Chapter XII
Web Service Orchestration and Choreography:
Enabling Business Processes on the Web

Florian Daniel
Politecnico deo Milano, Italy

Barbara Pernici
Politecnico deo Milano, Italy

ABSTRACT

The Web service domain is a fast growing and fast changing environment. From a business perspective, the trend over the last few years in the Web services area firmly points toward seamless business logic integration and inter-enterprise collaboration. However, in order to accomplish such 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. In this chapter, we try to provide a critical snapshot of current standards for Web service development and particularly we focus on Web service orchestration and choreography. We discuss problems and solutions from a conceptual point of view, exemplify the illustrated ideas by means of real-world technologies and standards, and highlight the mutual dependencies that exist among orchestration and choreography of Web services.[1]

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 toward the adoption of novel and emerging Web service technologies as basis for the next generation of (Web) applications and composite Web services. In this context, especially the need for flexible solutions for composing Web services into composite

applications or services is manifest. Composite applications or services leverage the functionalities provided by their individual component services by combining them in a value adding manner.

Web services are driven by the paradigm of the so-called *service-oriented architecture* (SOA), which describes the relationships that exist among service *providers*, service *consumers,* and service *brokers* and thereby provides an abstract execution environment for Web services. The research area of *service-oriented computing* (SOC) endorses the SOA paradigm and aims at producing technologies and solutions that address the efficient development, flexible composition, and execution of (composite) Web services. From their first appearance, SOA and SOC have emerged as key factors for the success of the world of Web services.

Just as the advent of *object-oriented programming* (OOP) was based on the notion of *objects* as means to modularize programming functionality, SOC could be defined as a paradigm that looks at *services* as basic functional modules that can be composed or newly defined. OOP per se did not suddenly provide revolutionary new programming capabilities with respect to conventional procedural techniques, it rather proved to be an efficient means for abstraction and 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 finally enhanced interoperability.

Analogously, current specification proposals for Web services can be interpreted as a transition toward a robust SOC framework. Several Web service standardization bodies are currently addressing issues that can be interpreted as 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 or transaction support) that have already been solved or are under study in other research areas. Past experiences taught us, however, that as long as there are no robust and commonly agreed on standards, real interoperation, and composition problems cannot be addressed adequately.

In this chapter, we will introduce the reader to the orchestration and choreography of Web services, which are becoming the cornerstones for the execution of business processes on the Web, and we will discuss the state of current research and open issues. More precisely, we will first try to clarify the main terminology in use, and then we will give an explanation for the actual need for coordination protocols and composition technologies. We will exemplify such a discussion by means of a possible protocol stack for Web service composition, and we also discuss some advanced issues. Finally, we will provide on outlook over expected future trends and draw our conclusions.

USING THE RIGHT TERMINOLOGY

Specifications and technologies for Web service composition in many cases still have to reach stable definitions and usage scenarios. Accordingly, 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…" (Peltz, 2003).

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 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 (i.e., an orchestration is the pattern of interactions that a Web service agent must follow in order to achieve its goal) (W3C, n.d.).

This terminological comparison shows that different authors prefer different names and thereby 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, we distinguish 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 Figure 1 also represents the dimension *actor*, which allows the distinction 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.

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 designer and execution time (Daniel & Pernici, 2006).

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

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

THE NEED FOR COORDINATION PROTOCOLS

According to the previous characterization, coordination and choreography describe the external message exchanges 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? In the following, we will try to provide answers and details to some of these questions by discussing the conceptual backgrounds and core ideas of the most representative coordination approaches.

Conversation Between Service and Client

WSDL, the *Web service description language* (W3C, 2001), in its function of 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 better describing conversation-related aspects. Figure 2, for example, graphically depicts the

Figure 2. Ordered message exchange between a Web service and its client

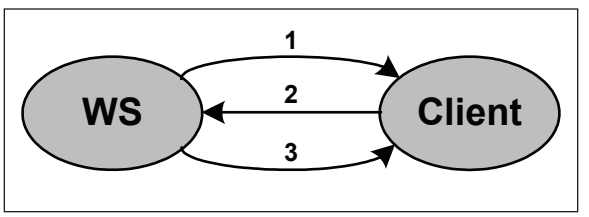
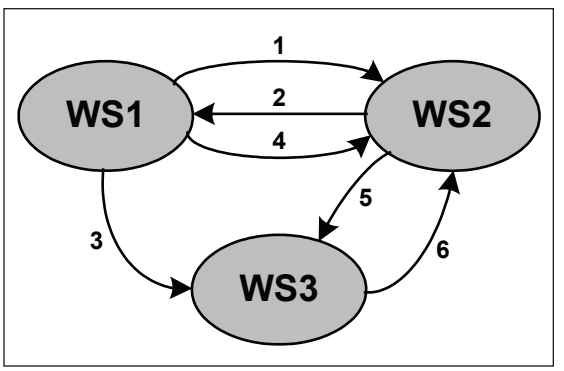


Figure 3. Interaction involving multiple Web services; messages depend semantically and chronologically from one another.



problem of ordering messages exchanged between a Web service and its client.

WSDL extensions such as WSCL (*Web services conversation language*; Hewlett-Packard Company, 2002) only had limited success, most likely due to the fact that its 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 the description of conditions and ordered messages, but does not distinguish between involved actors.

Multi-Service Conversations

Figure 3 depicts a slightly more complex conversation scenario that, for example, cannot be adequately described by means of a client-server protocol. The main novelty with respect to Figure 2 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, and only the strict adherence to such roles leads to the fulfillment of the common (business) goal. Roles are usually labeled with names like *supplier*, *purchaser*, or *broker*.

As a first representative choreography protocols, WSCI (*Web services choreography interface*; Arkin et al., 2002) 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 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 in fact 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 3 would thus require three different WSCI interface descriptions.

WS-CDL (*Web services choreography definition language*; Kavantzas et al., 2005; Ross-Talbot & Fletcher, 2006), the latest choreography protocol proposal, finally provides a global view over multiparty coordination through the explicit modeling of all the involved roles. 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 are intended to start a cooperation and require a language for formalizing their cooperation.

Advanced Protocols and Specifications

As opposed to the previous coordination protocols, which all can be considered domain-independent, there also exists a set of proprietary, domain-specific vertical protocols such as RosettaNet (RosettaNet, 2006), or xCBL (XML Common Business Library; xCBL.org, 2006), 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 exists specifications such as *WS-coordination* or *WS-transactions* that can be considered as meta-specifications that provide 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 conver-

sations 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—discusses 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, and WS-Transfer.

The careful reader might have derived from the names of the single specifications how all WS-* efforts together are laying the foundation for 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 a commonly agreed on, proper programming environment for the envisioned SOP infrastructure are thus being made.

Coordination Middleware

Before going on and discussing the composition of Web services, it is worth noting that the coordination protocol specifications described so far are all so-called *description languages*. They are not executable languages to actively coordinate conversations among different Web services. The necessary runtime logic that adheres to the described protocol 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, for example, 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 pushed forward by WS-coordination could be exploited for message correlation purposes.

FROM COORDINATION TO COMPOSITION

We have noted that coordination protocols are characterized by an intrinsic passive behavior with respect to the execution of a coordinated interaction. However, despite such a passive behavior, coordination protocols have proven to have enough expressive power in the context of service *coordination*, which indeed does not require any executable logic. Yet, when it comes to *orchestration*, things change and active support for the execution of explicitly provided process or flow definitions is required. Process execution implies the need for dedicated execution environments, so-called execution or process engines, able to interpret process definitions and to control the flow of data and service invocations.

There are several different interpretations of what orchestration actually should be. Some authors refer to it as 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 Web services, their composition rules have to be specified at design time. Various structured process models have been proposed using traditional workflow constructs at their basis. A classification of typical workflow constructs that originate from a structured programming language approach to workflow definition and also can be found in today's service composition languages has been proposed by Van der Aalst, ter Hofstede, Kiepuszewski, and Barros (2003).

In the following subsections, we will provide insights into the most prominent 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 or application. 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 editor* (Kennedy, 2005). Internally, these models are then translated into low-level process models for execution purposes.

Several approaches for internal process models and structures have been proposed in literature. 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 process properties and correctness analyses. 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, which provide constructs for the specification of 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 management 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 (*business process execution language*; Weerawarana & Curbera, 2002) is an XML-based Web service composition language that has its roots 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: *abstract 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 presents both coordination as well as composition characteristics. Services participating in a process 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, etc.).

BPEL is primarily designed as a composition language, but developers can use the same formalism for both service composition and conversation definition. As for the definition of conversations, it however lacks some necessary and, from a discovery and binding perspective, particularly useful properties that would be required for defining conversations (e.g., for service activation). As for the composition of services, the structure of BPEL is flat (i.e., sub-processes cannot be defined).

BPML (*business process management language*; BPMI.org, 2002) provides similar modeling capabilities as BPEL, but also supports some additional constructs making it more flexible in general, such as sub-processes, etc. In particular, the BPML specification provides an abstract model and an XML syntax for expressing executable business processes. Nevertheless, 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, i.e., 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. Yet, there is still less industry support for BPML in comparison to BPEL, which is reflected by the higher availability of commercial tools for process definition and execution based on the BPEL specification.

Semantics-Based Composition

Model-based service compositions are explicit process modeling approaches in that the desired process flow needs to be explicitly provided (i.e., modeled) by composite application/service designers. The semantic Web and Web service ontologies offer alternative ways for the composition and execution of compound services, which do not rely on explicit definitions of the flow or process logic. Such approaches typically aim at providing suitable frameworks for the semantic description and the automatic discovery

and selection of Web services and the automatic derivation and execution of composite services defined in an implicit manner by means of goals and pre- and post-conditions over service inputs and outputs.

The recent W3C effort for the definition of *semantic annotations for WSDL* (SAWSDL) (Farrel & Lausen, 2006) aims at standardizing the first three of the previous concerns (i.e., semantic description, automatic discovery, and selection. SAWSDL does not prescribe any formalism for the specification of the semantics of a Web service, it rather concentrates on how to flexibly annotate a WSDL description with pointers to external semantic descriptions to disambiguate Web service descriptors during automatic discovery and composition. To enable semantic annotation of WSDL components, SAWSDL defines three new extensibility attributes to WSDL 2.0 elements, while remaining completely agnostic to the language used for the external semantic representation.

Concerning the semantics-based, automatic composition of Web services, Arpinar, Aleman-Meza, Zhang, and Maduko (2004) for example propose an ontology-driven Web services composition platform where the requirements of the desired composite services are specified by the user in form of provided inputs and expected outputs. The described approach allows the automatic generation and execution of a composite service that produces the expected outputs by combining existing individual services based on their semantic descriptions. A human-assisted and an automatic composition mechanism are outlined.

Two Emerging Standards: OWL-S vs. WSMO

OWL-S (*ontology Web language for Web services*; Martin, 2003) 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 Web 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), which provides the necessary constructs for explicitly representing knowledge, the meaning of terms and the relationships that exist among those terms 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 process. The third part, the service *grounding*, provides the necessary details to access a specific service (i.e., 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 (*Web service modeling ontology*; Roman, Lausen, & Keller, 2004) as well aims at describing relevant aspects of Web services in a semantically rich fashion. Within the *Web service modeling framework* (WSMF), WSMO provides an open, semantics-based formalism for goal-driven service composition through extensive use of ontologies, semantic service descriptions and

pre- and post-conditions for service descriptions. Besides ontologies, goals and service descriptions, so-called mediators allow the bypassing of interoperability problems among different services. Efficient interoperability is one of the main issues that WSMO tries to solve, differentiating it from OWL-S.

Just as for OWL-S, also in WSMO *ontologies* are adopted to provide the formal semantics that allows the automatic processing of information and the human- and computer-understandable goal definition. A *goal* specification expresses the final objective that 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, i.e., 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, 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.

In Roman and Scicluna (2006) the authors describe how choreography requirements of Web services can be specified in WSMO, so to express an individual service's communication behavior exposed to its clients. The description of the behavior is based on the abstract state machine model and defined in terms of one or more WSMO ontologies and a set of transition rules, leading to the notion of evolving ontology for the representation of the state of the choreography.

For the execution of WSMO-based Web services, DERI has developed the so-called *Web services modeling execution environment* (WSMX) (Haller, Cimpian, Mocan, Oren, & Bussler, 2005), a comprehensive execution environment for semantic Web services and DERI's reference implementation of WSMO. WSMX is designed to allow the dynamic discovery, invocation, and composition of Web services. It offers a complete support for interacting with semantic Web services and also supports the interaction with non-WSMO services. WSMX is made available as Web service that requires in input a formal description of the requester's goal and the data the requester wants to use for the invocation. Starting from these data and the single services' choreography requirements, WSMX takes care of all other computations, such as dynamic discovery, selection, and composition of the Web services that fulfill the requester's requirements.

Quality of Service-Based Approaches

Orthogonally to semantics-based approaches, which provide open and domain-independent means for service description, there are approaches that particularly focus their attention on *quality of service* (QoS) parameters for service selection and composition. Once the functional compatibility between candidate services is ascertained, service selection in QoS-based approaches is driven by quality properties like response time, accuracy (of results), completeness (of covered data), price, availability, reputation, or similar. Representations of QoS parameters in literature range from simple parameter-value pairs to complex QoS ontologies.

In Meteor-S, process composition is annotated with information for selecting services according to quality of service characteristics (Sivashanmugam, Miller, Sheth, & Verma, 2003). 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.

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 An-

tonellis et al., 2006; Maurino, Modafferi, Mussi, & Pernici, 2004). Service substitution can be performed to guarantee QoS constraints at runtime in a variable execution environment.

With QoSOnt (Dobson, Lock, & Sommerville, 2005) the authors aim to provide a common QoS conceptualization to be used by all actors involved in a QoS-based service selection (i.e., clients, providers, and third party intermediary systems). The QoS ontology QoSOnt is formalized in OWL and describes non-functional aspects of Web services, which may be used by clients to judge the services' quality. QoSOnt allows the specification of QoS attributes (e.g., reliability or performance), the metrics that are used to measure the values of attributes and possible conversions between different measuring units. The ontology is modular and extensible.

Other Composition Approaches

Several further (academic) research works go one step further in service composition and also investigate the potential of additional aspects of the composition problem, such personalization or context. We only cite two representatives of such work; the discussion of other valuable work would be out of scope for this chapter.

Maamar, Mostefaoui, and Yahyaoui (2005), for instance, 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 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 the coexistence of heterogeneous workflow systems 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 all the mentioned approaches, traditional composition patterns are enriched with additional features that allow flexible process specifications and executions. 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 optimisation of composed services are discussed in the literature.

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

A POSSIBLE PROTOCOL STACK

The previous sections have shown that research on service coordination and composition has led to a variety of different approaches and protocol or language specifications. Figure 4 describes a possible protocol stack as it could be adopted for the development of composite applications or services, starting from a set of individual component services. The protocol stack is horizontally split according to two dimensions (i.e., the perspective of the observer and the conceptual approach underlying the described specification). The per-

spective is divided into public and private, where the *public* perspective refers to choreography, and the *private* perspective refers to orchestration. The conceptual approaches are divided into coordination-based, execution-based, semantics-based, and quality-based approaches.

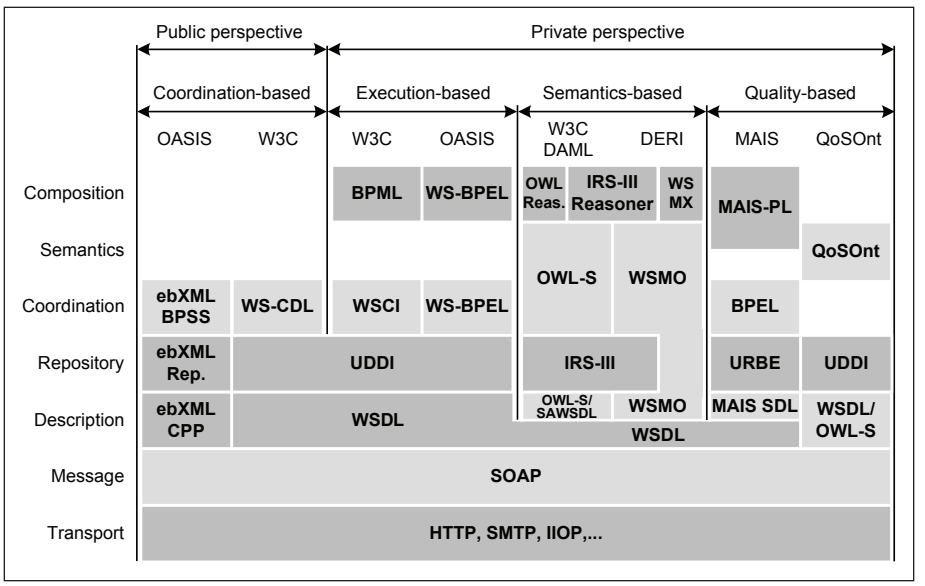
Interaction among services is based on traditional transport protocols such as HTTP, SMTP, or IIOP. The widely acknowledged basic message protocol is SOAP (nevertheless, other protocols could be used), and Web service description is primarily achieved by means of WSDL. But as can be seen in Figure 4 by moving along the vertical axis, when it comes to more advanced features, such as coordination and composition, the number of possible solutions grows, and the agreement becomes less.

In the following, we position the previously discussed approaches and technologies in the overall protocol stack and complete the resulting stack with some further specification, so as to provide an overview as complete as possible of the technologies that are at the basis of each approach. The proposed stack is not intended as exact picture of each single approach (some approaches indeed do not cover all aspects addressed in Figure 4), but rather represents reasonable configurations, as they could be adopted in a working system.

Coordination-Based Approaches:

- **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

Figure 4. 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.



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.
- **Repository/Registry:** The ebXML Registry is similar to UDDI in that it allows the discovery and binding of businesses, the definition of agreements between trading-partners, the exchange of XML messages in support of business operations. The goal is to allow all these activities to be performed automatically, without human intervention, over the Internet.
- **CPA (collaboration protocol agreement, not shown in Figure 4):** A CPA contains the business agreement among cooperating partners. It is derived from the intersection of the CPPs of the cooperating trading partners.
- **WS-CDL (Web services choreography definition language):** (W3C Working Draft; Kavantzas et al., 2005; Ross-Talbot et al., 2006). 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.

Execution-Based Approaches:

- **WSCI (Web services choreography interface):** (Initially Sun, SAP, BEA, and Intalio; now W3C Note; Arkin 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.
- 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 as well.
- **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 et al., 2002). It provides an XML-based grammar to describe 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.

Semantics-Based Approaches:

- **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 the definition of semantic descriptions as well as coordination rules. Previous releases of the 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); alternatively, service descriptions could also leverage SAWSDL. On top of OWL-S, proper reasoners allow automatic service composition and execution.

- **WSMO (Web service modeling ontology):** (DERI; Roman et al., 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 is grounded on WSDL, 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 on problem solving methods, using task descriptions in terms of input roles, output roles, pre-conditions, assumptions, and goals and ontologies. With the current version of IRS3, it is possible to execute WSMO services, but the binding of services occurs still at design time (Haller et al., 2005).
- **WSMX (Web service modeling execution environment):** (Haller et al., 2005). WSMX is the reference implementation of the WSMO execution environment developed by DERI International and allows the run-time discovery, selection, and composition of WSMO-based Web services. Discovery and selection are performed over a WSMO service repository, which is part of the WSMX implementation. WSMX internally adopts the WSML (Web service modelling language) for execution purposes.

Quality of Service-Based Approaches:

- **MAIS (multichannel adaptive information systems):** (Bianchini, De Antonellis, Pernici, & Plebani, 2006; Cappiello, Missier, Pernici, Plebani, & Batini, 2004; Maurino et al., 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 (Bianchini et al., 2006). Flexible 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.

- **QoSOnt:** (Dobson et al., 2005) provides means for semantically rich, QoS-based descriptions of Web services in OWL. The proposed approach adopts a standard UDDI registry for the discovery of service descriptions, which may be provided in either WSDL or OWL-S. QoSOnt is best used in combination with OWL-S; if service descriptions are provided in WSDL, OWL-S concepts cannot be referenced anymore, which slightly restricts the expressiveness of QoSOnt.

Based on Figure 4, one could say that composite service designers are confronted with a huge amount of partly mutually exclusive, partly dependent specifications in their composition task. Fortunately, they are not supposed to know and master all the above specifications together with their peculiarities. In fact, once they have chosen the composition or coordination approach that best matches their individual requirements, they only need to focus on those technologies and specifications that are necessary. Undoubtedly, the choice of the right approach is of crucial importance.

Besides due to real, functional needs, 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 proliferation of proprietary (or not) specifications and a lack of stability when it comes to choose reference specifications.

Figure 5. 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.

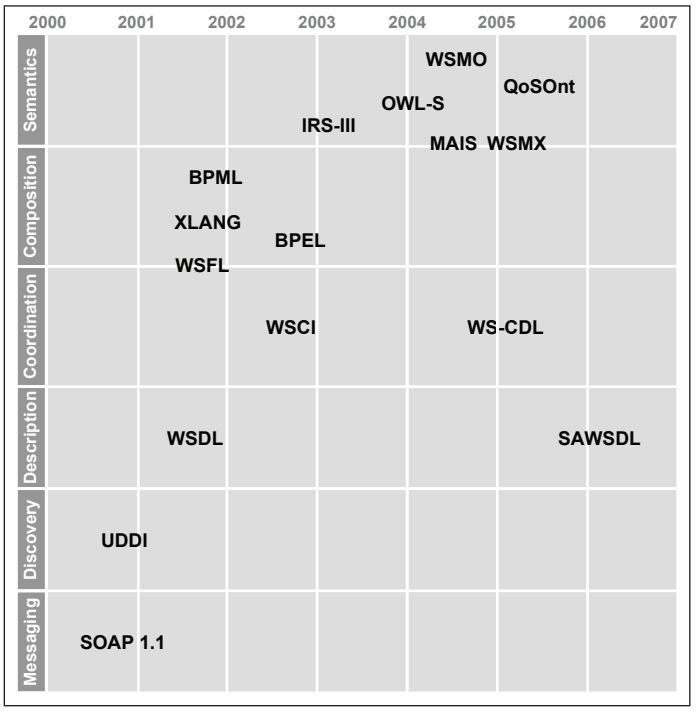


Figure 5 graphically depicts the temporal emergence of the previously listed standards and/or specifications. Along the diagram's diagonal, a trend toward high-level and semantically rich specifications can be derived (i.e., a trend toward enabling designers to comfortably specify or to automatically derive executable service compositions).

ADVANCED COMPOSITION PROBLEMS

The previous sections introduced some Web service coordination and composition solutions and characterized them by positioning them with respect to their driving conceptual approaches. Independently from the approach a developers chooses for his composite service or application, there are however a few typical, crosscutting composition problems that need to be addressed and which we did not yet cover. In this section we discuss a few of the most prominent ones of such problems (i.e., service selection, message correlation, transactions, and exception handling).

Service Selection

In one of the previous sections, we described two research efforts by Maamar et al. (2005) and Baïna et al. (2003). As the careful reader may have noticed, one of the main novelties introduced by these two research efforts (but also by some of the quality- or semantics-driven composition approaches) consists in the *dynamic selection* of the services to be composed, in addition to the dynamic service composition itself.

The purpose of dynamic service selection is mostly that of guaranteeing the availability and robustness of a composite service or application, being the Web a highly variable and fast changing environment. The question is whether component services are to be selected at process *definition time* or at r*untime*, during process execution; some authors distinguish between service selection at *design time* and *deployment time*. Service selection is probably the point where Web service orchestration approaches could learn from, but also add flexibility to traditional WfMSs, which typically 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.).

Currently, *static* (i.e., hard-coded within the process definition) selection approaches prevail over dynamic ones (Alonso et al., 2004). The URIs for locating the necessary services are typically 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 may be assigned to process variables and thus determined for example as result of a runtime operation call; this kind of service selection is 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 also require a certain level of middleware support.

Selection decisions are not only influenced by the selection time, but—and even at a higher degree—by the selection algorithm itself. As the ontology-driven approach shows, semantics- 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., 2006). 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. MAIS flexible process descriptions allow dynamic, context-aware selection, and binding at runtime.

Message Correlation

The next step after service selection is message correlation. For instance, there may be several concurrent instances of an individual service running in a specific execution environment (e.g., a service container) and engaged in different conversations with other services. Message correlation deals with the unique identification of such instances and the conversations they are involved in with external Web services in order to guarantee the overall, correct execution of the separate processes that are running.

As already seen earlier, 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 that implements the described mechanisms.

On the other hand, BPEL already proposes a solution at process level, namely so-called *correlation sets* that—similar to WSCI—allow the definition of sets of data items as unique identifiers. By assigning the same correlation set to multiple messages, the composition 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

Composite Web services and applications aim to support collaborations between business partners; such collaborations typically require robust transaction support. The classical ACID properties of relational databases have proven being too strict in a service-oriented environment involving several autonomous business partners. Thus, in the context of Web services, some of the ACID properties need to be slightly relaxed. Furthermore, proper compensation mechanisms need be taken into consideration, as already done 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 their 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 (e.g., by Java and currently implemented in BPEL), or *flow-based* mechanisms that consist in explicitly modeling the error checking logic within the proper process description. Also, *rule-based* approaches exist, which are particularly suited for handling temporal exceptions.

HOW ORCHESTRATION DEPENDS ON CHOREOGRAPHY

Choreography and orchestration represent two different conceptual interpretations of the col-

laboration problem, but the two ideas are far from being independent the one from the other. In this section, we therefore briefly highlight to what extent orchestration depends on choreography by concentrating three main dimensions (i.e., structural, functional, and resource dependencies).

Structural Dependencies

Structural dependencies drive the overall structure and organization of a process definition and thus concern activities, conditions and routing decisions in the process specification.

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 services participating in a coordinated conversation. Starting from an overall activity diagram of the process, the authors first extract the role-specific view of the process (the one of the chosen service) and then refine it in order to reach a granularity level where the single activities of the remaining diagram reflect the single service invocations required for achieving the role-specific functionality. This so-called process skeleton on the one hand describes the role-specific view of the process; on the other hand, it 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, etc. may lead 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 at the process specification level, and the composition language "delegates" the relative competencies to the underlying coordination protocols.

As already exemplified earlier, coordination can for example 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. To execute 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 to improve 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. In fact, a Web service may by subject to individual conversation rules in order to be executed correctly.

For example, before accepting a user's credit card number for payment, a service typically must be provided with the 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 in order to 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 imposed by the selected services (bottom-up). Therefore, the resulting process definition combines a coarse-grained top-down approach with a fine-grained bottom-up approach.

FUTURE TRENDS

In light of the developments and the evolutions achieved so far in the Web services area, one is inclined to ask what will happen next to orchestration and choreography of Web services and, thus, to processes on the Web. In the following, we provide our personal ideas about some of the most interesting questions.

Coordination or Composition?

In the previous sections, 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. 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.

The difference between coordination and composition in fact cannot just boil down to mere technical considerations; legal aspects also play an important role while orchestrated interactions have one central entity in charge of guaranteeing the correctness of the interaction, choreographed interactions do not. In the former case, there is one partner who has a higher responsibility concerning the success of the cooperation, while in the latter case each partner has the same responsibility.

Trends in Private Process Descriptions

As hinted at in the introduction when comparing SOC 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 the languages and specifications developed in the context of SOC, rather than in the languages or specifications 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 multi-attribute dynamic service selections within a composite service and peer-to-peer orchestrations. 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 provision 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 toward 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, and Plebani (2004).

From Web Services to Grid Services

Recently, also researchers from the field of grid computing have started to investigate the potential of orchestration and composition technologies, stemming from the world of Web services, for the distributed execution and management of complex processes on the grid. Grid computing is an emerging computing model that leverages a multitude of networked computers to model a virtual computer architecture that is able to distribute process executions across a parallel infrastructure. Especially in the context of large-scale scientific computations, such as genetic analyses, geological investigations or weather predictions, grid computing already provides promising results.

In Emmerich, Butchart, Chen, Wassermann, and Price (2006), the authors concentrate for example on grid services, that is, Web services that are deployed and executed in service-oriented grid computing infrastructures. By means of a real world scientific workflow problem, the authors show how BPEL can successfully be adopted to orchestrate complex, scientific workflows in grid systems, despite the typically huge number of activities that compose a scientific process (e.g., several thousands of work items!). Also in Fox and Gannon (2006), workflows of tens of thousands of participating entities or activities are described, and especially the role of robust exception handling mechanisms (as for example the one provided by BPEL) is highlighted. Scalability and robustness of orchestration and composition solutions are key ingredients for the success of Web services in grid computing.

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), Vinoski 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).

ACKNOWLEDGMENT

This work has been partially supported by the European FET-STREP project WS-Diamond and the Italian PRIN 2005 project Quadrantis.

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., Pogliani, S., Riemer, K., Struble, S., Takacsi-Nagy, P., Trickovic, I., & Zimek, S. (2002). *Web service choreography interface (WSCI) 1.0. W3C Note*, August 2002. 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. *Proceedings of the IEEE International Conference on E-Commerce Technology*, IEEE.

Baïna, K., Benali, K., & Godart, C. (2003). Dynamic interconnection of heterogeneous workflow processes through services. The *11th International Conference on Cooperative Information Systems* (CoopIS'03), In *Confederated International Conferences* (DOA/CoopIS/ODBASE'03), (LNCS) 2888, Catania, Sicily, Italy, November 3-7, 2003. 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. (2006). Ontology-based methodology for e-service discovery. *Information Systems*, *31*(4-5), 361-380.

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, Business Process Management Initiative. (n.d.). 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. *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. *Information Technology and Management, 5*, 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). Orchestration of semantic Web services in IRS-III. In *Proceedings of the 1st AKT Workshop on Semantic Web Services* (AKT-SWS04) KMi, The Open University, Milton Keynes, UK, December 8, 2004.

Daniel, F., & Pernici, B. (2006). Insights into Web service orchestration and choreography. *International Journal of E-Business Research*, *2*(1), 58-77, Idea Group Publishing, January-March 2006.

De Antonellis, V., Melchiori, M., De Santis, L., Mecella, M., Mussi, E., Pernici, B., & Plebani, P. (2006). A layered architecture for flexible e-service invocation. *Software Practice & Experience*, *36*(2), 191-223.

Dobson, G., Lock, R., & Sommerville, I. (2005). QoSOnt: An ontology for QoS in service centric systems. *Proceedings of the eScience All Hands Meeting*, Nottingham, September 2005.

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

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

Emmerich, W., Butchart, B., Chen, L., Wassermann, B., & Price, S. L. (2006). Grid service orchestration using the business process execution language (BPEL). *Journal of Grid Computing*, *3*(3-4), 283-304.

Farrell, J., & Lausen, H. (2006). *Semantic annotations for WSDL*. W3C Working Draft, September 2006. Retrieved January 2007, from http://www.w3.org/TR/sawsdl/

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

Fox, G., & Gannon, D. (2006). Workflow in grid systems. *Concurrency and Computation: Practice & Experience*, *18*(10), 1009-1019, August 2006.

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. *Proceedings of the 1st AKT Workshop on Semantic Web Services (AKT-SWS04)*; KMi, The Open University, Milton Keynes, UK.

Haller, A., Cimpian, E., Mocan, A., Oren, E., & Bussler, C. (2005). WSMX—A semantic service-oriented architecture. *Proceedings of the International Conference on Web Service* (ICWS 2005). Orlando, Florida, 2005.

Hewlett-Packard Company. (2002). *Web services conversation language (WSCL) 1.*0. *W3C Note*,

March 2002. 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, Jan-Feb 2004.

Kavantzas, N., Burdett, D., Ritzinger, G., Fletcher, T., Lafon, Y., & Barret, C. (2005). *Web services choreography description language version 1.0*. W3C Candidate Recommendation, 9 November 2005. Retrieved January 2007, from http://www.w3.org/TR/ws-cdl-10/

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

Khalaf, R., & Nagy, W. A. (2003). *Business process with BPEL4WS: Understanding BPEL4WS, Part 7, Adding correlation and fault handling to a process*. Research report, IBM developerWorks, April 2003. 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, Nov. 2004.

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, May 2005.

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. *WWW (Alternate Track Papers & Posters) 2004*, New York City, NY, pp.48-54.

Martin, D. (2003). *The OWL services coalition. OWL-S: Semantic markup for Web services*. White Paper. 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. *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 Nov.-Dec. 2004.

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

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

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

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

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

Roman, D., & Scicluna, J. (2006). *Ontology-based choreography of WSMO services*. WSMO Final Draft, May 2006. Retrieved January 2007, from http://www.wsmo.org/TR/d14/v0.3/

RosettaNet. (2006). Retrieved December 2006, from http://www.rosettanet.org

Ross-Talbot, S., & Fletcher, T. (2006). *Web services choreography description language: Primer*. W3C Working Draft, June 2006. Retrieved January 2007, from http://www.w3.org/TR/ws-cdl-10-primer/

Sivashanmugam, K., Miller, J., Sheth, A., & Verma, K. (2004). Framework for semantic Web process composition. *Special Issue of the International Journal of Electronic Commerce (IJEC)*, Eds: Christoph Bussler, Dieter Fensel, Norman Sadeh, Feb 2004.

Smith, M. K., Welty, C., & McGuinness, D. L. (2004). *OWL Web ontology language guide*. W3C Recommendation, February 2004. 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). *CAiSE 2003,* 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, July 2003.

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

Weerawarana, S., & Curbera, F. (2002). *Business process with BPEL4WS: Understanding BPEL-4WS, Part 1, Concepts in business processes*. Research report, IBM developerWorks, Aug. 2002. Retrieved January 2005, from http://www-106.ibm.com/developerworks/webservices/library/ws-bpelcol1/

WfMC - Workflow Management Coalition (n.d.). Retrieved January, 2007, from http://www.wfmc.org

W3C - World Wide Web Consortium (n.d.). Retrieved January, 2007, from http://www.w3.org

W3C. (2001). *Web services description language (WSDL) 1.1. W3C Note*, March 2001, http://www.w3.org/TR/wsdl

xCBL.org. (2006). *XML common business library*. Retrieved December, 2006, from http://www.xcbl.org

ENDNOTES

[1] The present work is a revision, extension and update of the survey work published by the authors in January 2006 in Daniel and Pernici (2006).