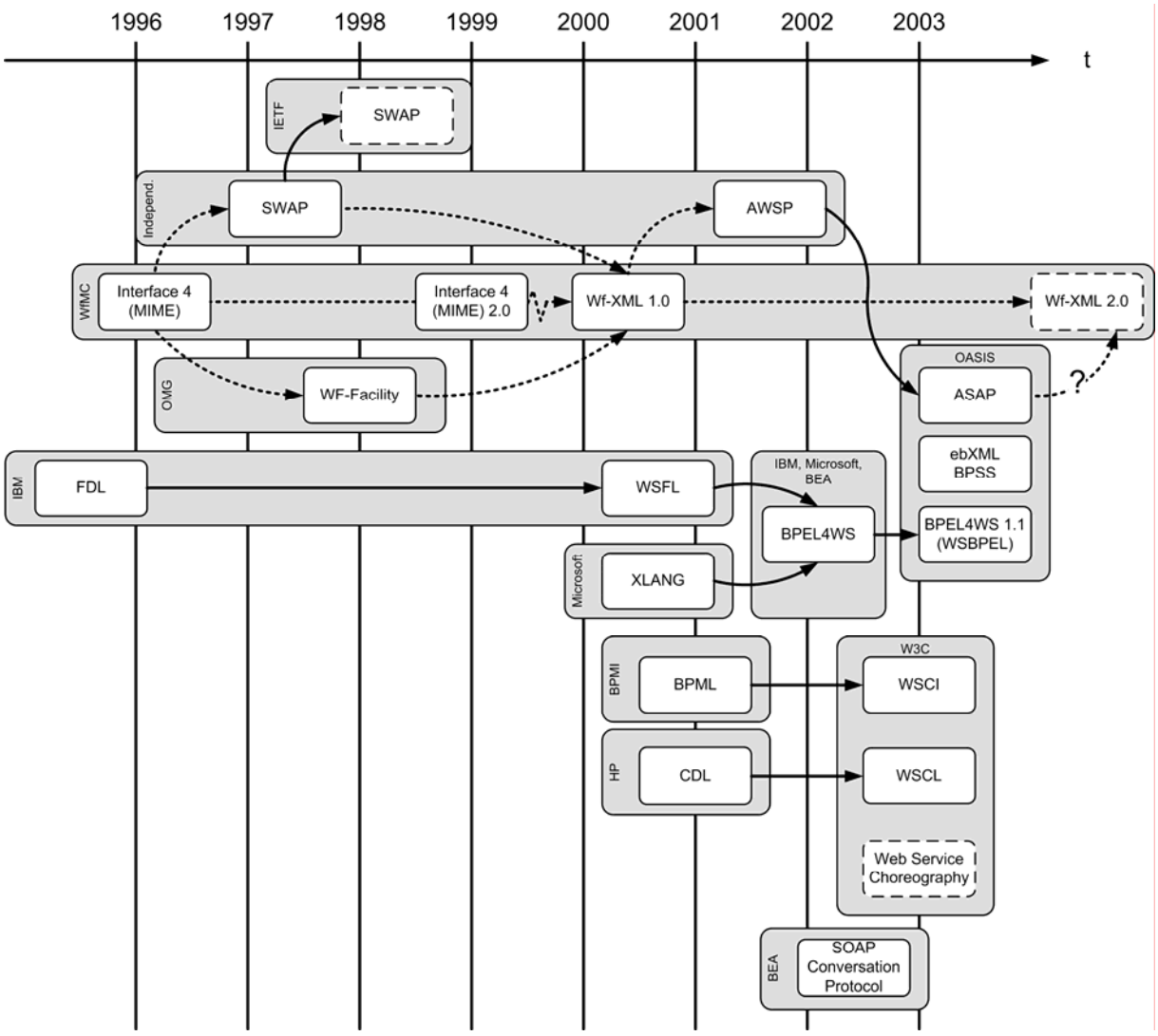

Figure 2: Evolution of Workflow and Web Service Interoperability Standards

2.1. The Charter of Standards

The development of workflow systems originated in the office automation prototypes of the late 1970s, and the first successful commercial ventures were started in the mid-1980s. By the early 1990s workflow management systems had reached a state of technical maturity that allowed for their use in a variety of enterprise scenarios. It became apparent that larger companies would be faced with the coexistence of workflow systems from different vendors, and that interoperability between these systems required the establishment of a common standard. In 1993 a number of influential CIOs, organized in the

Black Forest Group, chartered workflow vendors to come up with a standard to ensure the interoperability of their systems. This led to the founding of the Workflow Management Coalition (WfMC), which in turn produced a glossary [57] and a reference architecture [25] with five standard interfaces, which defined different modes of interaction between a workflow management system and its environment. While a standard API is part of the reference architecture [56], a separate interface (IF 4) defines operations for remote process invocation and control. The specification of this interface consists of an abstract specification [56], which defines the operations and data formats, and context-specific bindings, which realize the operations of the abstract specification using a particular technology. The first binding completed in 1996 was based on e-mail, and systems could be coupled using MIME messages, which was demonstrated in a prototype setup but did not receive adoption in practice. When the WfMC started working on IF 4 the focus of interoperability was on the coupling of workflow systems within one enterprise, but it became clear early on that the fast adoption of Internet technology would allow for the integration of business processes across enterprises, and that a new version of the standard would be necessary. This is the start of a pattern – outside broad-based standards development changes the course of the workflow-related standards.

2.2. From OMG to Wf-XML

Shortly after the completion of the IF4 MIME binding, the OMG (Object Management Group) issued a request for proposals for a workflow service based on CORBA in May 1997 [34]. Several WfMC members submitted proposals, and a unified team of several initial submitters under the lead of Marc-Thomas Schmidt (IBM) created the OMG workflow facility specification [35] (dubbed jFlow at the time of its creation). This specification was officially adopted in August 1998 and revised in 2000. The workflow facility introduced a workflow object model that consisted of a process factory, which offered a method to create individual process instances. This is the first of many recurring patterns in the development of the standard – object-oriented thinking is applied in the design of the standard. This object model idea was discussed among the WfMC members several times, but no immediate plans to

amend IF 4 were made, instead the original version of the IF4 was updated to a release 2.0, which was published in 1999. This is the second part of the pattern – the new design is blocked.

A number of WfMC members felt that a lightweight alternative to the existing IF 4 specification was needed and began developing an alternative protocol, which was called the Simple Workflow Access Protocol (SWAP) [12, 47]. The basic idea behind SWAP was the use of standard Internet protocols for workflow interoperability. A workflow model would be represented by a Uniform Resource Identifier (URI) and could be manipulated using a number of commands based on HTTP 1.1 extensions (LISTINSTANCES, CREATEPROCESSINSTANCE etc.). An initial draft was completed early 1998, and the authors submitted it to the Internet Engineering Task Force (IETF), which was the desired standards body for this endeavor. This is the third part of the pattern – the blocked designers propose a new standard.

By the end of 1998, after two birds-of-a-feather meetings [27], it became clear that the IETF was not going to adopt the SWAP specification, and members of the SWAP working group returned to the WfMC, where the ideas of the SWAP proposal were enhanced with the experiences of the OMG submitters. In addition, the proprietary extensions of the HTTP protocol were removed and replaced with standard HTTP POST commands [48]. The result was the Wf-XML specification, which was published in 2000 and revised in 2001 [58]. Wf-XML is described as an XML binding of the WfMC IF4, but it is not similar to any previous WfMC specification. What is notable about the process we have documented is how migratory it is – small groups form, merge with other groups, break off, and reform in different venues.

2.3. Web Services and Web Services Choreography

Parallel to the workflow-centric debate in WfMC, OMG and IETF, XML was gaining momentum not just as a uniform data formatting standard, but also as a means to exchange standardized messages between applications, which required the specification of a messaging protocol. We can see this as a

second example of external standards having an effect – just as HTTP changed workflow, XML changes messaging.

Wf-XML represents the results of the growing popularity of XML, but it was not the only solution developed. In December 1999 the World Wide Web Consortium (W3C) created a mailing list for XML protocol related issues, to coordinate the disparate efforts. In March 2000 , PRUD'HOMMEAUX identified 27 separate initiatives in this area, out of which the Simple Object Access Protocol (SOAP) [13] and the Web Services Description Language (WSDL) [17] emerged as the core components of today's web service framework.

In order to continue the narration and illustrate the technical debates, we need to first explain in detail some of the ideas behind web services. While SOAP focuses on the encoding of messages and the definition of remote function calls in XML, WSDL provides a mechanism to describe a set of remote function calls (operations) as a coherent port which can be addressed by a SOAP message. SOAP and WSDL enable application designers to create Web Service wrappers around their applications that expose application functionality through a standardized interface definition language. Figure 3 shows the possible interactions in the context of SOAP and WSDL. The client of an operation is described as a requester, while the server is described as a provider. Requests can be unidirectional (*request*), or bidirectional (*request-response*). If a process context has been established, notification and solicit-response interactions can be used, i.e. if the communication between requester and provider has already been established. In order to handle asynchronous communication, correlation IDs can be used to link requests, notifications, and solicit-response interactions.

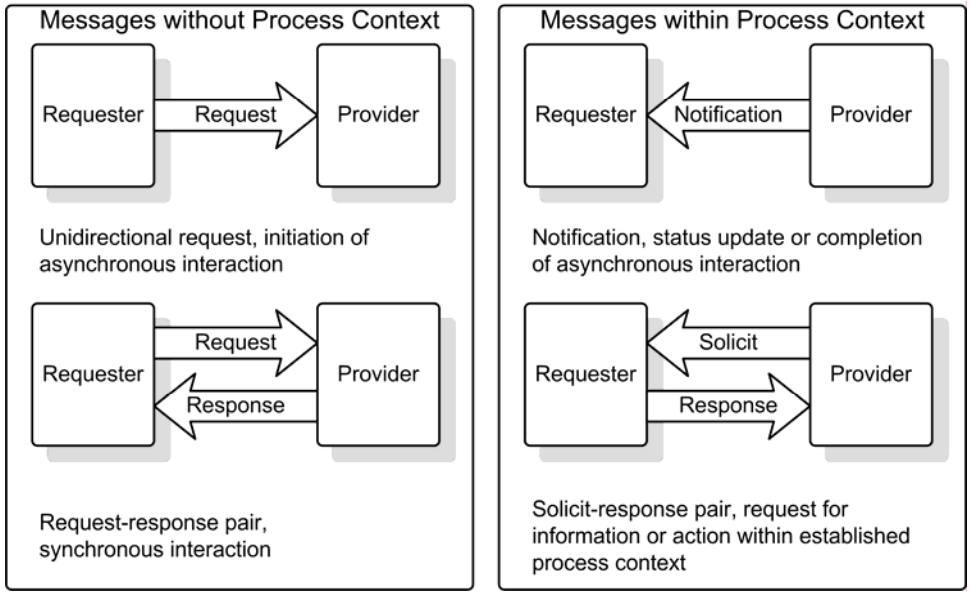

Figure 3: Peer-to-Peer Interactions using SOAP and WSDL

It should be noted that SOAP and WSDL do not provide mechanisms for the sequencing of messages beyond single message pairs (e.g. request-response). Figure 4 shows a complex business scenario using web services that illustrates this problem. A service provider can decide at which level of granularity he wishes to expose his internal operations. In the case of workflow A, the entire workflow is wrapped into a single operation which is exposed as a service, while in the case of workflow B the individual activities B_1 and B_2 are accessible as operations. Hidden from the clients of these operations is the outsourcing of activities A_3 and B_3, which in turn invoke an external web service to achieve their goals.

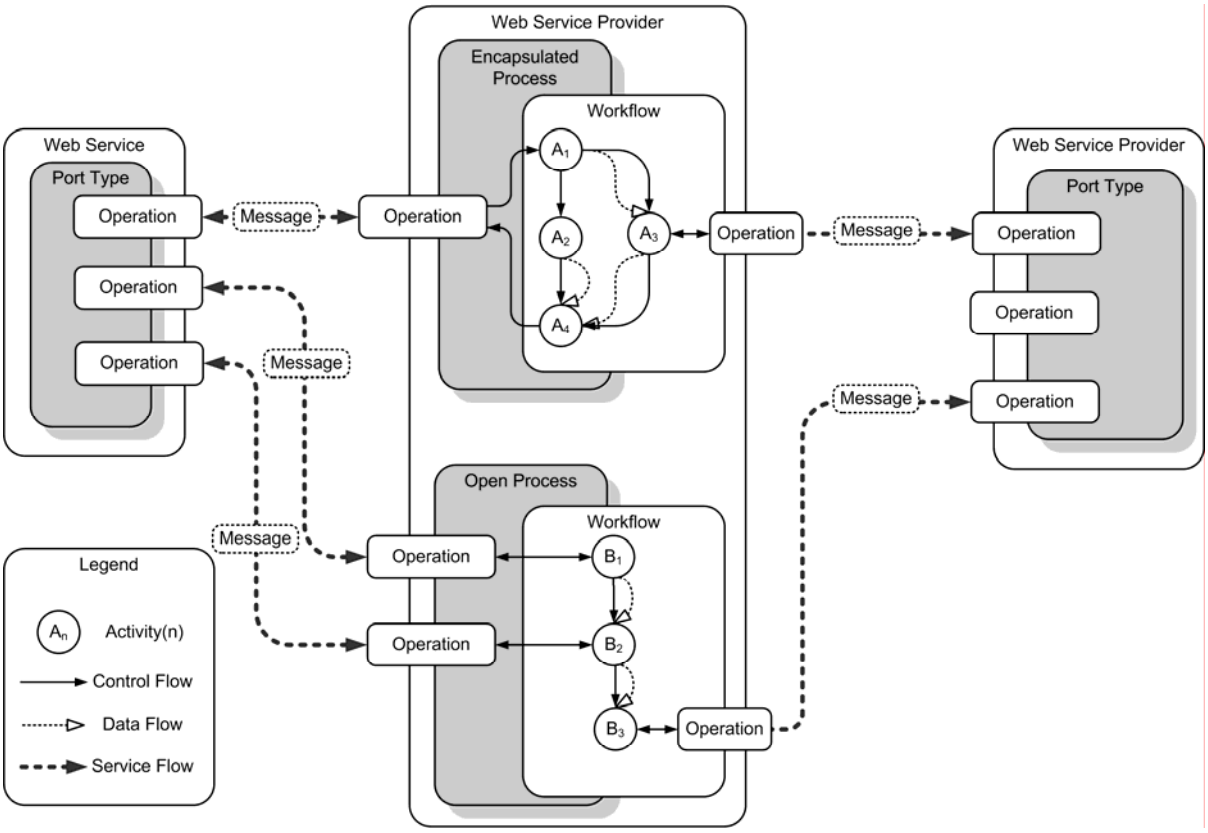

Figure 4: SOAP-oriented Integration of Ports

A client may choose to use all three operations in order to obtain a particular service. However, it is not apparent that the workflow *A* operation is independent of the workflow *B* operations, and that the workflow *B* operations share a control and data flow dependency, i.e. B_2 may only be invoked after B_1 has successfully been completed. In other words, complex business scenarios that require the sequencing of several message pairs cannot be described sufficiently using SOAP and WSDL, but an additional standard is needed. To further illustrate the need for such a standard, Figure 5 shows an example of a typical purchasing scenario. In this scenario, customer and supplier engage in a well structured exchange of information. First, the customer sends a request for quote to the supplier, which is answered with a quotation for the desired goods or services. The customer then sends a purchase order (which refers to the quote) to the supplier, who in turn acknowledges the receipt of the purchase order. While the purchase order is being processed the client decides to change an item and submits a request to change the purchase order, which is acknowledged or rejected by the supplier. Finally the supplier sends a delivery note to the

customer, which in turn is acknowledged by the other party. In order to facilitate such a complex interaction, an information system needs to keep track not only of the individual request-response or notify-response message pairs, but it also needs to correlate the different message pairs to an overall context, so that it can identify messages that are duplicate or out of sync. In essence, the description of the overall interaction requires a process model, and numerous formalisms are available for this purpose.

The PSL project identified more than 26 different approaches to process modeling in a 1998 study [28]. In addition to traditional modeling methods, several workflow-specific process modeling languages have been proposed. For example, VAN DER AALST and KUMAR have defined a formal Petri-net based description of a process, designed to be read and processed by a workflow engine [1], while the WfMC proposed an XML rendition of their Workflow Process Definition Language in 2003 [59].

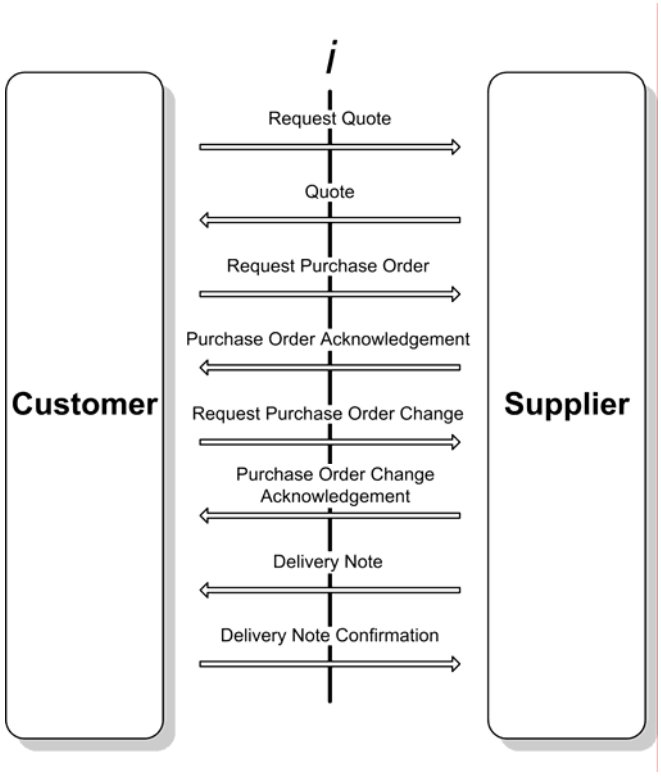

Figure 5: Complex interaction between customer and supplier

Several standards for the description of the large-scale interaction between web service-enabled applications as an extension of WSDL have been proposed since 2001. IBM had developed the Flowmark

Definition Language (FDL) as part of the workflow management system FlowMark (later called MQSeries Workflow) in 1994 [31]. In 2001, IBM published the specification of the Web Services Flow Language (WSFL) [30], which was largely derived from the FDL meta model [33]. WSFL allows for the definition of flows that may include the invocation of external web services, and that can be exposed as web services themselves. FDL allows organization elements to be specified, including the participants who perform workflow activities, whereas WSDL is designed to represent only fully automated processes.

In parallel to IBM's effort, Microsoft developed their own specification language for complex interactions with XLANG [51], which shared many characteristics with WSFL. In early 2001, both parties joined forces and created BPEL4WS [18]. While the development of the language was driven by IBM, Microsoft and BEA, it became apparent that the backing of a vendor-neutral standardization organization would increase the chances of the standard of being adopted. In May 2003 the vendors handed the BPEL4WS specification in a surprising move over to the Organization for the Advancement of Structured Information Standards (OASIS), where the working group has grown to more than 90 individual members, representing more than 40 different companies. These moves of standards between different groups seem to be thought through – those involved in the standards process appear to understand the nuances of each standards body, including not only their degree of friendliness toward vendors, but also the nature of their bylaws.

Parallel to the individual efforts of IBM and Microsoft, the Business Process Management Initiative (BPMI), an industry consortium led by the Silicon Valley startup Intalio, presented plans for an XML-based Business Process Modeling Language (BPML), which was backed by SUN and SAP, among others. An early draft of the BPML specification, written by a subset of BPMI members, was submitted to the W3C for adoption in the summer of 2002 and became known as the Web Services Choreography Interface (WSCI). However, the development of BPML continued in parallel to (and decoupled from) WSCI. When WSCI did not lead to the immediate formation of a working group within W3C, the submitting vendors returned their efforts to the advancement of BPML.

This can be explained by the sense of ownership and influence in a vendor-run consortium versus the free-for-all atmosphere that can be found in more independent standardization groups. For instance, BPMI is mainly run and controlled by the software company Intalio, whereas W3C is a non-profit venture managed by the academic organizations ERCIM, MIT and Keio. By submitting a proposal to a non-profit organization the submitting organization typically grants adopters of the technology a royalty-free license to use the published standard, which may not be in the best economic interest of the submitting party. OASIS is dominated by vendors. W3C comes out the efforts of Tim Berners-Lee, and is not dominated by vendors – it strives to preserve the open standards-creating process which started with early Internet standards such as HTTP.

To add to the standards proliferation, the W3C had already received another submission for web services choreography, which was the result of the e-speak research project at Hewlett Packard Laboratories. The HP Conversation Definition Language [29] was submitted to W3C in March 2002 and named the Web Services Conversation Language (WSCL) [10]. However, not much activity ensued around the WSCL submission while several workgroup meetings around WSCI were held in the fall of 2002 and the spring of 2003. The W3C has reacted to the existence of multiple standards in the web services choreography with the establishment of a web services choreography working group in January 2003. Instead of creating a standard of their own, this group has set out to define requirements for a Web Services Choreography language, and may eventually endorse one of the proposed standards (and/or contribute to its development). As a first step in this effort an initial requirements document was published in August 2003 [6]. Since its inception, the focus of the working group has been on fundamental constructs of a choreography definition language. This has led some vendors to withhold their endorsement of this effort, as they see it conflicting with the contents of their own standards.

While Wf-XML was formally adopted by the WfMC, and updated to the current version 1.1, the creators of SWAP felt that some of the purity of the SWAP approach was lost in the process. This concern over purity was what led to the break-off from the IF 4 specification earlier – we see the pattern begin to repeat.

In addition, WSDL and SOAP had not been standardized at the time Wf-XML was developed, so realizing a Wf-XML interaction in a SOAP/WSDL environment was not natively supported by the standard. As a straw man proposal, SWENSON and RICKER created the Asynchronous Web Services Protocol (AWSP) [49], and circulated a draft proposal among WfMC participants. Due to resource constraints, the WfMC was not able to accommodate the AWSP proposal in a short timeframe, so the authors of AWSP turned to OASIS and founded a technical committee that is chartered to develop a protocol for the invocation of asynchronous web services, the Asynchronous Service Access Protocol (ASAP) [38]. The pattern completes – a group wishes a cleaner design, breaks off from one group, and proposes a new standard.

Since OASIS is also the venue for the ebXML initiative (refer to section 2.4), current discussions point to the direction that ASAP might become part of the ebXML framework [50], which is described in the following section.

2.4. User-driven Standards with Process Content

Besides the standards discussed in sections 2.1 through 2.3 a number of standardization initiatives exist that provide process chorography as part of their overall framework, but are not necessarily focused on choreography. We present these as they are driven not by vendors, but by users of the standards. These users join together in industry coalitions. Their standardization process is in marked contrast to the ones we discussed above; discussion centers mainly on the business, not the technology – and the standards groups tend to be more stable, with less jumping and merging. We present first the most general of these efforts, and then consider industry specific examples.

In September 1999 the United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) and OASIS founded the ebXML initiative with the goal of making electronic commerce applications based on XML accessible for small and medium-sized organizations that may not have the means to purchase a large software package exclusively for this purpose, such as an EDI solution. The ebXML framework consist of standards that describe the structure and access mechanisms of a central

registry for web services, as well as mechanisms to describe these services in a standardized manner. Part of these specifications is a high level process specification schema standard (BPSS) [20], which describes the interaction between two parties in an ebXML scenario. UN/CEFACT was the standardization body for the UN/EDIFACT standard, which found widespread adoption in the commercial world, and is a *de jure* standardization body. OASIS and UN/CEFACT target organizations, for which the traditional UN/EDIFACT standard is too expensive to implement.

Moving to industry-specific examples, we look first at manufacturing. RosettaNet is a consortium of more than 400 companies representing supply chain participants and solution providers in the semiconductor, electronic components, telecommunications, and information technology industry. The RosettaNet General Implementation Framework [39] consists of domain-specific ontologies (business and technical dictionaries), transport-oriented specifications, and more than 70 specific process examples for supply chain processes (partner interface processes, or PIPs), e.g. the update of a purchase order. Instead of describing a generic language for process choreography, the designers of the RosettaNet PIPs chose to specify specific modes of interaction that may occur in a supply chain scenario.

In financial services, the Society for Worldwide Interbank Financial Transfer (SWIFT) has standardized messages that are exchanged between partners in the financial industry. Instead of specifying process standards, SWIFT has published a modeling methodology for the analysis and design of business-to-business processes [40], but in addition provides a value-added network for the exchange of financial data (SWIFTNet). Their efforts have folded into an ISO effort, ISO15022, which eventually will work on top of XML. What characterizes the ISO effort is the tight focus on the business transactions – the goal is not to solve a general computing problem, but to solve the problems of banking, clearance, and settlement. The exact messages are defined. And the users of the messages understand the messages and the sequences – so that the fully general description and discovery mechanisms of web services seem superfluous, at least to the older companies.

In the healthcare industry, Health Level Seven (HL7) has published standards for messages that are exchanged between providers of healthcare services, such as hospitals or doctors, and health insurance

providers. While the official HL7 standards focus on the message structure and content, recommendations were published regarding the coordination of messages [23, 24].

Within the insurance industry the Association for Cooperative Operations Research and Development (ACORD) has published standard data formats for insurance products such as life insurance, property and casualty, and reinsurance. While most ACORD specifications prescribe only the data format used for insurance-specific messages, newer developments aim at the description of XML-based transactions in terms of request-response pairs [5]. SWIFT, HL7 and ACORD are widely used in their respective industries.

In looking at these successful industry standards, it is clear that they are very different than the general standards discussed before. For the industry, it appears at first glance that full generality is not needed. And that the implicit knowledge of the industry participants is all that is needed to choreograph well-understood transactions. For industry groups, technical arguments are less important than social ones. If an entire industry can agree on something, then the benefits of agreement can outweigh any inefficiencies of implementation.

A deeper look suggests that there will at some point be a debate between horizontally and vertically driven standards – for the horizontal standards may favor the new company entering a space, or the established company in a different industry expanding its breadth of service.

3. The Technical Debate: REST vs. SOAP

Organizations engaging in cross-organization process integration specify the structure of their interaction at the model level. The actual enactment of the interaction happens at the instance level, where one instance embodies one distinct case of the interaction. For this purpose, the provider party typically supplies a process factory, which is capable of creating individual process instances upon request. A process instance can then be manipulated as a state machine. Workflow management solutions are typically composed of multiple independent state machines. At the process level, the overall state of the process (started, suspended, aborted) can be manipulated by the client. At the activity level, the execution

of an individual work item can be represented in form of a state machine. Finally, the data manipulated in the context of a process instance can have an individual state as well. (e.g., an invoice can be in the states unpaid, audited, paid).

At an abstract level web services choreography standardization efforts can be classified into standards that favor a tight coupling of components using SOAP, and standards that favor a loose coupling of components following the architectural principle of REST. We compare the two approaches using the above generic model, which represents the typical design found in workflow applications.

3.1. REST-oriented Integration

Representational State Transfer (REST) is an architectural style described by FIELDING [21]. REST is in many ways a retrospective abstracting of the principles that make the World Wide Web scaleable. FIELDING argues that unlike transactional systems, in which servers need to maintain complex sessions with clients, the web reduces the server's obligation that arises from a self-contained question to a simple response. In many cases, the response involves sending a pre-computed set of data, the representational state, across the network. The client can essentially navigate across a wide range of existing, pre-computed resources by following links from resource to resource. There are many objects, each identified by a URI, and very few methods (PUT, POST, GET, and DELETE in HTTP 1.0, while HTTP 1.1 allows extensions). In REST, each request sent to an object results in the transfer of a representation of this object (typically in form of an XML document). This document provides the client with the opportunity to change the state of the object by navigating to a different URI which is referenced in the document. The documents are self-contained and can be transformed through intermediaries, for example proxies.

Transferred to the area of cross-organizational workflow, a REST-style workflow consists of a process factory object, which is referenced by a URI. A request to the process factory results in the creation of a process instance, which in turn can be referenced using a URI. In its purest form, a HTTP GET command sent to either of these URIs results in the return of an XML document. In the case of the process factory, the XML document may contain the properties of the process model represented by the process factory

(for example quality of service information), whereas the XML document returned by the process instance may contain information about the current state of the process instance. In essence, every component of a cross-organizational process is identified by an individual URI, that is, each externally accessible process, activity, or operation is identified by a URI.

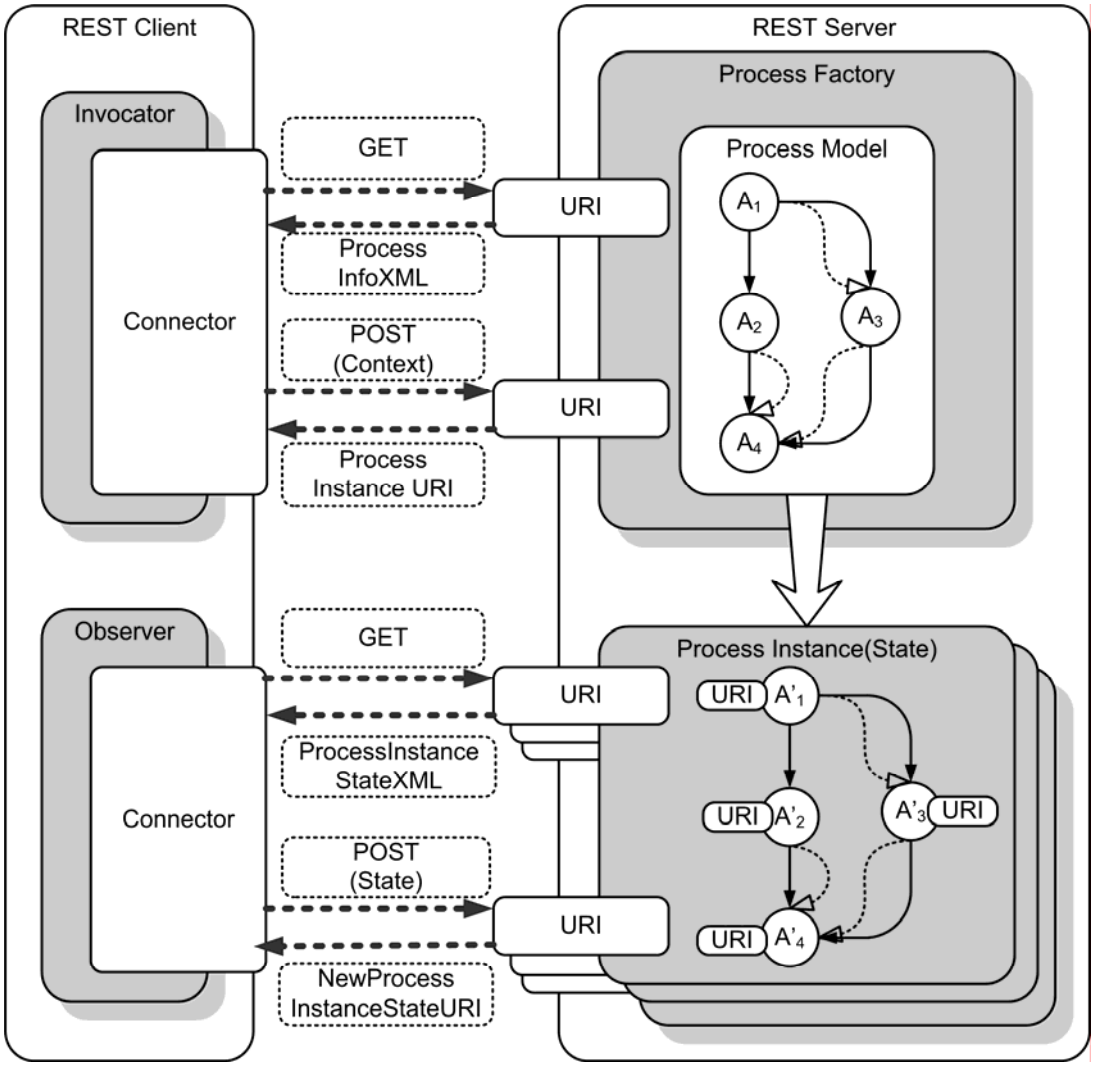

Figure 6: Pure REST-style Workflow

Figure 6 shows an example for a pure REST workflow application. All the client needs to know about the remote process is the address of this process in form of a URI, which is typically a combination of the machine name, a distinct object type and a process ID. A GET request sent to this URI results in an XML document that gives more information about the process, for example the URI where the process can be

invoked. The client then sends a POST operation to the process invocation URI which contains the context data (payload) for the process instance. The process factory creates a process instance and returns the URI of the process instance to the invocator. The subsequent interaction between client and process instance is also characterized through the use of GET and POST commands, but in this case the POST command is used to manipulate the state of the process instance, until it is completed.

Note that the process client does not need to know any details about the implementation of the process, but the data formats that are exchanged need to be agreed upon by both parties. In other words, the same client interface can be applied to all types of processes that may be provided by a server, given that the client can generate (and parse) the data structures that the server requires.

Several standards for REST style workflow interaction have been proposed, namely SWAP, Wf-XML [26, 52, 58], AWSP [49] and ASAP [38]. Instead of relying on the HTTP 1.0 commands, these standards provide higher level operations that are specifically designed for the interaction with remote processes. Figure 7 shows the same workflow as above using ASAP commands [38]. In this scenario a client can send a "*GetProperties*" command to the process factory URI to receive details about the implementation of the process instance. Upon the receipt of a "*CreateInstance*" request the process factory creates a new process instance. With "*ListInstances*" a client can retrieve a list of process instances that are accessible. Similar to the pure REST approach, the process instance can be addressed with "*GetProperties*", "*SetProperties*" and "*ChangeState*".

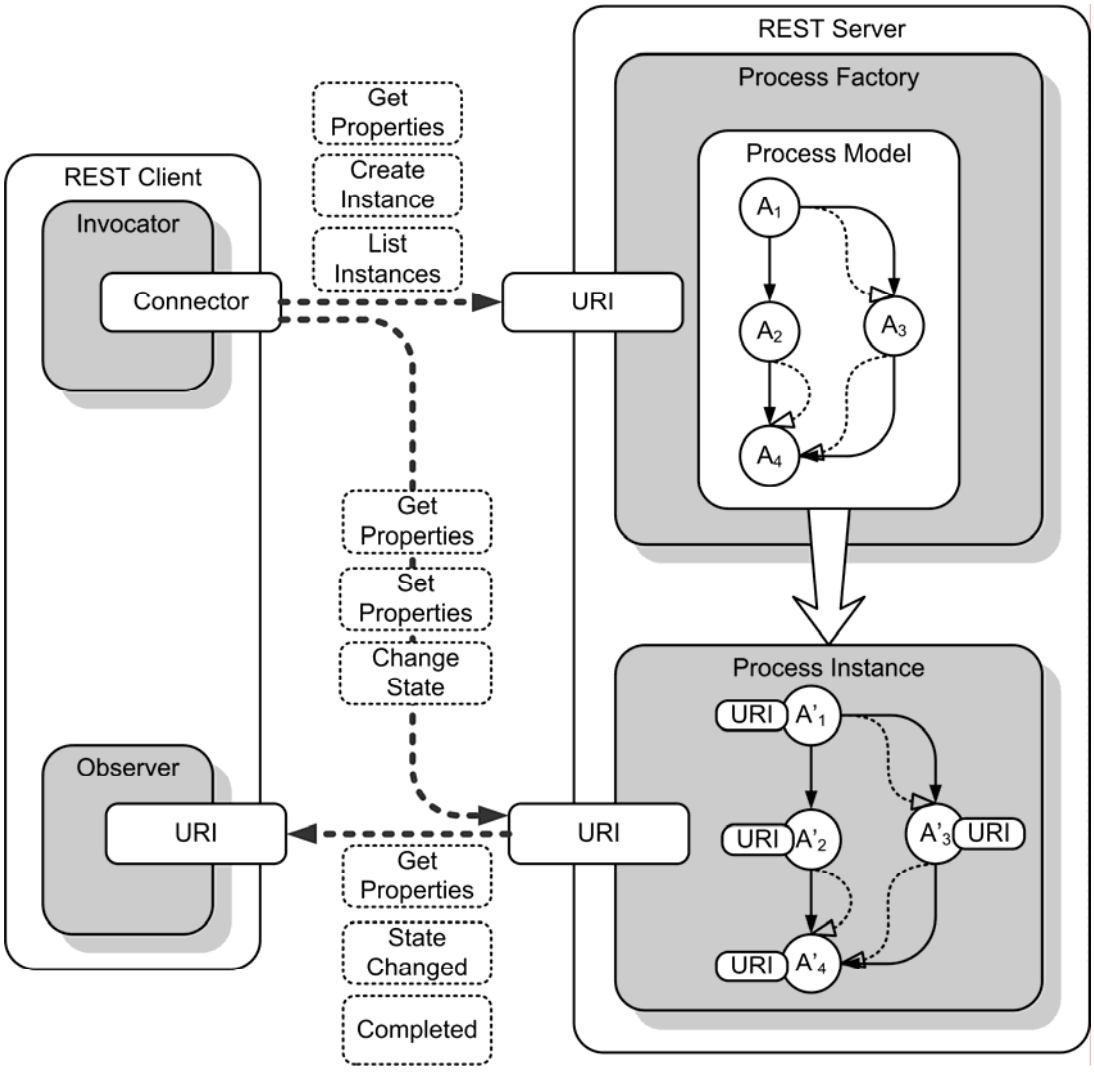

Figure 7: Advanced REST-style workflow (using ASAP messages)

3.2. SOAP-oriented Integration

The other approach to cross-organizational process integration relies on WSDL to describe the endpoints of the communication and SOAP to provide a basic messaging standard. These SOAP-oriented solutions are characterized by the fact that every operation is represented by its own communication endpoint rather than a message type. Pure SOAP solutions do not distinguish between the process factory and process instances like REST solutions. Instead, the mechanisms of creating and managing process

instances are hidden behind the web services façade of the SOAP endpoint. Figure 8 shows an example for a SOAP-style purchasing process.

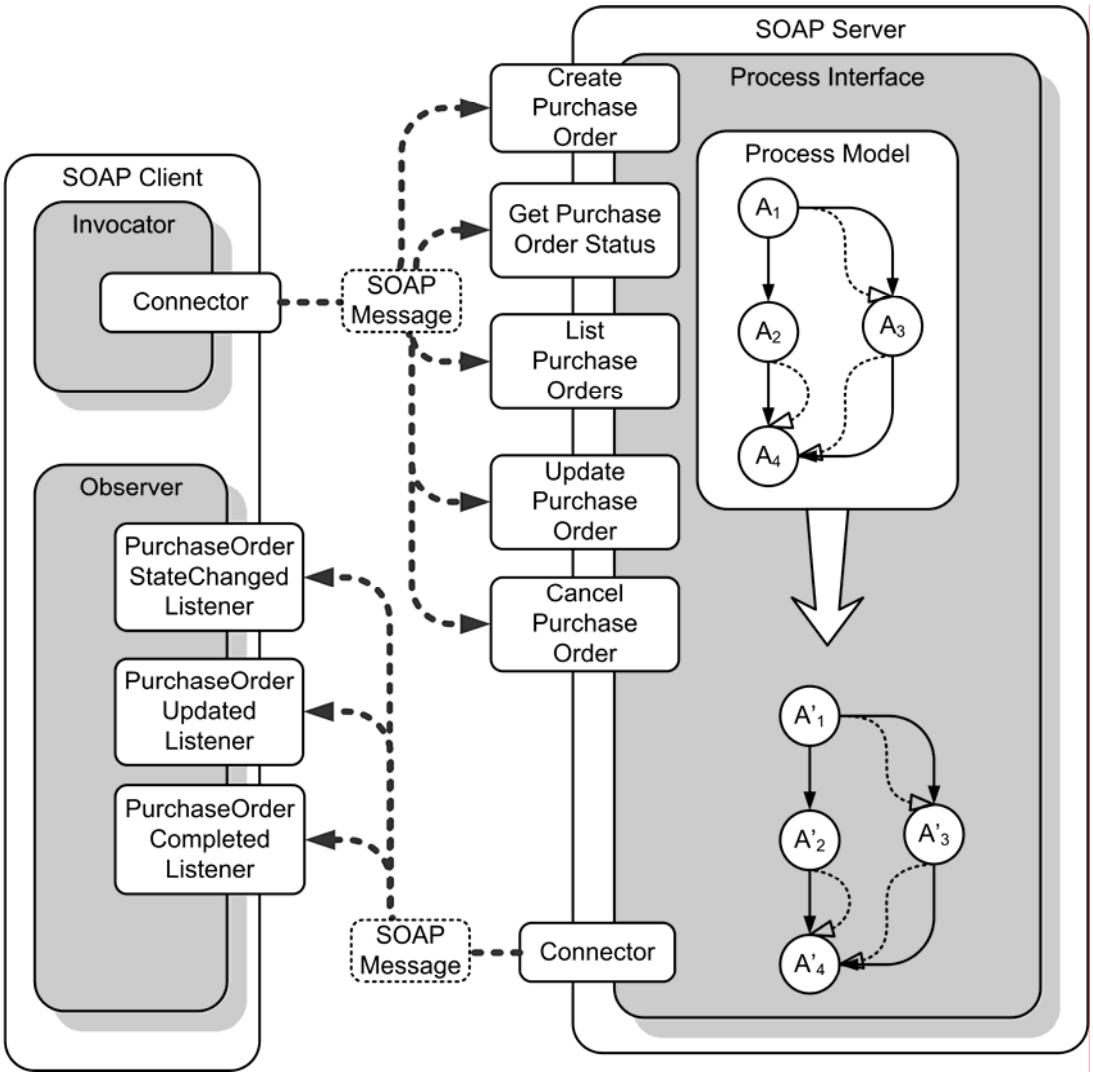

Figure 8: SOAP-style Workflow

Each operation that can be performed on the purchase order object is represented through its own endpoint. A client can create a purchase order object by invoking the "*CreatePurchaseOrder*" operation, upon which the identifier of the purchase order instance is returned. Subsequently, the purchase order instance can be manipulated through explicit "*UpdatePurchaseOrder*" and "*CancelPurchaseOrder*" operations. The client needs to pass a reference to the instance that is the target of the operation with his message, because individual instance are not distinguished by separate operations. Most of the current

web services choreography standards follow this SOAP-oriented design, most notably BPEL4WS [2] and WSCI [4]. Instead of providing an explicit operation for the creation of a process instance, the service provider may choose to create process instances en passant when an operation like "*UpdatePurchaseOrder*" is called (in fact, this is encouraged by the creators of BPEL4WS). However, this may lead to differences between the first call of an operation and subsequent calls, an issue that has not yet been resolved by the standards bodies.

One way to overcome the overloading of operations is the application of the REST-style separation between a process factory interface and a process instance interface, instead of a generic process object interface. Figure 9 shows the structure of a SOAP-style workflow that takes this distinction into account. In order to create a process instance, the client needs to send a SOAP-encoded message to the CreateInstance operation of the Process Factory Interface. This process factory interface may be generic, so the client may be required to send information about which process instance it wishes to instantiate. Upon creation of the process instance the client can manipulate this instance by referring to the methods of the Process Instance Interface. Since the individual process instances share a common port for operations, the client needs to include the identifier of the desired process instance in its message. On the observer side, the client needs to provide interfaces for active notification, such as the message that a particular process instance has completed.

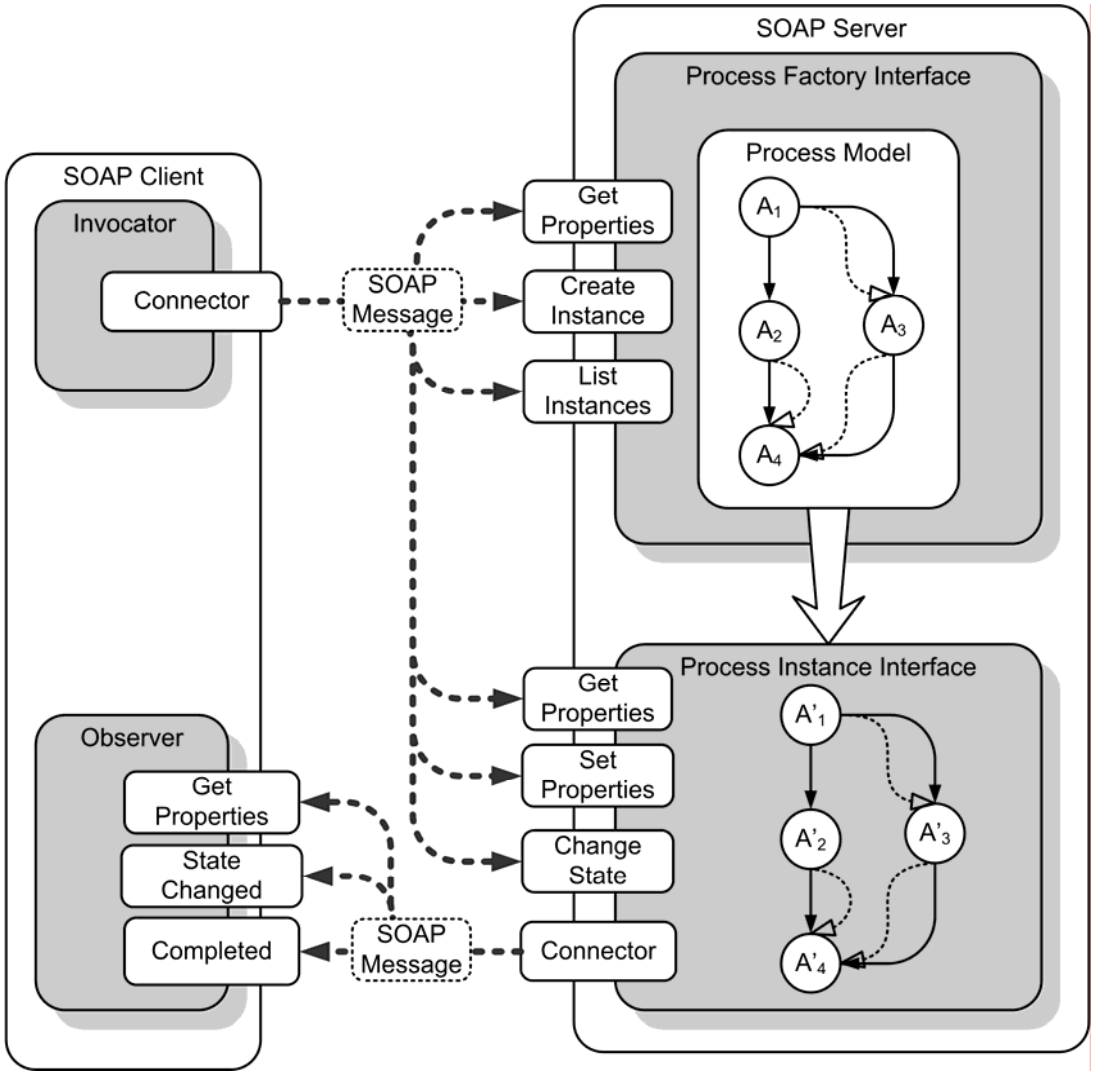

Figure 9: SOAP-Style Workflow separating Factory and Instance Interfaces

3.3. Advantages and Disadvantages

The advantages and disadvantages of REST-based and SOAP-based approaches to process integration are summarized in Table 1. While the core benefits of SOAP lie in the tight coupling of operations, which can be tested and debugged before an application is deployed, the advantages of the REST-based approach lie in the potential scalability of a REST-based system and the lightweight access to its operations due to the limited number of operations and the unified address schema.

Table 1: Characteristics of REST and SOAP

	REST	SOAP
Characteristics	Operations are defined in the messages Unique address for every process instance Each object supports the defined (standard) operations Loose coupling of components	Operations are defined as WSDL ports Unique address for every operation Multiple process instances share the same operation Tight coupling of components
Self-declared advantages	Late binding is possible Process instances are created explicitly Client needs no routing information beyond the initial process factory URI Client can have one generic listener interface for notifications	Debugging is possible Complex operations can be hidden behind façade Wrapping existing APIs is straightforward Increased privacy
Possible disadvantages	Large number of objects Managing the URI namespace can become cumbersome	Client needs to know operations and their semantics beforehand Client needs dedicated ports for different types of notification Process instances are created implicitly

The rift between SOAP and REST proponents seems big. FIELDING'S critique of SOAP is representative for many REST advocates [22]:

> *"In order for [the next generation SOAP protocol] to succeed as a Web protocol, it needs to start behaving like it is part of the Web. That means, among other things, that it should stop trying to encapsulate all sorts of actions under an object-specific interface. It needs to limit its object-specific behavior to those situations in which object-specific behavior is actually desirable."*

However, SOAP proponents argue that because technology exists that allows designers to encapsulate the complexity of a system, machine-generated interfaces make system design easier. SPOLSKY summarizes this argument as follows [41]:

> *"The point of SOAP is that you use higher level languages to call it and to implement it […] The claim that SOAP is bad because the wire format is ugly and hard is like claiming nobody should use Pentiums because their instruction sets are so much more complicated than the instruction set*

for the 8086. Yeah, it's true, but we have compilers that take care of that, and make it easier for the many who use compilers at the expense of the few who write compilers."

One of our initial questions coming into the case was whether the position of the REST group, perceived as a minority in the vendor fight, has technical merit. From our analysis, the REST group has many good points – their arguments are strong and well-thought through. So yes, there is merit to their claims. On the other hand, the SOAP group also presents strong technical arguments, as in the quote above. Our conclusion is that the technical debate is a valid one – there are arguments of merit being produced on both sides. This makes the decision about what to do more difficult. In section 4 we take a look at this decision process.

3.4. Testing and Usage Observations

For all the interest in web choreography standards, there are, to the authors' knowledge, no tests in which REST and SOAP versions of a workflow implementation have been compared directly. For the RESTful implementation of a workflow, BAKER published a proof-of-concept document on his website that illustrates, how certain BPEL constructs can be represented using HTTP commands in a REST fashion [7], but it contains no actual implementation of a workflow.

This lack of testing leads back to our original question – is the battle really over the technology, or over something different? If the battle is purely over the technology, then a test might resolve the issue. The absence of testing, and even of a *discussion* about testing, suggests the argument over REST vs. SOAP is not purely technical – something else is at work.

What do the statistics of usage suggest about the popularity or merits of the interfaces? There are few sites where both REST and SOAP interfaces are provided, whether for web services choreography or for other purposes. Amazon.com does provide both, and finds that 85% of developers use the REST interface, whereas only 15% choose the SOAP interface [11, 37]. This was surprising to the authors, given that SOAP has been so heavily promoted in the press. It could be this result is anomalous – or it could be an indicator that REST has more support than is immediately apparent. In order to understand

this result better, we need to first understand who is choosing the standards, and what criteria they might be using.

4. REST vs. SOAP: Comparing and Deciding

In following the commentaries on the development process, it becomes apparent that there are different types of standards participants. These standards participants have different criteria for making decisions. Without considering these different criteria, it is hard to make sense of the chronology. We proceed analytically. We can describe the companies that are making decisions about web services as belonging to one of the following groups:

Table 2: Roles of Web Service Decision-Makers

Decision-Maker Role	Sub role
Users (Web Service Providers and Consumers)	Pure Provider
	Pure Consumer
	Combined Provider/Consumer
Software Vendors	Package Providers
	Tool Providers

4.1. Users

For the possible consumer of a web service, the decision question is which interface to favor. For the possible provider of the service, the question is which interface to provide. In many cases, a company will be both a consumer and a provider of web services – it may consume services from suppliers and provide services to customers. In a situation with fragmented standards, a company may find itself in the cost-heavy position of both consuming and providing according to different standards. For the vendor, the decision question is which technology to base products on. If the standards are uncertain, a vendor may have to build for multiple standards.

We can classify companies seeking to provide web services into two types. Those that already have electronic coordination means through another technology are one type. In general, these are companies who have already picked an industry-related standards track. They are not likely to jump onto either REST or SOAP in the absence of compelling benefits. The other type consists of companies that are not

part of a network, which may decide to bet on one standard now and re-evaluate later. Or they may emulate their peers and form an industry-focused effort. Or they may simply wait until it becomes clearer which set of standards will dominate. This waiting strategy is the one that an options-oriented analysis [8] will probably favor – as the company has a lot to gain by preserving their choices as late as possible.

There is some evidence that the modularity which software exhibits decreases switching costs, and thereby increases hybrid standardization processes [53]. This may mean more switching – it may also mean multiple standards. If the cost to switch is sufficiently small, then lock-in effects may be weak. Consumers of web services have low development and switching costs, so mistaken decisions will not be disastrous. Their decisions may be influenced by the decision of their suppliers, the group just discussed, unless they have a compelling interest of their own.

Those that both supply and consume web services are not served by fragmented standards– they may find themselves communicating with clients and suppliers using different standards. They may care little for arguments of design purity or fit, and simply want one standard, whether it is optimal or not. They are likely to try to predict the winner of the standards battle. If the winner is unclear, they will probably decide to wait. This observation is supported by a recent survey results collected by the Gartner group and Dataquest, which show that 86 percent of surveyed enterprises use XML, but web services standards have found much less adoption. SOAP was used by 31 percent of the respondents, WSDL by only 3 percent [15]. Similarly, BRODIE observed that most initial web services projects target small applications and are limited simple operations (such as data retrieval) from internal applications [14].

The general wisdom is that there will be a point at which it is clear which standard will win. But in work on EDI Diffusion, some of the usual assumptions about the diffusion of technology did not hold. DAMSGAARD and LYYTINEN found that the diffusion of EDI was often dyadic – meaning that two trading partners would decide to do something, in contrast to a major buyer dictating suppliers which solution to choose. They also concluded that often the movement toward EDI represented a herd mentality more than a rational consideration of costs and benefits [19].

Most people seem to expect that web services choreography standards will spread through the work of a few large vendors or customers. But it is possible that the dyadic nature found in the diffusion of EDI may be mirrored by a similar phenomenon in web services choreography.

4.2. Vendors

Having discussed both the detailed state of the standards battles and the likely decision strategies we look at the implications of each of the decision-making groups in turn.

Vendors appear to be investing in these standards. In a market with high externality, this makes sense – the first vendor in has an advantage [36]. By looking at the participants on the different standards committees, one can see that companies are also hedging their bets by working on multiple standards. This also makes sense – consistent with decision theory, they are preserving their options. The potential revenue from an exploding market is great enough to compensate for the costs associated with participating in standards activities.

In the commentary on web services, one sees that many moves are imputed to vendors. In most game theoretic modeling, the vendor either backs or doesn't back a standard. But there are many more available strategies. A vendor can participate in many standards groups, and can profess to be backing a standard, but instead be delaying the process. A vendor may position itself as a standards supporter in order to form strategic alliances with other members of the standards group that lie outside the scope of the standard. A vendor may steer a standards body into merging with another standards body. Or wear down naysayers until those people eventually jump to another standards group. It is hard to determine how many of these strategies are really at work – but all of them have been imputed by other participants in the logs surrounding the creation of the web choreography standards. So even if the moves are not being used, they are being perceived as potential moves in the game. In a game-theoretic analysis, all should be considered.

In fairness to vendors, their motives are not always Machiavellian, and their contributions are sometimes substantial. For example, the creation of the ONC RPC standard is in contrast to the

proliferation web services choreography standards. In 1988 Sun Microsystems wrote an RFC (Request for Comment) for the IETF outlining the standard [45]. After some initial feedback, a revised version of the RFC was published by SUN [46]. Seven years later, Vint Cerf documented the intellectual property agreement between SUN and the Internet Society [16] and related standards documents followed [42-44].

The standardization process is rarely this simple, and this one wasn't either. Another camp, initiated by DEC and the Open Software Foundation, argued for the adoption of a different RPC standard - DCE RPC. And at this instant both DCE and ONC RPCs are in use. Even with the two different competing standards, things are not so bad - there are only two, and they are both well understood (partially through the open arguments that have occurred) by developers. The IETF and more specifically the RFC process served as a forum for negotiating the intellectual property transfer from Sun, and as a forum for a technical debate over ONC vs. DCE RPC.

The behavior of vendor representatives in standardization seems to span a wide range – from actions that look self-interested, to actions that look altruistic. This range of behavior suggests the following questions for future research – are the vendor representatives who make decisions which look altruistic really being altruistic? If so, are they really representing their vendor? The issue extends to other standardization efforts. For example, vendor backing of open source [54, 55] raises a similar issue.

5. Conclusions and Future Work

It is clear there is a piece missing on the web. Human to machine communication works well, but machine to machine communication is cumbersome. Both REST and SOAP provide different ways of solving this problem. Web services choreography becomes the nexus point as its implementation holds the promise of true automated integration and coordination across institutional boundaries. In a recent development, REST principles have been worked into the standards and guidelines associated with the new version of SOAP [32]. This looks like an interesting compromise. SOAP can be used – but used in such a way that it doesn’t violate REST principles. The language of the SOAP 1.2 standards document

actively encourages such application design. The problem will be in the adoption process – for there is nothing that prevents the non-RESTful use of the SOAP standard.

In describing our case, we have noted a series of interesting patterns. Groups spring up, merge, break off and begin again in a different standards venue. It was not what we expected to see, and not the way most standards processes are described. We also saw spirited technical debate, in line with our expectations. The technical argument is a valid one – both sides have a well-developed position. And the debate deserves a wider audience, as the adoption of choreography standards may, depending on one's viewpoint, enable or constrain future computing.

In keeping with our initial questions, it does appear that technical debate is a motivating factor in standards design. It also appears there are other forces at work, which appear to be more cultural. In this work, we have documented – and understood – a particular standards process and debate. We have focused on the technical argument. A future question to study might be why the participants in the standards process act the way they do.

With such high stakes, the standardization process looks Byzantine. Yet the process itself reflects the different decision criteria that vendors, companies, and researchers have in the process. A more open discussion on decision criteria related to this standard might reveal what the battle is based on. And such a discussion might also create the framework for testing. Claims about the relative performance and complexity of the different approaches are certainly verifiable, and it is possible that the certainty each side feels about the merits of their approach might be swayed by evidence.

We also have shown that the standards process involves a social dimension. So the technical arguments may not be the only driving force behind standards participants' behavior. Our decision analysis supports this. At the present time, the decision to adopt one or the other of the standards may be partially based on technical merit. But it is also based on an anticipation of which standard will dominate.

References

[1] W. van der Aalst and A. Kumar, XML Based Schema Definition for Support of Inter-organizational Workflow, Information Systems Research 14, No. 1 (2003), pp. 23-47.

[2] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic and S. Weerawarana, Business Process Execution Language for Web Services, Version 1.1, OASIS, (2003).

[3] A. Arkin, Business Process Modeling Language, Proposed Final Draft, BPMI.org, San Mateo, CA (2002).

[4] A. Arkin, S. Askary, S. Fordin, W. Jekeli, K. Kawaguchi, D. Orchard, S. Pogliani, K. Riemer, S. Struble, P. Takacsi-Nagy, I. Trickovic and S. Zimek, Web Services Choreography Interface (WSCI) 1.0, W3C Note (August 2002).

[5] Association for Cooperative Operations Research & Development, ACORD XML Business Message Specification for P&C Insurance and Surety, Standards Specification Version 1.3.1, ACORD Corp., Pearl River, NY (2003).

[6] D. Austin, A. Babir, E. Peters and S. Ross-Talbot, Web Services Choreography Requirements 1.0, W3C Working Draft (August 2003).

[7] M. Baker. (2004) Hypermedia Workflow. Retrieved February 9th, 2004 from http://www.markbaker.ca/2002/12/HypermediaWorkflow/

[8] C. Y. Baldwin and K. B. Clark, Design rules (MIT Press, Cambridge, MA, 2000).

[9] A. Banerji, C. Bartolini, D. Beringer, V. Chopella, K. Govindarajan, A. Karp, H. Kuno, M. Lemon, G. Pogossiants, S. Sharma and S. Williams, Web Services Conversation Language (WSCL) 1.0, Technical Report Hewlett Packard Company, Palo Alto,CA (2001).

[10] A. Banerji, C. Bartolini, D. Beringer, V. Chopella, K. Govindarajan, A. Karp, H. Kuno, M. Lemon, G. Pogossiants, S. Sharma and S. Williams, Web Services Conversation Language (WSCL) 1.0, W3C Note (March 2002).

[11] J. Barr, Personal Communication. (Hoboken, NJ, 2003).

[12] G. A. Bolcer and G. Kaiser, Leveraging the Web to Manage Workflow, IEEE Internet Computing 3, No. 1 (1999), pp. 2-5.

[13] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn., H. F. Nielsen, S. Thatte and D. Winer, Simple Object Access Protocol (SOAP) 1.1, W3C Note (May 2000).

[14] M. L. Brodie, Illuminating the Dark Side of Web Services. In: Proceedings of the 29th VLDB Conference (Springer Publishers, Berlin, 2003).

[15] M. Cantara, W. Andrews and N. Latimer, Gartner Surveys Show Web Services Are Entering the Mainstream, Report FT-19-9047 Gartner Group, (2003).

[16] V. Cerf, RFC 1790: An Agreement between the Internet Society and Sun Microsystems, Inc. in the Matter of ONC RPC and XDR Protocols. IETF (1995). Retrieved February 9th, 2004 from http://www.faqs.org/rfcs/rfc1790.html.

[17] E. Christensen, F. Curbera, G. Meredith and S. Weerawarana. Web Services Description Language (WSDL) 1.1. W3C Note (March 2001). Retrieved February 9th, 2004 from http://www.w3.org/TR/2001/NOTE-wsdl-20010315.

[18] F. Curbera, Y. Goland, J. Klein, F. Leymann, D. Roller, S. Thatte and S. Weerawarana, Business Process Execution Language for Web Services, Version 1.0, BEA, IBM, Microsoft, (2002). Retrieved February 9th, 2004 from http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/.

[19] J. Damsgaard and K. Lyytinen, Contours of Electronic Data Interchange in Finland: Overcoming technological barriers and collaborating to make it happen, The Journal of Strategic Information Systems 7 (1998), pp. 275-297.

[20] ebXML, ebXML Business Process Specification Schema. Version 1.01. OASIS and UN/CEFACT (August 2002). Retrieved February 9th, 2004 from http://www.ebxml.org/specs/ebBPSS.pdf.

[21] R. T. Fielding, Architectural Styles and the Design of Network-based Software Architectures, Doctoral Dissertation (University of California, Irvine, CA, 2000) 180p.

[22] R. T. Fielding, Response to: draft findings on Unsafe Methods (whenToUseGet-7). W3C WWW mailing list (April 2002). Retrieved February 9th, 2004 from http://lists.w3.org/Archives/Public/www-tag/2002Apr/0181.html.

[23] Health Level Seven Inc., Recommendations for HL7 Messaging Over Component Technologies Version 1.0, Rev. 9, (1998).

[24] Health Level Seven Inc., HL7 Reference Information Model, Standards Specification Version: V 02-01, Health Level 7, (2003).

[25] D. Hollingsworth, The Workflow Reference Model Version 1.1, WFMC-TC-1003, Workflow Management Coalition, Winchester, UK (1995).

[26] D. Hollingsworth, An XML based Architecture for Collaborative Process Management. In: L. Fischer, Ed., Workflow Handbook 2002, (Future Strategies, Lighthouse Point, FL, 2002), pp. 95-116.

[27] R. Khare, Minutes of the second SWAP BoF meeting. IETF (December 1998). Retrieved February 9th, 2004 from http://lists.w3.org/Archives/Public/ietf-swap/1998Dec/0025.html.

[28] A. Knutilla, C. Schlenoff, S. Ray, S. T. Ployak, A. Tate, S. C. Cheah and R. C. Anderson, Process Specification Language: An Analysis of Existing Representations, NISTIT 6160 National Institute of Standards and Technology (NIST), Gaithersburg (MD) (1998).

[29] H. Kuno, M. Lemon, A. Karp and D. Beringer, Conversations + Interfaces = Business Logic, Technical Report HPL-2001-127/20010601, Software Technology Laboratory, HP Laboratories, Palo Alto, CA, (2001).

[30] F. Leymann, Web Services Flow Language (WSFL 1.0), Technical report, IBM Corporation, Stuttgart (2001).

[31] F. Leymann and W. Altenhuber, Managing business processes as an information resource, IBM Systems Journal 33, No. 2 (1994), pp. 326-348.

[32] N. Mitra, SOAP Version 1.2 Part 0: Primer W3C Recommendation 24 June 2003, XML Protocol Working Group (June 2003). Retrieved February 9th, 2004 from http://www.w3.org/TR/soap12-part0/.

[33] M. zur Muehlen, Evaluation of Workflow Management Systems Using Meta Models. In: R. Sprague, Jr., Ed., Proceedings of the 32nd Hawaii International Conference on Systems Sciences (IEEE Publishers, Wailea, HI, 1999).

[34] Object Management Group, Workflow Management Facility Request for Proposals, RFP Document Number cf/97-05-03, OMG, Framingham, MA (May 1997).

[35] Object Management Group, Workflow Management Facility Specification Version 1.2, Document Number bom/00-05-02, OMG, Needham, MA (May 2000).

[36] T. Oliva, Technological Choice Under Conditions of Changing Network Externalities, Journal of High Technology Management Research 5, No. 2 (1994), pp. 279-298.

[37] T. O'Reilly, Rest vs SOAP at Amazon (2003). Retrieved February 9th, 2004 from http://www.oreillynet.com/pub/wlg/3005.

[38] J. Ricker, K. D. Swenson and M. Silverstein, Asynchronous Service Access Protocol Working Draft 01, Standards Specification Organization for the Advancement of Structured Information Standards, (2003).

[39] RosettaNet Consortium, RosettaNet Implementation Framework: Core Specification, Standards Specification Version: V02.00.01, RosettaNet, (2002).

[40] Society for Worldwide Interbank Financial Transfer, The SWIFTStandards Modeling Methodology, SWIFT, (2002).

[41] J. Spolsky, SOAP backlash (2003) Retrieved February 9th, 2004 from http://www.joelonsoftware.com/news/20020425.html.

[42] R. Srinivasan, RFC 1831 RPC: Remote Procedure Call Protocol Specification Version 2. IETF (August 1995). Retrieved February 9th, 2004 from http://www.faqs.org/rfcs/rfc1831.html.

[43] R. Srinivasan. RFC 1832 XDR: External Data Representation Standard. IETF (August 1995). Retrieved February 9th, 2004 from http://www.faqs.org/rfcs/rfc1832.html.

[44] R. Srinivasan. RFC 1833 Binding Protocols for ONC RPC Version 2. IETF (August 1995). Retrieved February 9th, 2004 from http://www.faqs.org/rfcs/rfc1833.html.

[45] SUN Microsystems. RFC 1050 RPC: Remote Procedure Call Protocol specification. IETF (April 1988). Retrieved February 9th, 2004 from http://www.faqs.org/rfcs/rfc1050.html.

[46] SUN Microsystems. RFC 1057 RPC: Remote Procedure Call Protocol specification. IETF (June 1988). Retrieved February 9th, 2004 from http://www.faqs.org/rfcs/rfc1050.html.

[47] K. Swenson, Simple Workflow Access Protocol (SWAP), IETF Internet-Draft, IETF (August 1998). Retrieved February 9th, 2004 from http://xml.coverpages.org/draft-swenson-swap-prot-00.txt

[48] K. Swenson. Response to: SWAP related query, IETF SWAP mailing list (June 2000). Retrieved February 9th, 2004 from http://lists.w3.org/Archives/Public/ietf-swap/2000Jun/0006.html.

[49] K. Swenson and J. Ricker, Asynchronous Web Services Protocol, Draft Specification San Jose, CA, USA (2002).

[50] K. D. Swenson, Personal Communication. (Hoboken, NJ, 2003).

[51] S. Thatte, XLANG - Web Services for Business Process Design, Initial Public Draft, Microsoft Corporation, Redmond, WA (2001). Retrieved February 9th, 2004 from www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm.

[52] R. Weber, Workflow-Interoperabilität über das Internet mit dem Standard Wf-XML, Wirtschaftsinformatik 45, No. 3 (2003), pp. 345-348.

[53] M. Wegberg, Switching costs and the choice of a standard setting process, In: Proceedings of the 3rd IEEE Conference on Standardization and Innovation in Information Technology, (IEEE, Delft, 2001).

[54] J. West, How open is open enough? Melding proprietary and open source platform strategies, Research Policy 32 (2003), pp. 1259-1285.

[55] J. West and J. Dedrick, Open Source Standardization: The Rise of Linux in the Network Era, Knowledge, Technology & Policy 14, No. 2 (2001), pp. 82-112.

[56] WfMC, Workflow Standard - Interoperability Abstract Specification Version 1.0, Document Number WFMC-TC-1012 Workflow Management Coalition, Winchester, UK (1996).

[57] WfMC, Interface 1: Process Definition Interchange Process Model, Document Number WfMC-TC-1016-P Version 1.1 Final Workflow Management Coalition, Winchester, UK (1999).

[58] WfMC, Workflow Standard - Interoperability, Wf-XML Binding, Version 1.1, Document Number WFMC-TC-1023 Workflow Management Coalition, Lighthouse Point, FL (2001).

[59] WfMC, Workflow Process Definition Interface - XML Process Definition Language. Document Status - 1.0 Final Draft., Document Number WFMC-TC-1025 Workflow Management Coalition, Lighthouse Point, FL (2002).

[60] R. K. Yin, Case Study Research : Design and Methods, 3rd Ed., (Sage Publications, Thousand Oaks, CA, 2003) 181p.

Biographical Notes

Michael zur Muehlen is Assistant Professor of Information Systems at Stevens Institute of Technology in Hoboken, NJ. Prior to joining Stevens, Michael was a senior lecturer at the Department of Information Systems, University of Muenster, Germany, and a visiting lecturer at the University of Tartu, Estonia. He has conducted various workflow and process reengineering projects in the utility, financial services, industrial, and telecommunications sector both in Germany and the US. His research interests include the organizational impact of process automation, workflow-based monitoring and controlling solutions, as well as process-oriented information system architectures. Michael is member of the editorial board of the International Journal on Business Process Integration and Management. He is member of the Technical Committee and served as the country chairman of the Workflow Management Coalition. He is a founding director of the AIS special interest group on process automation and management (SIGPAM). Michael holds a Ph.D. and a Master's degree in Information Systems from the University of Muenster, Germany.

Jeffrey V. Nickerson is Associate Professor and Director of Electronic Commerce at Stevens Institute of Technology, Hoboken, NJ. Prior to his appointment at Stevens Institute of Technology, Dr. Nickerson was a partner at PricewaterhouseCoopers, where he managed a group focused on advanced technologies, and performed consulting related to software design with many companies and government entities. Prior to PwC, Dr. Nickerson worked on Wall Street – he was an Associate Director at Bear Stearns, and a Vice-President at Salomon Inc. He was responsible for designing and building some of the earliest complex distributed systems, in the domain of program trading and mortgage allocation. Early in his career he worked as a member of the programming staff at AT&T Information Systems Laboratories. Mr. Nickerson is on the executive board of the Association of Information Systems' Special Interest Group in Process Automation and Modeling. Jeffrey Nickerson holds a Ph.D. in Computer Science from New York University and a B.A. from the University of California, Berkeley.

Keith D. Swenson is Chief Architect and Director of Development at Fujitsu Software Corporation in San Jose, CA. He began his tenure at Fujitsu in 1991 where he developed TeamWARE Flow. He returned to Fujitsu Software Corporation in 2002 to direct the development of the Interstage family of products. A pioneer in web services, Mr. Swenson has helped the development of standards such as the WfMC Interface 2, the OMG Workflow Facility, SWAP, Wf-XML, AWSP and WSCI. He is currently working on standards such as WS-CAF and ASAP. Prior to joining Fujitsu Software Corporation, Mr. Swenson was Director of Engineering at MS2, Inc., where he was responsible for developing Accelerate, a distributed enterprise scaled system. Before MS2, Mr. Swenson was a software architect at Netscape, responsible for developing software tools and a new workflow engine. Mr. Swenson has also held senior positions at Ashton-Tate and Software Products International. Mr. Swenson holds both a Master's degree in Computer Science and a Bachelor's degree in Physics from the University of California, San Diego.