ISBN: 972-8924-10-0 © 2006 IADIS

PEER-TO-PEER BUSINESS COMMUNITIES: MAPPING PUBLIC CHOREOGRAPHY PROTOCOLS TO INDIVIDUAL IMPLEMENTATION MECHANISMS

Adomas Svirskas
Vilnius University
Vilnius LT-03225, Lithuania

Bob Roberts
Kingston University
Kingston upon Thames, KT12EE, United Kingdom

Michael Wilson
CCLRC Rutherford Appleton Laboratory
Chilton, Didcot, Oxfordshire, OX11 0QX, United Kingdom

ABSTRACT

This paper explores certain issues of business-oriented Web Based Communities also referred to as Virtual Organisations or Virtual Enterprises. One of the important B2B and Virtual Enterprise (VE) implementation aspects is support of commonly agreed collaboration scenarios between the peers. In order to for the scenarios be understood and accepted by the members of a VE, the former need to be specified in a standard, machine-readable form and mapped to the implementation mechanisms of the VE participants. Such specifications, which describe the sequence and the format of message exchange between the collaboration participants, are known as choreographies. We discuss the challenges of constructing architecture for B2B interactions based on choreographies and Service Oriented Architecture (SOA), Web Services in particular. One of the WS-* specifications, which attempts to address business process integration issues is the Web Services Choreography Definition Language (WS-CDL). In this paper we present an approach to harmonisation of the "global" or neutral definition of business collaborations, which WS-CDL is used for, with partner-specific implementations, which can differ in terms of platform, environment, implementation technology, etc. By introducing the concept of pluggable business service handlers we draw on work carried out by the ebXML initiative, business services interfaces, in particular. We analyse a possibility to use flexible business service handlers, which would smoothen the transition between public and private business processes by using flexible mapping techniques.

KEYWORDS

Choreography, Service Oriented Architecture, B2B, Web Services, ebXML

1. INTRODUCTION

Modern architectures and implementations of Virtual Enterprises are increasingly based on the concept of Service-Oriented Architectures (SOA) (He 2003). SOA is a framework, which represents individual business functions as reusable services in order to implement complex business applications and processes efficiently. Furthermore, SOA is an approach to IT that considers business processes as reusable components or services which are loosely-coupled and that are platform and implementation neutral (Weaver 2005). Firstly, we would like to emphasize the need of choreography support for SOA-based e-business solutions and Web Services in particular. We also provide a brief overview of major efforts in this area before presenting a possible solution of choreography usage in Web Services based B2B collaborations.

The SOA approach allows to design solutions as assemblies of services in which the assembly description is a managed, well-defined first-class aspect of the solution, and hence, amenable to analysis, change, and evolution. The solution can then be viewed as a choreographed set of service interactions (Weaver 2005). We see the aspect of change and evolution as the key to successful adoption of choreography-based collaboration specification.

Recent developments in Web Services field provide promising opportunities for integrating data, applications and business processes. The latter, however, is the most complex case of integration as it requires strong support for both business process semantics and technical infrastructure in order to tackle heterogeneity at all levels. According to Wombacher et al. (2004), in today's B2B solutions landscape loosely coupled business processes are quite rare. Despite many promising advancements, Web Services technology faces a number of issue to address heterogeneity at different levels of integration. Usage of simple stateless Web Services is not sufficient for implementing business processes while static binding does not use full potential of loosely coupled systems and the SOA advantages (Wombacher 2004).

The rest of this paper is structured as follows. Section 2 briefly discusses the motivation for choreography in B2B collaborations and differentiates between the choreography and orchestration. Section 3 introduces the Web Services Choreography Description Language and some other choreography alternatives; Section 4 describes the main concepts of our approach and relates it to the ebXML (Enabling Electronic Business with ebXML 2000) framework followed by Section 5, which concludes the paper.

2. MOTIVATION

Vinoski (2001) argues that Web Services choreographies must take business processes into account: trivial Web Services solve only trivial issues; non-trivial web services must play a part in business processes. Business-to-business integration (B2Bi) requires standardized choreographies, i.e. definitions of the "conversations" between cooperating applications that allow them to work together correctly (Vinoski 2001). Simply put, choreography is a model of the sequence of operations, states, and conditions that control the interactions involved in the participating services (Web Services Architecture 2004). The interaction prescribed by a choreography results in the completion of some useful function. Examples include the placement of an order, information about its delivery and eventual payment, or putting the system into a well-defined error state. Gortmaker et al. (2004) have presented an extensive of choreography definitions along with a thorough discussion of choreography and orchestration.

For the sake of clarity we would like to note the difference between choreography and orchestration, as these two terms are being used improperly sometimes, which causes confusion. **Figure 1** depicts inter-relation between the choreography and orchestration in business collaboration context.

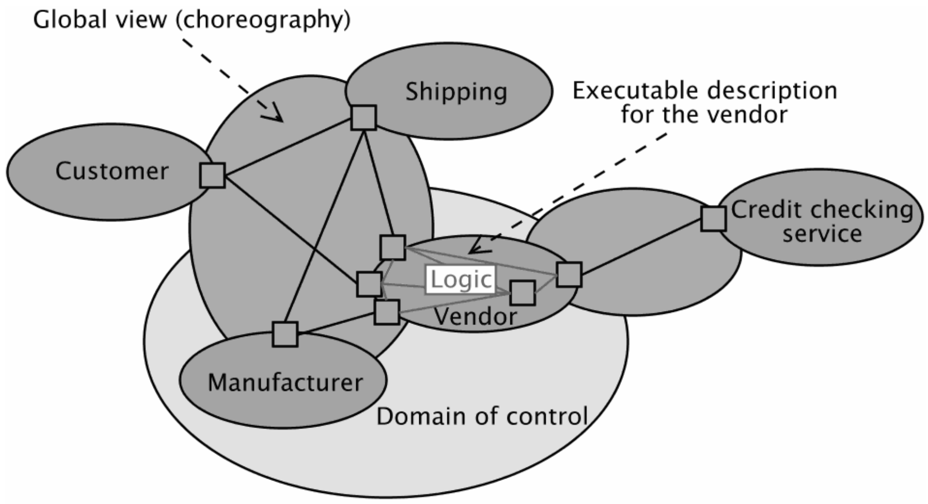

Figure 1. Web Service Choreography vs. Orchestration or executable description (Booth 2005)

Orchestration specifies the behaviour of a participant in a choreography by defining a set of "active" rules that are executed to infer what to do next, once the rule is computed, the orchestration runtime executes the corresponding activity(ies). Orchestration assumes existence of an entity, which is the central point of control and governs overall workflow of activities, effectively composing a new service from existing services. The standardization of orchestration and the emergence of a new application model will also benefit from a robust B2B layer, such as ebXML, in the Web services stack. As a matter of fact, orchestration could take its full dimension from the extension of the business semantics to the application model (Dubray 2004).

Choreography, as explained before, is meant to be enacted by peers without an intermediary, at runtime, the choreography definition can be used to verify that everything is proceeding according to plan. Choreography can also be used to generate a public interface, e.g. abstract BPEL (WS-BPEL 2005) that can be used to tie in internal activities to support the choreography. Dubray (2004) also differentiates between the two concepts by arguing that choreography defines the fabric of an SOA while orchestration, helps to build "processing entities" – non-trivial services, which can perform tasks, needed to support complex business interactions. Ross-Talbot (2005) defines choreography as a description of the *peer to peer externally observable interactions* that exist between services. The interactions are described from a global or neutral point of view and not from any one services perspective. Choreography can be used to generate the necessary behavioural contract for each of the peers; further non-observable logic (code) maybe required to manifest the full set of service implementations (Ross-Talbot 2005).

3. CHOREOGRAPHY

The considerations mentioned above and some earlier developments, such as Web Service Choreography Interface (WSCI 2002), ebXML Business Process Specification Schema (ebBP 2006) served as input for W3C to establish the Web Services Choreography Working Group and begin work towards a language that can be used to describe collaboration protocols of cooperating participants, which act as peers and their interactions may be long-lived and stateful. Web Services Choreography Requirements document (WS-CHOR 2004), provides the following definition of choreography: *"Web Services Choreography concerns the observable interactions of services with their users. Any user of a Web Service, automated or otherwise, is a client of that service. These users may, in turn, be other Web Services, applications or human beings. A specific set of interactions maybe related over time to some form of collaboration grouping that is initiated at some source and runs through a set of Web Services and their client. A choreography description is a multi-party contract that describes from global view point the external observable behaviour across multiple clients (which are generally Web Services but not exclusively so) in which external observable behaviour is defined as the presence or absence of messages that are exchanged between a Web Service and it's clients."* (WS-CHOR 2004).

Goland (2003) contributed notably to these requirements by advocating the need for complementary but separate languages – choreography programming languages and choreography description languages. Goland (2003) showed rather clearly, using motivating use cases, the difference (from choreography point of view) between executable languages such as Java, C#, BPEL and similar, and declarative description languages, which capture a global view of messaging activity and are not designed to provide information about how participants implement their individual tasks. Goland (2004) also explained the need for generating role-specific code skeletons from choreography description in order to facilitate faster and more convenient implementation of individual functionality. The choreography description language uses roles to differentiate between the participants in choreographies. We will discuss this aspect in greater level of detail in subsequent sections.

The result of the mentioned W3C WS Choreography Working Group effort is WS-CDL language (WS-CDL 2005), which is the means to define a technical multi-party contract, mentioned above. WS-CDL specification is aimed at being able to precisely describe collaborations between any types of participants regardless of the supporting platform or programming model used by the implementation of the hosting environment, thus addressing heterogeneity issues (WS-CDL 2005). Choreographies must also completely hide component-level implementation details. Moreover, the same choreography definition (potentially involving any number of parties or processes) needs to be usable by different parties operating in different contexts (industry, locale, etc.) with different software (e.g. application software) (WS-CDL 2005).

Choreography definition using WS-CDL allows building of more robust services because they can be validated statically and at runtime against a choreography description, verification absence of deadlocks and live-locks, etc. It also helps to ensure effective interoperability of services, which is guaranteed because services will have to conform to a common behavioural multi-party contract, mentioned earlier (Ross-Talbot & Brown 2005).

However, compliance of the participating services to the common contract might result that the choreography enactment is hard coded into the implementations of the services and/or their composition mechanisms. This approach poses a two-fold problem: it reduces reusability of the services and also makes it difficult to change choreography description without a need for massive programmatic changes at the participating end-points. Goland (2003) discusses these issues from developers' point of view in his contribution to WS Choreography requirements.

A possible alternative to global-contract choreography is a technique called mediation, where an intermediary agent is involved in communication between parties and ensures compliance of message flow to expected/requested behaviour of each party (Cimpian & Mocan 2005). A notable example of such approach is Web Service Modelling Ontology (WSMO) Choreography (Roman et al. 2005). WSMO is an ontology for describing various aspects related to Semantic Web services (Feier 2005).

The WSMO framework provides support for choreography and orchestration as part of interface definition of a WSMO service description. An interface describes how the functionality of the service can be achieved (i.e. how the capability of a service can be fulfilled) by providing a twofold view on the operational competence of the service: Choreography decomposes a capability in terms of interaction with the service (service user's view) Orchestration decomposes a capability in terms of functionality required from other services (other service providers' view) With this distinction different decompositions of process/capabilities are provided to the top (service requester) and to the bottom (other service providers). This distinction reflects the difference between communication and cooperation. The choreography defines how to communicate with the web service in order to consume its functionality. The orchestration defines how the overall functionality is achieved by the cooperation of more elementary service providers (Roman & Lausen 2004). WSMO choreographies are based on the Abstract State Machines methodology. Cimpian & Mocan (2004) in their work describe a process mediation approach based on WSMO choreographies and Web Service Execution Environment (WSMX) (Cimpian et al. 2005), a reference implementation for WSMO.

4. SUGGESTED APPROACH

In this paper we propose software components called handlers to represent the logic of harmonizing public choreographed processes with private functionality of end-point services, as shown in **Figure 2**. Handlers are registered with and coordinated by a choreography support service, which, in turn, used by the end-point services to support global choreography contracts.

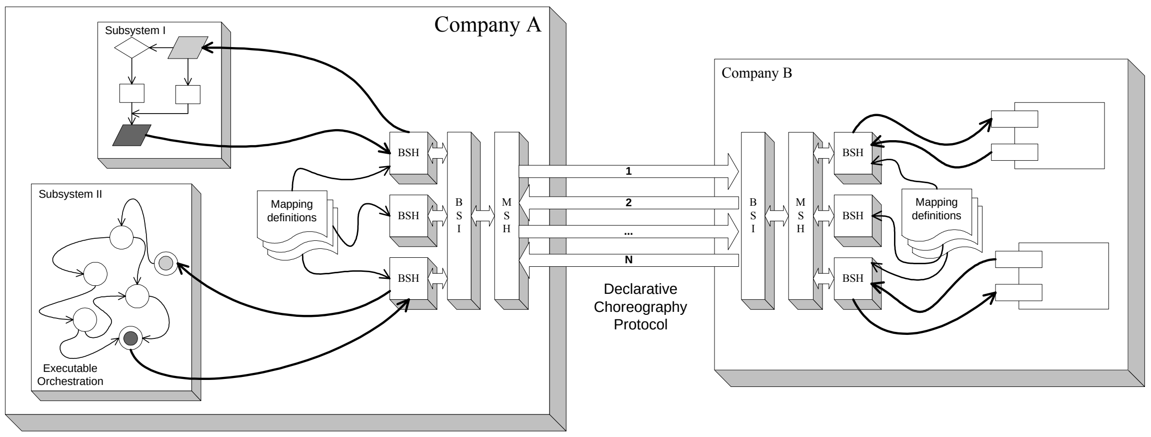

Figure 2. Pluggable handlers map choreography to local mechanisms at run-time

Configured with pluggable handlers choreography support service mediates two-way message exchange between the “outside world” and the local processing entities. Based on the available handlers, various request types and formats can be routed, translated, and fulfilled by the business services. Choreography support service can relatively easily be reconfigured, adapted, and extended as new processing entities need to be supported. In addition, dynamic selection of processing entities to play the prescribed roles, policy enforcement, trust and security support and other non-functional tasks can be performed by the handlers. The handlers can be implemented as a chain of message filter put in front of processing entities deployments. Handlers take a choreography definition, the role, which processing entity is supposed to play and maps the choreography messages to local operations at run-time.

4.1 The role and status of the ebXML

We call the handlers business service handlers, drawing a parallel with the naming used in ebXML framework (ebBP 2006). The original name for these components in ebXML framework was Business Service Interface (BSI), which can be described as a piece of software that handles incoming and outgoing messages at either end of the transport (Enabling Electronic Business with ebXML 2000). The ebXML concept of a business transaction and the semantics behind it are central to predictable, enforceable e-business. It is expected that any Business Service Interface (BSI) will be capable of managing a transaction according to these semantics. Dubray (2003) explains the purpose of ebXML BSI in his overview of ebXML : "The Business Service Interface (BSI) should enforce the business collaboration protocol (ebXML BPSS). At any point in time, the BSI is able to determine if a message makes sense from a business perspective (is format correct? did it come on time? in the right sequence? ...). The BSI may be directly communicating with an application, but it is certainly wise to use a broker that will dispatch ebXML requests and responses to and from your business applications. Typically, this broker is going to be a business process management system. ". This explanation actually outlines the core functionality of BSI and justifies the need of pluggable brokers to support a variety of business process management systems. The OASIS ebXML Business Process (ebBP) Technical Committee (TC) later discussed a possibility to differentiate between Business Services Interface and Business Service Handler (BSH) in order to separate the abstract interface from its implementations. By proposing the name change to business service handler, the ebBP committee harmonised the naming between the business and messaging domains - the ebXML Message Service Specification (ebMS 2002) defines the Message Service Interface and Handler separately.

In ebBP Specification v 2.0.1 (ebBP 2006) the BSI is defined from a different perspective: as a logical definition for a party's actions, exposed as business services. It may be seen as a logical shared definition at different nodes. Logically, a BSI is a partner's implementation of the shared definition of business states and actions relevant to a common business goal. The BSI specifies the allowed set of business process and business object states of a business process, and the rules governing transitions between those states. In the context of the ebBP technical specification, only the shared business process is being managed. The interface to the BSI is through business messages and signals (ebBP 2006). This defines the functionality of the BSI closely to the functionality of an individual partner required to support WS-CDL based common multi-party contract. Therefore the ebXML BPSS and W3C WS-CDL define substantially similar approach to enactment of common business goal and idea of pluggable business handlers follows this paradigm.

The ebBP technical specification does not, however, specify how the BSI is implemented. For example, the BSI may be enabled through a BSI-aware business application or through behaviour implemented as a part of a Message Service Interface component. The business application may business signals that are sent (realized) by the Message Service Handler (ebBP 2006). Similarly, WS-CDL (2005) does not specify how collaborating parties implement/map their services to comply with the common contract. We think that it useful to turn to the ebBP TC work when architecting choreographed Web Services solutions, as the ebBP v2.0 specification takes Web Services into account and explicitly relies on choreographed collaborations (no relation to WS-CDL is defined yet; the ebBP TC, however, is working on ebBP and WS-CDL layering). An ebBP Choreography is an ordering of Business Activities within a Business Collaboration and is specified in terms of Business States and transitions between those Business States. Execution of the backend systems, which instruct the BSI to send or receive messages, advances the state of a collaboration. Similarly to WS-CDL, there is no execution engine associated to the collaboration itself. Although WS-CDL and ebBP address similar problem domains, the divergent foci of the two enables them to be layerable - while WS-

CDL focuses primarily on the web service perspective, ebBP describes the pure business message flow and state alignment. As such they are not mutually exclusive.

4.2 Operation Mapping

One more notable aspect of ebBP v2.0 specification is mapping of Business Transaction patterns to abstract operations through the OperationMapping constructs (still work in progress) (ebBP 2006). An operation mapping specifies a possible mapping of a business transaction activity to a set of Web Services operation invocations to enable the participation of non-ebXML capable business partner in an ebXML relationship. An ebBP definition does not itself contain a reference to a WSDL file, but rather references to operation names which can be deferenced with specific WSDL files specified at the Collaboration Protocol Profile (ebBP 2006).

The goal of the operation mapping is to offer a flexible mapping scheme to map all document and signal interchanges to any combination of Web Services operation interactions. The mapping is also designed to define an operation mapping on both sides of a Business Transaction Activity (BTA). BTA represents the performance of a Business Transaction within a collaboration and is similar to WS-CDL interaction. This means that the ebBP specification can be used to define the abstract behaviour of complex collaborations between Web Services even in the case where no role in the collaboration is capable of ebXML (ebBP 2006).

Barreto (2005) argues that WS-CDL and ebBP could be used in a loosely coupled, yet complementary manner, where WS-CDL supports the choreography based on endpoint references related to WSDL, while ebBP specifies the operation mapping to the recognized business transaction patterns. This association is beneficial and useful where complex activities occur in the collaboration environment.

If one or more parties wish to participate on the basis of one or more Web Service definitions the corresponding WSDL file(s) associated to the BTA(s) that is(are) representing the ebXML compliant party may be generated and may be referenced in the Collaboration Protocol Agreement (CPA) (Sachs 2001) if necessary. Guided by the Collaboration Protocol Profiles (CPP) (Sachs 2001) and CPA specifications the resulting XML document then may become the configuration file for one or more Business Service Interfaces (BSI), i.e. the software that may manage either partner's participation in the collaboration (ebBP 2006). This facility may be used not only in conjunction with the ebXML CPA but also with other configuration capabilities to enable the use hybrid ebXML and web services in business message exchange. We think that this principle of generated configuration artefact is also suitable to configure run-time operation our proposed business service handlers.

The concepts found in ebBP v2.0 specification, therefore, provide several benefits for mapping choreography contracts to the end-point implementations:

- There are clear signs and concrete steps to support Web Services at WSDL operations level. The use of run-time correlation and endpoint references based on emerging addressing mechanisms such as WS-Addressing, WS-MessageDelivery etc. is recommended.
- The ebBP operation mapping is designed to support not only Web Services but other implementation techniques as well.
- The ebBP v2.0 specification suggests flexible approach to operation mapping (and common contract enforcement, in turn) by allowing operations to be mapped both in collaboration description and the partner's endpoints.

Where the ebBP schema is used but the OperationMapping is not explicitly defined, the partners should manage the service mappings. Through a business service, the OperationMapping MAY also support Business Transactions defined in other than XML where different identification mechanisms are used. This allows the binding of service and business endpoints (ebBP 2006).

Therefore, ebBP (2006) serves as a blueprint for architecting choreography-based solutions in general, not only ebXML compliant systems. The ebBP specification is being created (or closely watched) partially by the same individuals as WS-CDL specification, which creates synergy between the two efforts. We took into account many ideas described in ebBP specification and this will greatly help in our future work on this topic.

5. CONCLUSIONS

In this paper we have introduced an approach to the issue of choreography support in heterogeneous peer-to-peer business interactions. We base our concepts on the assumption that service choreographies can be mapped to end-point operations using either rich service universal descriptions or end-point specific operation mappings. In both cases it is possible to derive programmatically the configuration artefacts, which can be used to configure business service handlers dynamically at the runtime. This possibility is attractive from many points of view, the most important perhaps being clean separation of business services interfaces and business services implementation.

Pluggable business service handlers can also be used for various other purposes – policy enforcement, trust and security support, collaboration correctness monitoring, QoS monitoring, transaction logging, etc. Choreography languages, such as WS-CDL can perhaps be enhanced to support declarative specification of the mentioned aspects for subsequent programmatic propagation of these specifications to the service end-points and mapping to the end-point specific mechanisms.

These ideas make basis for future research in this area alongside with detailed design of the pluggable business handler framework, which is described in this paper at conceptual level and many decisions still need to be made. Standard operation mappings between choreographies and implementation languages such as Java, C#, BPEL are one of the main issues in this area along with rich service description and matchmaking problems. It is a promising sign that the ebXML BPPS specification takes into account these issues and drives the effort to solve them in standard interoperable manner, as support of the open standards is crucial for adoption of solutions.

REFERENCES

Barreto, C 2005, WS Choreography WG conference call 15 March 2005. Minutes. http://www.w3.org/2002/ws/chor/5/03/15-minutes.html

Booth, D 2005, From Web Services to the Semantic Web: Global Data Reuse. 2005. http://www.w3.org/2005/Talks/0110-dbooth-semweb

Cimpian, E & Mocan, A 2005, WSMX Process Mediation Based on Choreographies. 1st International Workshop on Web Service Choreography and Orchestration for Business Process Management (BPM 2005), Nancy, France

Cimpian, E, Moran, M, Oren, E, Vitvar T, Zaremba, M 2005, Overview and Scope of WSMX. Technical report, WSMX Working Draft. http://www.wsmo.org/TR/d13/d13.0/v0.2

Dubray, JJ 2003, ebXML. http://www.ebpml.org/ebxml.htm

Dubray, JJ 2004, WS-CDL - Choreography Description Language. http://www.ebpml.org/ws_-_cdl.htm

ebBP, ebXML Business Process Specification Schema Technical Specification v2.0.1. Public Review Final Draft r03, 2 January 2006. http://www.oasis-open.org/committees/download.php/16060/ebxmlbp-v2.0.1-Spec-pr-r03-en-pdf.zip

ebMS, ebXML Message Service Specification 2002, Version 2.0 rev C, http://www.oasis-open.org/committees/ebxml-msg/documents/ebMS_v2_0rev_c.pdf

Enabling Electronic Business with ebXML 2000, OASIS Whitepaper, http://www.ebxml.org/white_papers/whitepaper.htm

Feier, C (ed.) 2005, WSMO Primer. WSMO Final Draft 01, http://www.wsmo.org/TR/d3/d3.1/v0.1/

Goland, YY 2003, A proposal for W3C Choreography Working Group Use Cases & Requirements, http://lists.w3.org/Archives/Public/www-archive/2003May/att-0029/chor.htm

Gortmaker, J & Janssen, M 2004, & Wagenaar, R. SOBI Business Architectures and Process Orchestration: Technical Overview. Telematica Instituut, Enschede, The Netherlands

He, H 2003, What is Service-Oriented Architecture? http://webservices.XML.com/lpt/a/WS/2003/09/30/SOA.html

Roman, D &Lausen, H (eds.) 2004, Web Service Modeling Ontology – Standard (WSMO Standard). WSMO Working, http://www.wsmo.org/2004/d2/v03/

Roman, D, Scicluna, J, Feier, C (eds.) 2005, Ontology-based Choreography and Orchestration of WSMO Services. WSMO Final Draft, http://www.wsmo.org/TR/d14/v0.1/

Ross-Talbot, S & Brown, G 2005, Dancing in time with the services. WS-CDL., NY Java SIG, NYC, USA. http://www.javasig.com/Archive/lectures/JavaSIG-CDL-SRT.ppt

Ross-Talbot, S 2005, Orchestration and Choreography: Standards, Tools and Technologies for Distributed Workflows. NETTAB Workshop - Workflows management: new abilities for the biological information overflow. Naples, Italy. http://www.bioinformatics.org/NETTAB/2005/docs/NETTAB2005_Ross-TalbotOral.pdf

Sachs, M 2001, ebXML Collaboration Protocol Profile and Agreement specification, ver. 1.0, http://www.oasis-open.org/committees/ebxml-cppa/meetings/ebxml-cpp-cpa.pdf

Vinoski, S 2001, The Truth about Web Services. Web Services and Component Technologies, http://www.proinfo.com.cn/chinese/menu2/3/102501.ppt

Weaver, R 2005, The Business Value of the Service Component Architecture (SCA) and Service Data Objects (SDO). 2005, http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-sca/SOAProgrammingModelBusinessValue.pdf

Web Services Architecture 2004, W3C Working Group Note, http://www.w3.org/TR/ws-arch/

Wombacher, A et al. 2004, Matchmaking for Business Processes Based on Choreographies, IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'04) pp. 359-368

WS-BPEL 2004, OASIS Web Services Business Process Execution Language, http://www.oasis-open.org/committees/wsbpel/charter.php

WS-CDL 2005, Web Services Choreography Description Language Version 1.0. W3C Candidate Recommendation, http://www.w3.org/TR/2005/CR-ws-cdl-10-20051109/

WS-CHOR 2004, Web Services Choreography Requirements. W3C Working Draft, http://www.w3.org/TR/ws-chor-reqs/

WSCI 2002, Web Service Choreography Interface. W3C Note, 2002. http://www.w3.org/TR/wsci/