Insights into Web Service Orchestration and Choreography

Florian Daniel, Politecnico di Milano, Italy

Barbara Pernici, Politecnico di Milano, Italy

ABSTRACT

As the Web service domain is a fast growing and equally fast changing environment, this paper tries to provide a critical snapshot of currently available standards, particularly focusing on Web service orchestration and choreography. The trend over the last few years in the Web services area firmly points towards seamless business logic integration and inter-enterprise collaboration. In order to reach these business goals, both technological and conceptual advances are required; some already have proven their viability, others still have to be made. Among them, Web service orchestration and choreography are of crucial importance, but still lack a widely agreed on development framework comprising both technological and conceptual aspects. Besides discussing problems and solutions regarding orchestration and choreography of Web services, especially from a conceptual point of view, this paper further tries to highlight mutual dependencies existing among orchestration and choreography.

Keywords: business process management; choreography; orchestration; service composition; service coordination; survey; Web services

INTRODUCTION

When analyzing the current literature on Web services and the main problems the authors focus on, it is possible to identify one main trend towards the adoption of novel and emerging Web service technologies as basis for the next generation of (Web) applications and composite Web services. Flexibly composing different services into composite ones that benefit from the functionalities provided by their single component services, and expose them as higher-level composite services by combining them in a value adding manner, becomes thus of crucial importance.

Web services are driven by the paradigm of the so-called *Service-Oriented Architecture* (SOA), which describes the relationships that exist among service providers, consumers, and service brokers, and thereby provides an abstract execution environment for Web services. Accordingly, the overall current research addressing service composition is based on technologies and solutions from the area of *Service-Oriented Computing* (SOC). From their first appearance, SOA and SOC have emerged as key conceptual frameworks for the world of Web services. Interestingly, only few authors (mainly from the industrial sector) mentioned the concept of *Service Oriented Programming* (SOP) (Bieber & Carpenter, 2001). Web service choreography and particularly orchestration actually face the problem of *programming*, rather than the one of *computing*, which is a somewhat abstract concept not easily mappable to the concept of *service*. Within the academic area, maybe Wiederhold, Wegner, and Ceri (1992) already envisioned a SOP-like paradigm when speaking about *Megaprogramming* of large software modules encapsulating business logic at a granularity level comparable to that of today's Web services, but this was in the early 1990s! Obviously, cutting down the whole research on service composition and related issues to the mere concept of programming would be to simplistic, and we definitely do not intend to narrow it down to such a low level of abstraction. Nevertheless, we think a rough comparison of the two concepts represents a challenging intellectual exercise and allows drawing interesting conclusions.

Just as the advent of *Object-Oriented Programming* (OOP) was based on the notion of *objects* as means for modularizing programming functionality, SOP could be defined as a paradigm that looks at *services* as basic functional modules that can be composed or newly defined, just as it happens with objects in object-oriented programming languages. OOP per se did not suddenly provide revolutionary new programming capabilities with respect to conventional procedural techniques, it rather proved to be a good means for isolation and thus fostered reuse, robustness, and scalability. These factors encouraged the emergence of higher-level concepts like object brokers, Java Beans, object containers, which — and actually it is this what we are interested in — finally enhanced interoperability.

Analogously, current proposals can be interpreted as a transition towards a robust SOP framework. Several Web service standardization bodies are currently addressing issues that can be related to the definition of a proper new programming framework. For example, even if we are already speaking about service composition and seamless inter-enterprise integration, there is still discussion over standardization of other system aspects (e.g., reliable messaging, transaction support…) that have already been solved or are under study in other research areas. And as long as there are no robust and commonly agreed upon standards, real interoperation and composition problems cannot be addressed adequately.

HANDLING THE COMPOSITION TOOLKIT

The Mess with the Right Terminology

As standards and technologies still have to reach stable definitions, also authors writing about service composition are far from using a commonly agreed on terminology. Peltz (2003) defines *orchestration* as executable business process that interacts with both internal and external Web services, and *choreography* "…tracks the message sequences among multiple parties and sources — typically the public message exchanges that occur between Web services — rather than a specific business process that a single party executes…"

Alonso, Casati, Kuno, and Machiraju (2004) prefer the terms coordination (protocol) and composition rather than choreography and orchestration. Literally, they clarify "…we will use the term *conversation* to refer to the sequences of operations (i.e., message exchanges) that could occur between a client and a service as part of the invocation of a Web service. We will use the term *coordination protocol* to refer to the specification of the set of correct and accepted conversations…" And "…we refer to a service implemented by combining the functionality provided by other Web services as a *composite service*, and the

Figure 1. A contextualized view on currently used terminology; the two main nomenclatures concerning (respectively) public and private perspective on Web services can further be specialized by actor and execution time

		Perspective	
		public	private
Actor	Composition Designer (design time)	Coordination	Composition
	Execution Engine (runtime)	Choreography	Orchestration

process of developing a composite Web service as *service composition…*"

The W3C's Web Services Choreography Working Group defines *choreography* as the definition of the sequences and conditions under which multiple cooperating independent agents exchange messages in order to perform a task to achieve a goal state. Web services choreography concerns the 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. An *orchestration* defines the sequence and conditions in which one Web service invokes other Web services in order to realize some useful function, that is, an orchestration is the pattern of interactions that a Web service agent must follow in order to achieve its goal (W3C, n.d.).

As this terminological comparison outlines, different authors prefer different names and thus emphasize different aspects even within the same Web service domain. Figure 1 attempts to characterize and aggregate the currently used terminology through contextualizing the most commonly used terms. For this purpose, it distinguishes two main dimensions: the perspective of the observer and the kind of observer along with its observation time. According to a common approach, the perspective is divided into *public* and *private*, with respect to the observer's view, whereas the novel aspect of Figure 1 is represented by the dimension *actor*, which allows distinguishing between composition designers and execution engines. An *execution engine* executes a composite service (runtime orchestration: the engine is already provided with the set of component services, the *orchestra*) that has previously been defined by a composite service designer (design time composition: the orchestra is *composed* by selecting the right services). A *service designer* thus composes a new service driven by a final goal and by taking into account the restrictions imposed by the coordination protocols of the component services, and by specifying the composition rules for the selected services and the coordination rules, which constrain possible interactions with the services. At runtime, externally visible coordination effects can be interpreted as choreography with respect to the orchestra of compound services.

The taxonomy of Figure 1 should provide the reader with a coarse contextualization of the most used terms and serves merely orientation purposes. Therefore, it should not be considered a widely acknowledged categorization.

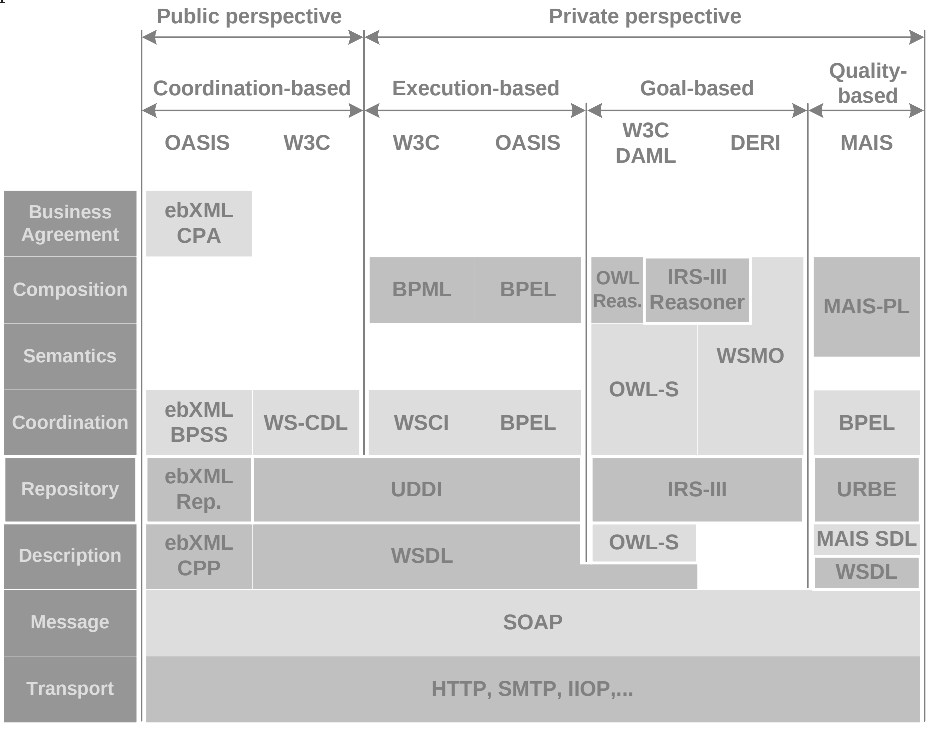
The Mess with the Right Standards

After this interpretation of the most commonly used nomenclature conventions, another similar concern arises: Why are there so many different standards and specifications that want to become such?

A Possible Protocol Stack

Figure 2 shows a possible Web service protocol stack that concentrates on service

Figure 2. Web service composition-oriented protocol stack of vendor-specific and standardized protocols and languages. Within the composition layer, we propose BPML in on top of WSCI as they share a common process model. However, other executable BPM languages could be adopted as well.



coordination and composition. Interaction among services is based on traditional transport protocols such as HTTP, SMTP, or IIOP, and the widely acknowledged basic message protocol is SOAP (nevertheless, other protocols could be used). Web service description is primarily achieved by means of WSDL, but when it comes to service coordination and composition, a wide range of different protocols and languages are proposed by different vendors or organizations:

- ebXML (*Electronic Business using eXtensible Markup Language*); UN/CEFACT, OASIS (Eisenberg & Nickull, 2001). ebXML is a (vertical) suite of specifications of how electronic commerce exchanges should be specified, documented, and conducted, and can be subdivided into three different protocols:

- CPP (*Collaboration Protocol Profile*). A CPP is similar to a UDDI registry entry and includes interface and message descriptions as well as business data and data exchange capabilities of a particular trading partner.
- BPSS (*Business Process Specification Schema*). The BPSS protocol can define both the choreography and communications between services. The definition of a proper business process execution language is explicitly outside the scope of ebXML.
- CPA (*Collaboration Protocol Agreement*). A CPA contains the business agree-

ment among cooperating partners. It is derived from the intersection of the CPPs of the cooperating trading partners.

- WSCI (*Web Services Choreography Interface*); initially Sun, SAP, BEA and Intalio; now W3C Note (Arkin, Askary, Fordin, Jekeli, Kawaguchi, Orchard, et al., 2002). It is an XML-based interface description language that describes the flow of messages exchanged by a Web service participating in choreographed interactions with other services. WSCI is a coordination protocol, in that it does not address the definition and the implementation of the internal processes that actually drive the message exchange.
- WS-CDL (*Web Services Choreography Definition Language*); W3C Working Draft (Kavantzas, Burdett, Ritzinger, Fletcher, & Lafon, 2004). WS-CDL is an XML-based language that describes peer-to-peer collaborations of parties by defining, from a global viewpoint, their common and complementary observable behavior, where ordered message exchanges aim at accomplishing a common business goal. It is neither an "executable business process description language" nor an implementation language.
- BPML (*Business Process Management Language*); Business Process Management Initiative (BPMI.org, 2002). BPML is a language for the modeling of business processes and was designed to support processes that a business process management system could execute. BPML and WSCI share the same underlying process execution model; therefore developers can use WSCI to describe public interactions among business processes and reserve, for example, BPML for developing private implementations. However, other coordination protocols than WSCI can be adopted.
- BPEL (also BPEL4WS, *Business Process Execution Language for Web Services* or WS-BPEL); initially Microsoft, IBM, Siebel Systems, BEA, and SAP; now OASIS (*Web Services Business Process Execution Language*) (Weerawarana & Francisco, 2002). It provides an XML-based grammar for describing the control logic required to coordinate Web services participating in a process flow. BPEL can act both as coordination protocol and proper composition language. BPEL orchestration engines can execute this grammar, coordinate activities, and compensate activities when errors occur.
- OWL-S (*Ontology Web Language for Web services*); DAML.org (Martin, 2003). OWL-S is an ontology-based description language that supplies Web service providers with a set of markup language constructs for describing the properties and capabilities of their Web services at a semantic level and in an unambiguous, computer-interpretable form. It allows defining semantic descriptions as well as coordination rules. Previous releases of this language were built upon DAML+OIL and known as DAML-S. Theoretically, OWL-S is not limited to one specific grounding, but its current version provides a predefined grounding for WSDL that maps OWL-S elements to a WSDL interface (Polleres & Lara, 2005). On top of OWL-S, a *reasoner* will allow automatic service composition and execution.
- WSMO (*Web Service Modeling Ontology*); DERI (Roman, Lausen, & Keller, 2004). Based on the conceptual basis provided by the WSMF *(Web Service Modeling Framework*) (Fensel & Bussler, 2002), WSMO serves the purpose of describing various aspects of semantic Web services, ranging from coordination constraints over semantics to composition issues, and aims at solving existing integration problems. The vision of WSMO is that of an automated, goal-driven service composition that builds on pre- and post-conditions associated to component services. In its current version, WSMO does not define any grounding of services, but DERI is planning to allow multiple groundings for their service descriptions.
- IRS (*Internet Reasoning Service*) (Confalonieri, Domingue, & Motta, 2004); IRS is KMi's

Semantic Web services framework, for semantically describing and executing Web services. The IRS supports the provision of semantic reasoning services within the context of the Semantic Web. The primary goal is to support the discovery and retrieval of knowledge components (i.e., services) from libraries over the Internet and to semi-automatically compose them according to specified goals. It is based over problem solving methods, using task descriptions in terms of input roles, output roles, pre-conditions, assumptions, and goals and ontologies.

- MAIS (*Multichannel Adaptive Information Systems*) (Bianchini, De Antonellis, Pernici, & Plebani, in press; Cappiello, Missier, Pernici, Plebani, & Batini, 2004; Maurino, Modafferi, Mussi, & Pernici, 2004); the Italian MAIS research project proposes a quality-based approach to service description, selection, and composition. Web services, described with a MAIS-SDL (Service Description Language) based on WSDL and annotated with quality properties defined in WSOL (Tosic, Pagurek, Patel, Esfandiari, & Ma, 2003), are dynamically composed in context variable process executions. Web services are selected from URBE, a UDDI-compatible registry with a service ontology and service quality information. Flexibly process descriptions are specified in MAIS-PL (MAIS Process Language) and formulated associating to BPEL local and global quality constraints on the basis of information available in the current context of execution.

As the previous list and Figure 2 show, composite service designers are currently confronted with a huge amount of partly mutually exclusive, partly dependent specifications that all serve similar purposes. They are supposed to know and master all the previous specifications together with their peculiarities in order to be able to choose the right combination for their particular composition problem.

Evolution of Today's Standards

The high number of candidate standards is mainly due to two reasons: firstly, vendor-related political and strategic aspects (each one wants his own specification to become a common standard); secondly, the relatively young age of the overall Web service technologies themselves. Unavoidably, this results in a lack of stability when it comes to choose reference specifications.

Figure 3 graphically depicts the emergence of the previously-listed standards and/or specifications. Along the diagram's diagonal, a trend towards high-level and semantically enriched specifications can be derived, which enables designers to comfortably specify or to automatically derive executable service compositions.

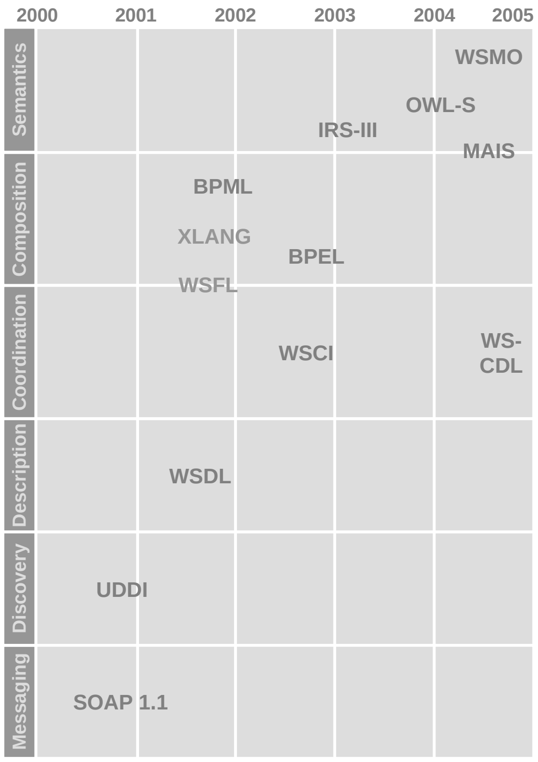
THE NEED FOR COORDINATION PROTOCOLS

As already introduced earlier, coordination and choreography describe the external message exchange that occur between a Web service and its client or among several collaborating Web services. The main concerns that have to be addressed within the coordination layer are: Can messages be sent and received in any order? Which rules govern message sequences? Is there a relationship among incoming and outgoing messages? Is it possible to undo (parts of) already executed sequences? The following sections will try to provide answers and details by discussing the conceptual backgrounds and core ideas of the most representative coordination approaches.

Conversation between Service and Client

WSDL as interface description language already provides a limited set of constructs that aim at specifying how to correctly interact with a particular Web service. Several extensions have been investigated that tried to extend the basic WSDL description with concepts for bet-

Figure 3. Emergence and evolution of today's principal standards and languages concerning WS composition. The figure tries to reflect the official release or publication dates of the specifications (at the best of the authors' knowledge), first appearance of or discussions about them could differ from the proposed dates. XLANG and WSFL are not treated in this paper; they heavily contributed to BPEL and are reported for the sake of completeness.



ter describing conversation-related aspects. Figure 4, for example, graphically depicts the problem of ordering of exchanged messages.

WSDL extensions such as WSCL (Hewlett-Packard Company, 2002) only had limited success, probably since the underlying client-server conversation model does not really fit into the service-oriented architecture of Web services. Graphically, the functionality of WSCL could best be described by a state machine model, whose expressive power allows describing conditions and ordered messages, but does not distinguish between involved actors.

Multi-service Conversations

Figure 5, for example, depicts a conversation scenario that cannot be adequately described by means of client-server protocols. The main novelty with respect to Figure 4 here is that now support for an arbitrary number of interacting services is required. Each of them plays a different role within the overall conversation. Roles are usually labeled with names like *supplier*, *purchaser*, or *broker*.

As first representative, WSCI goes one step further in its support for long lasting, choreographed and stateful message exchanges with respect to WSCL. In particular, it supports order, rules, and boundaries of messages, correlation, transactions, and compensation as

Figure 4. Ordered message exchange between a Web service and its client

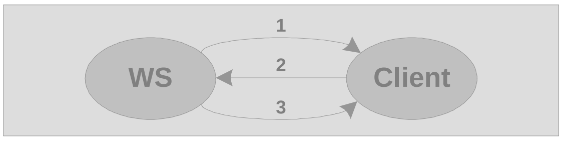
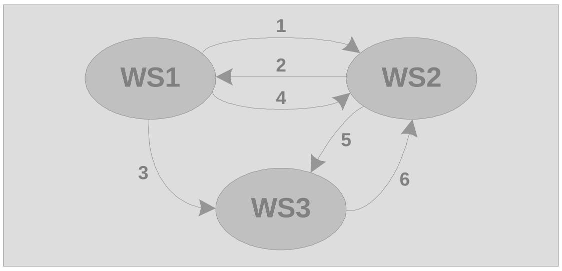


Figure 5. Interaction involving multiple Web services; messages depend semantically and chronologically from one another



well as exception handling. Through its concept of *interface*, it goes beyond simple client-server interface descriptions and supports interaction contexts with different external services, despite lacking an overall global view of the conversations a service is involved in. A WSCI interface only describes one partner's participation in a message exchange and, therefore, a WSCI choreography must include a set of WSCI interfaces, one for each partner constituting an interaction. The sample scenario in Figure 5 would thus require three different WSCI interface descriptions.

WS-CDL, the latest choreography protocol proposal, finally provides a global view over multiparty coordination through explicitly modeling all the involved roles (Kavantzas et al., 2004). Its purpose can be considered as twofold: on the one hand it provides syntactical primitives for describing involved roles and the messages exchanged during interaction, on the other hand it can be interpreted as well as binding interaction agreement between business partners that intend cooperating and require a language for formalizing their cooperation.

Other Protocols and Specifications

There also exists a set of proprietary vertical protocols, such as RosettaNet, or xCBL (*XML Common Business Library*), which provide conversation description mechanisms for specific domains. RosettaNet, for example, aims at facilitating dynamic and flexible trading relationships between business partners in the context of IT supply chains. xCBL, in the context of order management, combines an XML version of EDI (*Electronic Data Interchange*) with predefined business protocols.

Along a somewhat orthogonal dimension of the composition problem, there further exist specifications such as *WS-Coordination* or *WS-Transactions* that can be considered as meta-specifications providing a framework for the definition of proper coordination protocols with particular characteristics. For example, WS-Coordination proposes some solutions for the problem of message correlation within conversations involving several different partners. For this purpose, it defines a reference data-structure called *coordination context*, to be added to the exchanged SOAP headers, that serves the purpose of passing a unique identifier between interacting Web services.

Vinoski (2004) — in a quite critical way and without the claim for completeness — further provides an impressive list of WS-* specifications, each concerned with the support for particular functionalities:

- WS-Addressing
- WS-Agreement
- WS-Attachments
- WS-BusinessActivity
- WS-Coordination
- WS-Discovery
- WS-Enumeration
- WS-Eventing
- WS-Federation
- WS-Inspection
- WS-Manageability
- WS-MetadataExchange
- WS-Notification
- WS-PolicyFramework
- WS-Provisioning
- WS-ReliableMessaging
- WS-Resource
- WS-Security
- WS-Topics
- WS-Transactions
- WS-Transfer

As can be derived from the names of the single specifications, together all WS-* efforts are re-inventing a distributed computing platform on top of standard Web technologies. Comparable to the number of APIs available to .Net or Java/J2EE developers, the amount of WS-* specifications is continuously growing in order to provide suitable APIs and wire protocols for satisfying emerging novel interoperability requirements. The first steps towards commonly agreed on, proper programming libraries for the envisioned SOP infrastructure are being made.

Coordination Middleware

The coordination protocol specifications described in the last subsections are all so-called description languages. They are not executable languages that actively coordinate conversations among different Web services. Therefore, the necessary runtime logic must be implemented either by the services themselves or by higher-level process management languages.

Alonso et al. (2004), in order to actively support service coordination, suggest an additional middleware layer on top of the coordination layer, containing so-called *conversation controllers* with message routing and protocol compliance verification capabilities. Such conversation controllers could address the message dispatching problem arising when it comes to one Web service being engaged in several concurrent conversations. For this purpose, the *coordination context* as described by WS-Coordination could be exploited for messages correlation purposes.

FROM COORDINATION TO COMPOSITION

Despite the intrinsic passive behavior of description languages or protocols, they have proven to have enough expressive power in the context of service coordination, which indeed does not require any executable logic. However, when it comes to orchestration, things change and active support for the execution of process or flow definitions is required. Furthermore, process execution implies the need for dedicated execution environments, so-called execution or process engines able to interpret process definitions and to carry out the specified activities.

There are several different interpretations of what orchestration actually should be. Some authors refer to it as to proper programming languages, others tend to prefer a more general and evolutionary interpretation: "…these systems are often labeled the second generation *Workflow Management Systems* (WfMSs) because they provide much richer integration capabilities than traditional WfMSs…" (BPMI.org, n.d.). This second interpretation is probably too simplistic and puts too much emphasis on the business perspective of the problem. Nevertheless, current orchestration approaches definitely inherit their core modeling concepts from research in the field of WfMSs. To orchestrate services, their composition rules have to be specified at design time. Various structured process models have been proposed using traditional workflow constructs as a basis. A classification of typical workflow constructs, originating from a structured programming language approach to workflow definition, has been proposed by Van der Aalst, ter Hofstede, Kiepuszewski, and Barros (2003). The following subsections provide insight into composition approaches and issues in the context of Web services.

Model-Based Composition

Model-based service composition approaches concentrate on the explicit definition of the possible process flow that governs a composite Web service. Such process definitions are fed into a process or execution engine that manages the overall execution of the compound activities and thus actively orchestrates the composite service. Commercial composition tools usually provide intuitive high-level visual modeling tools that aid designers in the predominantly explicit definition of processes, such as Microsoft's BizTalk *Orchestration Designer* (Microsoft Corporation, n.d.) or Oracle's *BPEL Process Manager* (Kennedy, 2005). Internally, these models are then translated into low-level process models for execution purposes. Several approaches for internal process structures have been proposed; in the following we provide a brief overview, without going too deep into detail.

State Charts and Petri Nets

State charts and Petri nets (or extensions of them) are classical and well known formalisms within computer science. They have already proven their viability in the context of workflow modeling and are mentioned here merely for the sake of completeness; further details can be found in (Alonso et al., 2004). Within the Web service domain, IBM's WSFL, for example, internally uses Petri net models for expressing the process logic; Benatallah, Sheng, and Dumas (2003) ground their declarative service composition approach *Self-Serv* on state charts.

Pi-Calculus

Less intuitive and without graphical representation are process specifications based on Pi-Calculus (Alonso et al., 2004). Pi-Calculus is a process algebra and an attempt at developing a formal theory for process models. As happens with Petri nets, the main advantage is represented by the fact that a precise and well-studied formalism can provide the basis for the verification of its properties. Microsoft's XLANG specification, for example, is inspired by Pi-Calculus theory.

Rule-Based Orchestration

Another textual technique for specifying orchestration schemas is provided by rule-based orchestration languages that provide constructs for specifying processes by means of sets of rules (Alonso et al., 2004). Usually, such rules are based on the so-called *event-condition-action* (ECA) paradigm known from active database systems. This technique is less structured with respect to the previous models and is mainly suited to model orchestrations that have only few constraints among activities.

Two Representatives of Structured Process Models: BPEL(4WS) vs. BPML

BPEL is an XML-based Web service composition language that is rooted in both Microsoft's XLANG and IBM's WSFL. In BPEL, a composite service is named a *process*; processes export and import functionality by using Web service interfaces exclusively. Two main kinds of processes are distinguished: *ab-*

stract processes describe business protocols, specifying the mutually exchanged messages and their invocation order by each of the parties involved, *executable processes* bind the specified behavior to concrete services. According to this twofold applicability, BPEL occupies both the *Coordination* and *Composition* layers within the protocol stack depicted in Figure 2. Besides processes, participating services are called *partners*, and message exchanges or intermediate result transformations are called *activities*. BPEL distinguishes between basic and structured activities. *Basic activities* represent synchronous and asynchronous calls (<invoke>, <invoke>…<receive>), *structured activities* manage the overall process flow (<flow> to denote parallelism, <switch> for alternatives...).

BPEL is designed primarily as a composition language, but developers can use the same formalism for both service composition and conversation definition. As such, it lacks many of the necessary and, from a discovery and binding perspective, particularly useful properties needed for defining conversations (for activation, for example). Furthermore, the structure of BPEL is flat, that is, sub-processes cannot be defined.

BPML, with respect to BPEL, provides similar modeling capabilities, but also supports some additional constructs, making it more flexible in general, such as sub-processes, and so on. In particular, the BPML specification provides an abstract model and an XML syntax for expressing executable business processes. But, BPML itself does not define any application semantics, it rather defines an abstract model and grammar for expressing generic processes. This allows BPML to be used for a variety of purposes that include, but are not limited to, the definition of enterprise business processes, the definition of complex Web services, and the definition of multi-party collaborations. BPML is conceived as block-structured programming language. Recursive block structures play a significant role in scoping issues that are relevant for declarations, definitions and process execution.

Both BPEL and BPML provide support for long-running business transactions and robust exception handling facilities. BPML does not provide constructs for the definition of message coordination protocols as BPEL does, but developers easily can use WSCI for this purpose, which shares the same underlying process execution model. This apparent shortcoming of BPML, on the other hand, allows for a more flexible use of BPML and WSCI when it comes to defining conversations, due to the good separation of concerns. Currently, there is, however, less industry support for BPML in comparison to BPEL.

Ontology-Driven Composition

Besides explicit process modeling approaches, the Semantic Web and service ontologies offer alternative ways for the composition and execution of compound services. This kind of approach, rather than concentrating on an explicit definition of the flow logic, aims at providing suitable frameworks for the automatic derivation and execution of composite services, defined in an implicit manner by means of goals as well as pre- and post-conditions over service inputs and outputs.

For example, Arpinar, Aleman-Meza, Zhang, and Maduko (2004) propose an ontology-driven Web services composition platform where the requirements of the composite services are specified by users as inputs and expected outputs. The described approach allows automatically generating and executing a composite service that produces the expected outputs by combining existing individual services using their semantic descriptions. A human-assisted and an automatic composition mechanism are outlined.

Two Emerging Standards: OWL-S vs. WSMO

OWL-S allows providers of Web services to describe properties, capabilities, and behaviors of their services by means of ontologies, and provides proper language primitives for their semantic description. Final goal of OWL-

S is to provide a machine-interpretable description of services, in addition to the human-understandable descriptions already provided by WSDL, and thus to support automatic discovery, execution and composition. The core of OWL-S, the ontology-driven description approach, builds on the *Ontology Web Language* (OWL) (Martin, 2003), which provides the necessary constructs for explicitly representing the meaning of terms and the relationships existing among them within a specific domain. OWL and OWL-S are evolutions of DAML+OIL, a semantic markup language for Web resources.

OWL-S ontologies are structured into three main parts: A *service profile* serves the purpose of advertising and discovering services published by service providers and contains a semantically enriched and machine-interpretable service description. A *process model* describes how a service operates (by means of proper control constructs and conversation descriptions) and comprises inputs, outputs, preconditions, results and effects of the service. According to their complexity *atomic*, *simple* and *composite* processes are distinguished, being the latter the most complex one. The third part, the service *grounding* provides the necessary details for accessing a specific service, that is, protocols and message formats. Whereas *profile* and *model* provide rather abstract representations, the *grounding* refers to the concrete specification. The semantics- and ontology-based approach adopted by OWL-S is particularly suited for advanced service and conversation description.

WSMO aims as well at describing relevant aspects of semantic Web services. Within the *Web Service Modeling Framework* (WSMF), WSMO provides an (open source) executable solution for goal-driven service composition through extensive use of ontologies, semantic service descriptions and pre- and post-conditions for service description. Besides ontologies, goals and service descriptions, so-called mediators should bypass interoperability problems. Interoperability is one of the main issues WSMO tries to solve, and this aspect differentiates it from OWL-S.

Just as for OWL-S, *ontologies* provide the formal semantics that allows for automatic information processing and for human- and computer-understandable goal definitions. A *goal* specification expresses the final objective a client may have when interacting with a service and consists primarily of constrains over post-conditions after service execution. *Mediators* provide the necessary support for integrating heterogeneous elements when combining several component services. They define mappings and transformations between connected elements. Four types of mediators exist, according to the elements they link: goal-goal mediators, ontology-ontology mediators, Web-service-goal mediators, and service-service mediators. Finally, *Web services* are described by means of their non-functional properties, the mediators they use, their capabilities, interfaces and groundings.

DERI is further working on an execution environment for WSMO-based Web services, the so-called *Web Services Execution Environment* (WSMX) (Haller, 2005). The goal of WSMX is that of providing an environment for the dynamic inter-operation of Web services, including automatic discovery, selection, mediation and invocation mechanisms.

Other Composition Approaches

Besides proper language or protocol standardization efforts, several academic research works go one step further in service composition and also investigate the value of additional aspects of the composition problem, such as QoS, personalization, or context.

In Meteor-S, process composition is annotated with information for selecting services according to quality of service characteristics (Sivashanmugam, Miller, Sheth, & Verma, 2004). Optimization of service selection has been considered and evaluation functions discussed. The approach is mainly oriented to design, giving the possibility of transforming the process representation into BPML or BPEL process specifications.

Maamar, Mostefaoui, and Yahyaoui (2005) extend their state-chart-based service composition model with an agent-based and context-oriented approach to composite service execution. The term *context* reflects the point of view of services rather than to the one of users. At runtime, agents are engaged in conversations with their peers on behalf of the user to agree on the actual Web services to participate in the process, according to the runtime context conditions and the global composition model.

Baïna, Benali, and Godart (2003), finally, provide a valuable approach to Web service composition within the initially mentioned workflow domain and with special focus on enterprise workflow interconnection. The process interconnection model presented by the authors builds on Web service-based workflow integration and allows for heterogeneous workflow systems coexisting in a so-called "workflow of workflows". The main contribution of the work consists in the introduction of a certain level of dynamism, proper of the Web services area, into workflow definitions; more precisely, the authors postpone the selection of nested sub-processes from build-time to runtime, by introducing proper discovery, negotiation and wrapping mechanisms for so-called process services.

In MAIS, services are selected at runtime according to constraints on functionalities and quality of service expressed at design time and the current context for process execution (De Antonellis, Melchiori, De Santis, Mecella, Mussi, Pernici, et al., 2005; Maurino et al., 2004).

In all these approaches, traditional composition patterns are enriched with additional features that allow flexible process specification and execution. The principal trends are toward providing a precise definition of context and of local and global constraints and dynamic service selection and invocation. No new composition constructs are defined; however, new composition mechanisms and optimization of composed services are discussed in the literature.

In choreography specifications, there is less attention toward such quality related aspects, except from temporal constraints on the conversations. However, in this paper we do not discuss in depth these issues since they are only marginally relevant in the comparison of coordination and composition approaches.

Service Selection

As the reader will have noticed, one of the main novelties introduced by these two research efforts, as well as by the ontology- or semantics-driven composition approaches, consists in the dynamic selection of the services to be composed, besides the dynamic service composition itself.

Service selection is probably the point where current orchestration approaches definitely could add flexibility with respect to traditional WfMSs, which usually include a (centralized) resource manager that at runtime decides to which resource instance, respecting a precise role definition, a specific task should be assigned (WfMC, n.d.). The question, hence, is whether component services should be selected at process *definition time* or at r*untime* during process execution. Some authors even distinguish between service selection at *design time* and *deploy time*. The overall purpose of dynamic service selection is mostly that of guaranteeing the availability of a composite service, being the Web a highly variable and fast changing environment.

Currently, *static* (hard-coded within the process definition) selection approaches prevail over dynamic ones (Alonso et al., 2004). The URIs for locating the necessary services are uniquely defined at design time and each process instance refers to the same set of services. Instead of hard-coding the URIs within the process definition, they can also be assigned to process variables and thus determined as a result of a previously executed operation. These approaches are known as *dynamic by reference*. A further degree of flexibility is provided by so-called *dynamic by lookup* binding mechanisms that support, for each activity, the definition of a query to be executed on some service directory and thus require a certain level of middleware support.

Selection decisions not only are influenced by the selection time, but — and even at a higher degree — by the selection algorithm itself. As the ontology-driven approach shows, semantic and goal-driven considerations could drive the selection algorithm (Arpinar et al., 2004), as well as context-based or QoS-driven ones. Also, syntactical similarities or abstract services as representatives for a specific class of equivalent services could constitute the decision domain.

UDDI provides basic functionalities to retrieve services according to their classification, providers and/or tModels. Recent proposals have emerged to support WSMO and OWL-S service selection using IRS (Confalonieri et al., 2004), using the IRS discovery and retrieval mechanisms, mapping semantic service descriptions provided by those two approaches to the knowledge representation language OCML (Hakimpour, Domingue, Motta, Cabral, & Lei, 2004).

In the URBE registry developed for MAIS, services are selected from the registry according to their functional characteristics, organized according to a service model), their quality characteristics, the invocation context, and application or user requirements (Bianchini et al., in press). Similarity functions are provided to assess the functional suitability of a service, according to given functional and non-functional requirements, in conjunction with a lightweight ontology model.

Message Correlation

Once the services that constitute the composite service have been selected, another (runtime) problem must be addressed: message correlation. As there may be several concurrent instances of the same composite service running within one and the same execution environment, these process instances and the conversations they are involved in with external Web services must be uniquely identified for guaranteeing a correct overall process execution.

WS-Coordination proposes identifiers (the *coordination context*) carried by SOAP headers for uniquely associating messages to conversations. When using WSCI, designers can identify certain data items within exchanged messages that act as unique identifiers of the conversation. A possible process specification on top of these protocols must explicitly provide the necessary logic implementing the described mechanisms.

On the other hand, BPEL already proposes a solution at process level, namely so-called *correlation sets* that — similar as within WSCI — allow defining sets of data items as unique identifiers. By assigning the same correlation set to multiple messages, the designer can specify that messages — whenever the respective data items have the same values — belong to the same process instance or conversation.

Transactions and Exception Handling

As Web services aim at supporting collaborations between business partners, robust transaction support is required. The classical ACID properties of relational databases have proven being too strict in a service-oriented environment involving several autonomous business partners, and thus, in this context, they have to be slightly relaxed. Also, compensating mechanisms must be taken into consideration, as already happened for WfMSs (Grefen, Pernici, & Sanchez, 1999).

In August 2002, IBM, Microsoft, and BEA proposed WS-Transaction, a standard protocol for long-running business transactions that builds on the framework provided by WS-Coordination. Transactions are one way to handle exceptions, but due to its compensation mechanism not in every exceptional situation transactions provide the right functionality. Several exception handling approaches are known, the most important ones are *try-catch-throw* mechanisms as provided, for example, by Java and currently implemented in BPEL, or *flow-based* mechanisms that consist in explicitly modeling the error testing logic within the proper process description. Also, *rule-based* approaches exist, which are particularly suited for handling temporal exceptions.

A more detailed discussion of transactions and exception handling mechanisms would exceed the scope of this paper. As well, other issues like data conversion between different component services or execution monitoring are not addressed here.

HOW ORCHESTRATION DEPENDS ON CHOREOGRAPHY

After this overview over Web service choreography and orchestration and the main concerns they address, in this section we will try to briefly highlight to what extent the one depends on the other. For this purpose, we distinguish three main dimensions: structural, functional, and resource dependencies.

Structural Dependencies

Structural dependencies are those driving the overall structure or organization of a process definition, and thus concern involved activities, conditions, ramifications within the process flow, and so on.

Alonso et al. (2004) well explain the dependencies between coordination protocols and composition schemas by stepwise refining the portion of a process definition relative to only one of the participating services. Starting from an overall activity diagram, the authors first extract the role-specific view of the process and then refine it in order to reach a granularity level where the single activities of the remaining diagram reflects the single service invocations required for achieving the specific functionality. This so-called process skeleton on the one hand describes the role-specific view of the process, on the other hand provides a proper protocol description of that participant's public interactions. In this way, the authors show how the definition of the executable process intrinsically must match the constraints imposed by the underlying coordination protocol.

Functional Dependencies

Functionalities or capabilities like transaction support, security, reliability, correlation, and so on may yield to functional dependencies among orchestration languages and coordination protocols, like those provided by the wealth of WS-* specifications. Dependencies arise, whenever the functionalities they provide are used within a process specification and the composition language "delegates" the relative competencies to the underlying coordination protocols.

As already exemplified earlier, for example, coordination can be achieved either explicitly at process level or implicitly at coordination level. For example, once the choice of adopting the WS-Coordination framework has been made, the process definition does not anymore require explicit coordination constructs. The same considerations also hold in case of transaction support, reliable messaging, or the like.

Resource Dependencies

Most of the process definition languages have inherited their modeling approaches from the field of workflow management. At process or composition design time, however, service composition presents some methodological differences that are rooted in the dependencies that exist between coordination and composition.

WfMSs allow for a straightforward top-down structure of the process model, describing, for example, an administrative workflow. Resources executing a specific work item are provided with the exact amount of data that is required for the correct execution of that task. For executing one task, there is no need to know about possible other tasks before or after that specific task within the same process flow. Possible task constellations are subject only to the constraints imposed by the final goal of the underlying business process. Involved resources do not have a task-surviving behavior with constraints affecting the overall process definition. Rearranging tasks (i.e., putting some in parallel), when specifying process definitions, is common practice for improving process efficiency.

When defining the logic that constitutes a composite Web service, a strict top-down approach does not guarantee anymore that the resulting process definition is always executable. As already outlined earlier when speaking about the need for coordination protocols, a Web service may by subject to certain conversation rules in order to be executed correctly. For example, before accepting a user's credit card number for payment, the service must be provided with an appropriate list of goods the user wants to buy. This externally visible behavior of Web services distinguishes the resource *Web service* from those we have in WfMSs. Single tasks cannot anymore be rearranged arbitrarily without loosing functionality.

Composite service designers must know about the coordination requirements of the services they use and take them into account when defining composite services. Thus, starting from an initial process idea (top-down), designers select the services providing the right functionality, and then refine their initial idea (by rearranging initially presumed invocations or adding new ones) in order to conform with the coordination requirements the selected services impose (bottom-up). Therefore, the resulting process definition combines the advantages of both a coarse-grained top-down approach and a fine-grained bottom-up method.

FUTURE TRENDS

The previous considerations outlined the main characteristics of Web service choreography and orchestration and also presented some mutual dependencies between the two, by paying particular attention to the various ongoing standardization efforts that finally should lead to commonly agreed upon protocols and languages. In particular, we argued that coordination protocols are public documents focusing on external interactions, and composition schemas are private documents that describe the internal implementation of composite Web services.

Coordination or Composition?

First, we will focus on the trends concerning coordination and composition approaches and their relationships. In our view, both perspectives will be needed also in the future and more research work should focus on formally relating the two approaches, also in order to be able to prove formal properties which are published against formal properties of private process descriptions.

Trends in Private Process Descriptions

In order to close the circle started in the introduction when speaking about *Service-Oriented Programming*, it is interesting to remark that the solutions found so far do not provide radically novel programming capabilities. In fact, one could even imagine specifying a composite service that makes use of several third-party services by using conventional programming languages; for example, Java provides all the necessary primitives for coping with coordination and composition. But the emerging languages will provide higher-level reasoning capabilities and better, service-centered abstractions.

As further alluded within the introduction when comparing SOP with OOP, where really valuable and novel concepts primarily emerged as result of the object-oriented paradigm and less because of the availability of object-oriented languages, also in the context of Web services the real potential resides in what will be build on top of SOP languages rather than in such languages themselves. Just as today's enterprise application servers run so-called *object containers* as execution environment for business logic and offering various services to its components, similar concepts are being investigated also for Web services and probably will substantially enhance current composition capabilities.

Benatallah et al. (2003), in their *Self-Serv* research project, are concentrating on a middleware infrastructure for the composition of Web services that allows for multi-attribute dynamic service selection within a composite service and peer-to-peer orchestration. Furthermore, they build on the concept of *service container* aggregating several substitutable services.

A similar approach is followed by the

MAIS project (MAIS, n.d.) that — among others — aims at the definition of a platform for dynamic service selection and provisioning on the basis of context and QoS information. Compatible services are grouped into so-called *abstract services* and allow dynamically selecting and when necessary substituting (concrete) services at runtime according to the current context and the result of a negotiation over QoS parameters.

In general the trend is towards providing a middleware (environments supporting WS-*) to support dynamic process execution and more integration with programming environments, both in the Semantic Web service line, which is strictly related to logic programming, and in the composition line, such as for instance in BPEL extensions allowing Java code to be included in the process specification.

Trends in Public Process Description

In this area the trend is to define constraints on messages being exchanged among several partners, without enforcing coordination through execution engines. Some support can be provided to verify, at runtime, whether a given coordination specification has been violated (such as, for instance, in Maamar et al., 2005).

Open or Closed Worlds?

Slightly different approaches are emerging from the recent trend towards Semantic Web services and still have to be profoundly investigated. Most of the efforts in this context, like OWL-S and WSMO, are covered by research and academic communities and still have to prove their commercial viability. Nevertheless, especially for dynamic service selection the potential seems to be promising.

However, in this research area much effort is devoted to the capability of handling multiple ontologies, such as in OWL-S, or in providing mediators between them, such as in WSMO. The ability of combining logics and providing general reasoning mechanisms is limited, so the trend could be a greater focus on closed world or communities of service providers and users such as defined in (Marchetti, Pernici, & Plebani, 2004).

CONCLUDING REMARKS

The time being seems of crucial importance for the success of Web services. Decisions have to be made about future standards, which will heavily influence the potential for success. In his critical article on the practice of standardization, *WS-Nonexistent Standards*, Vinoski (2004) not only complains about the numerous proposed standards, but also about the way they are proposed. As a charter member of the World Wide Web Consortium's Web Services Architecture working group, he asks for more consensus in the standardization processes. Today, he says, traditional standardization procedures are often bypassed by powerful vendors, which develop their own specifications and only afterwards submit them to an official standards body with the hope of fast acceptance and minimal changes. In this short-circuited standardization effort he identifies both a disadvantage for users and a threat for the overall success of the technologies to be standardized.

Therefore, let us hope in shared and agreed on standards as basis for the next generation applications and services, because "…a standard that is not generally agreed on is a standard on paper only" (Vinoski, 2004).

REFERENCES

Aissi, S., Malu, P., & Srinivasan, K. (2002). E-business process modeling: the next big step. *IEEE Computer, 35*(5), 55-62.

Alonso, G., Casati, F., Kuno, H., & Machiraju, V. (2004). *Web services — Concepts, architectures and applications*. Berlin Heidelberg: Springer-Verlag.

Arkin, A., Askary, S., Fordin, S., Jekeli, W., Kawaguchi, K., Orchard, D., et al. (2002, August). *Web Service Choreography Interface (WSCI) 1.0* (W3C Note). Retrieved January, 2005, from http://www.w3.org/TR/wsci

Arpinar, B., Aleman-Meza, B., Zhang, R., &

Maduko, A. (2004). Ontology-driven Web services composition platform. In *Proceedings of the IEEE International Conference on E-Commerce Technology*. IEEE.

Baïna, K., Benali, K., & Godart, C. (2003, November 3-7). Dynamic interconnection of heterogeneous workflow processes through services. In *Proceedings of the 11th International Conference on Cooperative Information Systems (CoopIS'03); Confederated International Conferences (DOA/CoopIS/ODBASE'03), LNCS 2888*, Catania, Sicily, Italy. Springer-Verlag.

Benatallah, B., Casati, F., & Toumani, F. (2004). Web service conversation modeling: A cornerstone for e-business automation. *IEEE Internet Computing, 8*(1), 46-54.

Benatallah, B., Sheng, Q. Z., & Dumas, M. (2003). The Self-Serv environment for Web services composition. *IEEE Internet Computing, 7*(1), 40-48.

Bianchini, D., De Antonellis, V., Pernici, B., & Plebani, P. (in press). Ontology-based methodology for e-Service discovery. *Information Systems*.

Bieber, G. & Carpenter, J. (2001). *Introduction to service-oriented programming* (rev. 2.1). Retrieved January, 2005, from http://www.openwings.org/download/specs/ServiceOrientedIntroduction.pdf

BPMI.org. (2002). *BPML/BPEL4WS — A convergence path toward a standard BPM stack* (BPMI.org Position Paper). Retrieved January, 2005, from http://www.bpmi.org/

BPMI.org. (n.d.). *Business process management initiative*. Retrieved January, 2005, from http://www.bpmi.org/

Cappiello, C., Missier, P., Pernici, B., Plebani, P., & Batini, C. (2004). QoS in multichannel IS: The MAIS approach. In *Proceedings of the International Workshop on Web Quality (WQ'04) in conjunction with the ICWE 2004*, Munich, Germany.

Cardoso, J., Bostrom, R. P., & Sheth, A. (2004). Workflow management systems and ERP systems: Difference, commonalities, and applications. In *Information Technology and Management 5* (pp. 319-338). Kluwer Academic Publishers.

Chappell, D. (2004). *Understanding BPM servers*. Chappell & Associates. Retrieved December, 2004, from http://www.microsoft.com/biztalk/techinfo/default.asp

Confalonieri, R., Domingue, J., & Motta, E. (2004, December 8). Orchestration of Semantic Web services in IRS-III. In *Proceedings of the First AKT Workshop on Semantic Web Services (AKT-SWS04) KMi*, The Open University, Milton Keynes, UK.

De Antonellis, V., Melchiori, M., De Santis, L., Mecella, M., Mussi, E., Pernici, B., et al. (2005). *A layered architecture for flexible e-service invocation*. Software Practice & Experience, in press.

Dustdar, S., & Schreiner, W. (2004). *A survey on Web services composition*. Technical University of Vienna, Distributed Systems Group.

Eisenberg, B., & Nickull, D. (2001). *ebXML technical architecture specification v1.04*. Retrieved January, 2005, from http://www.ebxml.org/specs/index.htm

Fensel, D., & Bussler, C. (2002). The Web Service Modeling Framework WSMF. *Electronic Commerce: Research and Applications, 1*(2002), 113-137.

Grefen, P., Pernici, B., & Sanchez, G. (1999). *Database support for workflow management — The WIDE project*. Kluwer.

Hakimpour, F., Domingue, J., Motta, E., Cabral, L., & Lei, Y. (2004). Integration of OWL-S into IRS-III. In *Proceedings of the First AKT Workshop on Semantic Web Services (AKT-SWS04), KMi*, The Open University, Milton Keynes, UK.

Haller, A. (2005, January). *D7.3v1.0 mission statement — WSMX* (WSMX Working Draft). Retrieved December, 2004, from http://www.wsmo.org/2005/d7/d7.3/v1.0/20050109/

Hewlett-Packard Company. (2002, March). *Web Services Conversation Language (WSCL) 1.0* (W3C Note). Retrieved December, 2004, from http://www.w3.org/TR/wscl10/

Jung, J., Hur, W., Kang, S., & Kim, H. (2004).

Business process choreography for B2B collaboration. *IEEE Internet Computing, 8*(1), 37-45.

Kavantzas, N., Burdett, D., Ritzinger, G., Fletcher, T., & Lafon, Y. (2004, October). *Web services choreography description language version 1.0* (W3C Working Draft). Retrieved December, 2004, from http://www.w3.org/TR/ws-cdl-10/.

Kennedy, M. (2005, April). *Oracle BPEL process manager quick start guide, 10g (10.1.2)* (Beta Draft). Retrieved May, 2005, from http://download-uk.oracle.com/otndocs/products/bpel/quickstart.pdf

Khalaf, R. & Nagy, W. A. (2003, April). *Business process with BPEL4WS: Understanding BPEL4WS, Part 7, Adding correlation and fault handling to a process* (Research Report). IBM developerWorks. Retrieved January, 2005, from http://www-106.ibm.com/developerworks/ webservices/ library/ws-bpelcol7/.

Langdon, C. S. (2003). The state of Web services. *IEEE Computer, 36*(7), 93-94.

Leavitt, N. (2004). Are Web services finally ready to deliver? *IEEE Computer, 37*(11), 14-18.

Maamar, Z., Mostefaoui, S. K., & Yahyaoui, H. (2005). Toward an agent-based and context-oriented approach for Web services composition. *IEEE Transactions on Knowledge and Data Engineering, 17*(5), 686-697.

MAIS. (n.d.). MAIS project home page. Retrieved January, 2005, from http://www.mais-project.it

Marchetti, C., Pernici, B., & Plebani, P. (2004). A quality model for multichannel adaptive information. In *WWW (Alternate Track Papers & Posters) 2004* (pp. 48-54), New York.

Martin, D. (2003). OWL-S: Semantic markup for Web services (White Paper). The OWL Services Coalition. Retrieved December, 2004, from http://www.daml.org/services/owl-s/1.0/owl-s.html

Maurino, A., Modafferi, S., Mussi, E., & Pernici, B. (2004). A framework for provisioning of complex e-services. In *IEEE International Conference on Services Computing (SCC 2004)*, Shanghai.

Microsoft Corporation. (n.d.). Microsoft BizTalk Server. Retrieved January, 2005, from http://www.microsoft.com/biztalk/

Milanovic, N. & Malek, M. (2004). Current solutions for Web service composition. *IEEE Internet Computing, 8*(6), 51-59.

Paulson, L. D. (2002). Choreographing Web services. *IEEE Computer, 35*(11), 25.

Peltz, C. (2003a). *Web services orchestration — A review of emerging technologies, tools, and standards*. Hewlett-Packard Company.

Peltz, C. (2003b). Web services orchestration and choreography. *IEEE Computer, 36*(10), 46-52.

Polleres, A. & Lara, R. (2005, January). *D4.1v0.1 A Conceptual Comparison between WSMO and OWL-S* (WSMO Working Draft). Retrieved January, 2005, from http://www.wsmo.org/2004/d4/d4.1/v0.1/20050106/

Roman, D., Lausen, H., & Keller, U. (2004, September). *D2v1.0. Web Service Modeling Ontology (WSMO)* (WSMO Working Draft). Retrieved January, 2005, from http://www.wsmo.org/2004/d2/v1.0/20040920/

Sivashanmugam, K., Miller, J., Sheth, A., & Verma, K. (2004, February). Framework for Semantic Web process composition [Special issue]. *International Journal of Electronic Commerce*.

Smith, M. K., Welty, C., & McGuinness, D. L. (2004, February). *OWL Web ontology language guide* (W3C Recommendation). Retrieved January, 2005, from http://www.w3.org/TR/2004/REC-owl-guide-20040210/

Tosic, V., Pagurek, B., Patel, K., Esfandiari, B., & Ma, W. (2003). Management applications of the Web Service Offerings Language (WSOL). In *CAiSE 2003* (pp. 468-484).

Van der Aalst, W. M. P., ter Hofstede, A. H. M., Kiepuszewski, B., & Barros, A. P. (2003). Workflow patterns. *Distributed and Parallel Databases, 14*(3), 5-51.

Vinoski, S. (2004). WS-nonexistent standards. *IEEE Internet Computing, 8*(6), 94-96.

Weerawarana, S. & Francisco, C. (2002, August). *Business process with BPEL4WS: Under-*

standing BPEL4WS, Part 1, Concepts in business processes (Research Report). IBM developerWorks. Retrieved January, 2005, from http://www-106.ibm.com/developerworks/webservices/library/ws-bpelcol1/

WfMC (Workflow Management Coalition). (n.d.). Retrieved January, 2005, from http://www.wfmc.org

Wiederhold, G., & Wegner, P., & Ceri, S. (1992). Toward megaprogramming. *Communications of the ACM, 35* (11), 89-99.

W3C (World Wide Web Consortium). (n.d.). Retrieved January, 2005, from http://www.w3.org

Florian Daniel is a PhD candidate in Information Technology Department at Politecnico di Milano. His main research interests include conceptual modeling techniques for data-intensive Web applications and Web services, active/reactive Web applications and workflow management systems. His current research activities focus on modeling of multi-channel and context-aware Web applications within the conceptual framework provided by WebML (Web Modeling Language). He is actively participating in the Italian research project FIRB MAIS (Multichannel Adaptive Information Systems). He graduated with full marks (cum laude) in computer science engineering at Politecnico di Milano in 2003. In 2004 he won a three-year grant for his PdD studies by the Italian Department of Education (MIUR).

Barbara Pernici is full professor of computer engineering at Politecnico di Milano. Her research interests include cooperative information systems, workflow management systems, information systems modeling and design, temporal databases, and applications of database technology. She holds a Dr. Eng. from Politecnico di Milano and a master's of science in computer science from Stanford University. She has published 35 papers in international journals, including IEEE *and* ACM Transactions, *co-edited 10 books, and published about 130 papers at the international level. She is an editor of the* Requirements Engineering Journal. *She has participated in several ESPRIT/IST projects (TODOS, Equator, ITHACA, F3, WIDE, Chorochronos). She is chief scientist of the Italian FIRB MAIS (Multichannel Adaptive Information Systems), 2002-2005. She is chair of Working Group 8.1 Design and Evaluation of Information Systems of IFIP (International Federation for Information Processing).*