Comparing WSDL-Based and ebXML-Based Approaches for B2B Protocol Specification

Martin Bernauer, Gerti Kappel, and Gerhard Kramler

Business Informatics Group, Vienna University of Technology, Austria
{lastname}@big.tuwien.ac.at

Abstract. When automating business processes spanning organizational boundaries, it is required to explicitly specify the interfaces of the cooperating software systems in order to achieve the desired properties of interoperability and loose coupling. So-called B2B protocols provide for the formal specification of relevant aspects of an interface, ranging from document types to transactions. Currently, there are two main approaches proposed for the specification of B2B protocols, the WSDL-based approach supporting Web Service languages, and the ebXML-based approach supporting languages defined along the ebXML project. Unfortunately, these approaches are not compatible, thus an organization wanting to engage in B2B collaboration needs to decide whether to embark on any of these new approaches, and which ones to use. This paper introduces a conceptual framework for B2B protocols, and based on this framework, a methodical comparison of the two approaches is provided, answering the questions of what the differences are and whether there are chances to achieve interoperability.

1 Introduction

The automation of business processes spanning organizational boundaries has potential. First steps towards this goal have proven highly successful, namely the use of email for communication between human agents, and the use of web applications for communication between humans and business applications published to the extranet or internet. The complete automation of business processes, however, still suffers from high implementation cost. Basically, the additional complexity is that the business applications of cooperating organizations cannot be developed independently of each other but need to be interoperable.

Interoperability of cooperating business applications which provides automation requires explicit specification of requirements and constraints on the business application's interfaces. Such specifications are referred to as B2B protocols [3], business protocols [16], public workflows [23], or conversation processes [5]. It has to be emphasized that the specification of a B2B protocol should be separated from the specifications of workflows within organizations that support the protocol, in order to facilitate loose coupling and design autonomy of the intra-organizational workflows [4,6].

M.E. Orlowska et al. (Eds.): ICSOC 2003, LNCS 2910, pp. 225–240, 2003.

A B2B protocol defines various aspects of an interface, such as the transport protocol, document types, security requirements, and transactional properties, to mention just a few. An organization playing a certain role in a collaboration has to support the protocol specifications in order to guarantee interoperability. If a protocol specification does not cover certain aspects, these have to be agreed on out of band by organizations willing to cooperate in order to achieve interoperability of their applications. Examples of widely used B2B protocols are EDIFACT, which covers only document types, and RosettaNet, which covers all of the above mentioned aspects.

Defining protocol specifications by means of formal languages – such as W3C's XML Schema for the specification of document types – is beneficial in various respects. First, it enables tool support for development tasks such as consistency checks, development of data transformations, and customization of predefined protocols in a controlled manner, thus easing the adoption of a B2B protocol by an organization. Second, interpretation of the specification allows for a generic, re-useable implementation of functions such as schema validation, messaging, and security management. Finally, formal specifications which are published on the web or in specialized repositories provide the basis for automated dynamic discovery and integration with any organization supporting a matching B2B protocol.

Currently, there are two main technologies proposed for the specification of B2B protocols. Most prominently, the Web Services idea [12] subsumes a set of specification languages, with WSDL as its core and several proposed extensions, such as BPEL4WS for the specification of behavioral aspects. Although the intended application domain of Web Services is not limited to B2B protocols, B2B is considered the most prominent one. In parallel to Web Services, the ebXML initiative[1] has developed a set of standards specifically targeted at the specification of B2B protocols. Vendor support for ebXML, however, is not as strong as for Web Services. Furthermore the ebXML-based and the WSDL-based approaches are not compatible, thus an organization wanting to engage in B2B collaboration needs to decide whether to embark on any of these new technologies, and which ones to use.

There have already been efforts in comparing ebXML and Web Services. In [3], Bussler identifies the required elements of a B2B protocol, and classifies various B2B standards using the categories "business event", "syndication", and "supporting". Languages for the specification of (aspects of) B2B protocols, such as ebXML and WSDL, are identified as supporting standards. No further evaluation of the classified B2B standards concerning the required protocol elements is provided. In [22], van der Aalst uses a comprehensive set of control flow and interaction patterns to evaluate the features of several languages proposed for the specification of processes, including BPEL4WS and workflow management products. The evaluation is focused on the control flow aspect, and does not include languages specific for B2B protocol specification. It is concluded that languages proposed by software vendors are often influenced by that vendors' product inter-

[1] `http://www.ebxml.org/`

ests, neglecting the real problems. In [18], Shapiro presents a detailed comparison of BPEL4WS, XPDL, the WfMCs proposed standard for XML-based workflow specification languages, and BPML, a language similar in scope with BPEL. The comparison focuses on the specification of executable workflows and leaves out the concepts provided by BPEL4WS and BPML for protocol specification. Similarly, our previous work [2] provides an overview of various process specification languages including WSDL-based and ebXML-based ones, but without making a clear distinction between protocol specification and implementation specification, and without specifically highlighting the differences between WSDL-based and ebXML-based approaches. The relationship of Web Services and ebXML has also been discussed in various magazine articles. For example, [10] argues that ebXML is advantageous in typical "regulated" B2B scenarios, whereas Web Services are considered adequate for more loose collaborations without formal commitments.

This paper intends to provide further insight by presenting a methodical comparison of languages focusing on protocol specification based on a framework for the classification of protocol layers and aspects. Specifically, we aim to answer the questions of what the actual differences are between WSDL-based and ebXML-based languages, and whether there are chances to achieve interoperability. In the following section, we present the framework used to guide the comparison. Section 3 gives a short overview of the languages and their relationship to our framework. The detailed comparison is given in Section 4. Section 5 concludes with a summary of the comparison and an outlook to future research.

2 Framework for Comparison

To describe and compare the capabilities of the two approaches, this section introduces a conceptual framework to provide for common terms. The framework is based on the eCo Framework [7], which provides for a general description of e-commerce systems, and on the workflow model proposed in [17], which provides for a more specific description of workflow systems.

First, the conceptual framework is based on the eCo Framework. The latter is a layered model and can be used by businesses to define and publish descriptions about their e-commerce systems. It defines seven layers, whereby the upper three layers (i.e., the "networks", "markets" and "business" layer) are not relevant for the description of a B2B protocol. The fourth layer, the *services layer*, is used to describe services by their interfaces which are provided and used by businesses. A service may be composed of sub-services and may invoke other services. These interactions between services are described at the *interactions layer*. It describes the types of interactions behind each service, and the types of messages which are exchanged during each interaction. A message type may contain several document types, which are described at the *documents layer*. Finally, the *information items layer* describes the types of information items that may be used in document types.

Each layer of the eCo Framework provides for the layer above and builds on the layer beneath. The layered architecture implies that an artefact defined at one layer is independent of any layer above. For example, a document type is defined independent of interactions or services it is used in. Thus it can be reused across interactions and services.

Second, the conceptual framework is based on the workflow model proposed by Rausch-Schott [17], which describes several aspects of workflows that workflow descriptions have to cope with. While the *functional aspect* specifies what is to be executed, i.e., the semantics of a function provided by a workflow, the *operational aspect* defines how the function is implemented. The *behavioral aspect* describes how functions can be composed, e.g., as a sequence or alternative. Concentrating on data, the *informational aspect* describes data structures and data flow between functions. The *organizational aspect* describes personal and technical resources. The *transactional aspect* deals with consistency, i.e., how transactions can be used to guarantee consistent execution of functions or whole workflows. The *causal aspect* defines why a certain B2B protocol is specified in a certain way and why it is being executed. And finally, the *historical aspect* defines which data should be logged at which point in time.

While Rausch-Schott's model is intended to describe workflows that execute within a single business, all but one aspect apply for B2B protocols as well and can thus be leveraged for their characterization. The historical aspect cannot be leveraged because it describes aspects that each participating business is responsible for separately. Since B2B protocols cross business boundaries in contrast to traditional workflows, it is necessary to introduce an additional *security aspect*. It describes confidentiality, non-repudiation, integrity, authorization, and authentication.

Table 1. Supported combinations of eCo layers and workflow aspects

	func.	org.	info.	behav.	secur.	trans.	causal	oper.
services	X	X	X	X	-	X	-	-
interactions	X	-	X	X	X	X	-	X
documents	-	-	X	-	-	-	-	-
info. items	-	-	X	-	-	-	X	-

The conceptual framework uses the layers of the eCo framework as a classification of requirements on a B2B protocol specification language, whereby each layer is refined by relevant workflow aspects. Considering relevance of combinations of aspects and layers, we consider only combinations of aspects and layers that are supported by the approaches. The respective meaning of each of these combinations has been derived from the idiosyncrasies of the ebXML-based and WSDL-based approaches and will be described along the layer-by-layer comparison of approaches in Section 4. Table 1 summarizes the supported combinations of layers and aspects.

3 Overview of Approaches

Each of the two approaches employs a set of specific languages for the specification of different parts of a B2B protocol. The languages employed in the WSDL-based approach have been selected from the various proposals made in the Web Services area. Since Web Service languages are developed by software vendors in loose cooperation, different options are available for certain specification tasks. For the purpose of the comparison, we have included those languages which we consider as having the broadest support among software vendors. The languages employed in the ebXML-based approach are those developed along the ebXML project and following efforts. The comparison is based on the most recent language specifications. This section introduces the languages employed by either approach, and how these languages relate to each other and to our conceptual framework (cf. Figure 1).

The WSDL-based approach employs XML Schema (cf. [25,26]) for the specification of information items. The documents layer is not supported. Interaction types are specified using WSDL (Web Service Description Language, cf. [24]) in combination with WSSP (Web Services Security Policy, cf. [9]), whereby WSSP complements WSDL in that it focuses on the security aspect. It should be noted that WSSP is in an initial public draft state, which exhibits inconsistencies. Nevertheless, it has been included in this comparison because it is the only option available for specifying the security aspect. Service types are specified using BPEL (Business Process Execution Language for Web Services, cf. [1]). Note that WSDL also supports specification of service types, but WSDL's concept of service type refers to software components, whereas BPEL specifies service types from a business case point of view, which is also the view taken in this paper.

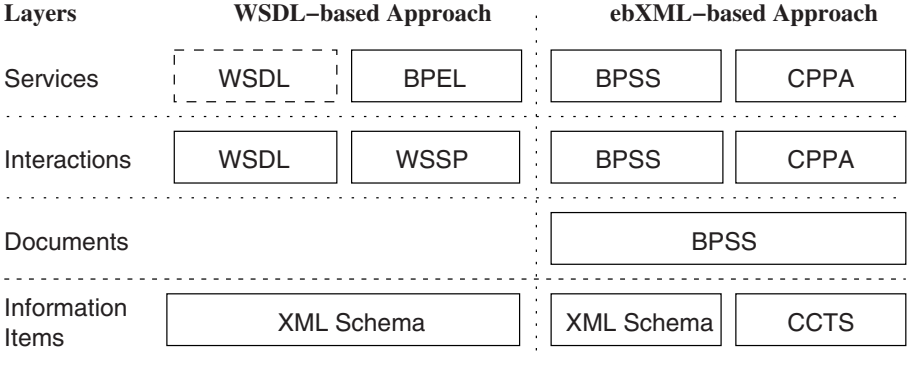

Fig. 1. Layers of the conceptual framework and supporting languages

The ebXML-based approach also employs XML Schema for the specification of information items. Furthermore, CCTS (Core Components Technical Specifi-

cation, cf. [20]) defines a methodology and language for identification of information items, which can be used in the process of defining information items. Document types are specified using BPSS (Business Process Specification Schema, cf. [21]). Interaction types are specified in terms of BPSS and CPPA (Collaboration-Protocol Profile and Agreement Specification, cf. [13]). BPSS provides for the technology- and business-independent aspects, whereas CPPA is used to supplement technology and business details. In particular, CPPA can be used to overwrite certain properties of interaction types as defined with BPSS in order to adapt them to the needs of a specific business. Service types are specified using also BPSS and CPPA. Similar to the interaction layer, CPPA can be used to adapt a service type to a specific business.

4 Comparison

The comparison is performed along the layers of the eCo model. Beginning at the base layer, layer by layer and aspect by aspect we will detail the conceptual framework and analyze and compare the two approaches. The approaches are described in terms of our conceptual framework, with links to the keywords of the specific languages to provide for a better understanding. Keywords are denoted in sans serif font using the above introduced language-specific acronyms as namespace prefix (e.g., wsdl:message). Note that this comparison is performed at the level of language concepts, for an example specification expressed using both approaches it is referred to [11].

4.1 Information Items Layer

The Information Items layer specifies re-useable data types, such as address, product code, and price, independent of their use in particular documents. Aspects of workflow modeling supported in the specification of information items are the informational and the causal aspects.

Both the WSDL-based and the ebXML-based approach support XML Schema as the preferred language for the specification of information items. We will briefly review XML Schema in the light of our conceptual framework.

The Informational Aspect is concerned with the structure and semantics of information items, including refinement of information items, composition of information items, and various constraints such as cardinality.

In this respect, XML Schema provides built-in datatypes and structuring mechanisms. XML Schema's built-in datatypes are very generic and do not provide semantics specific to the needs of B2B applications. It is therefore necessary to define more specific ones. Recently, standardization efforts in the B2B area, such as OAG[2] and UBL[3], have begun to support XML Schema for the specifica-

[2] http://www.openapplications.org/

[3] http://www.oasis-open.org/committees/ubl/

tion of information items, thus such standard information items can be directly employed in a WSDL-based or ebXML-based B2B protocol specification.

The Causal Aspect is concerned with the reasons behind the design of information items, i.e., the identification of influence factors such as the requirements of a specific industry or a certain country.

XML Schema does not support the causal aspect of information item specification. However, CCTS, a part of the ebXML project, addresses this aspect. CCTS defines a methodology and language for the identification and specification of so-called core components, i.e., generic information items which are independent of any particular business context such as business process, industry, and official constraints, and thus widely reusable. To make core components usable in a specific application context, they are adapted by means of restrictions and/or extensions in order to incorporate the specific requirements. As the semantics of a specific information item can be derived from the semantics of the core component it is based upon, semantic expressiveness and interoperability is improved. CCTS does not specify a concrete schema language for core components and information items, which makes the methodology applicable to different technologies such as EDI and ebXML. Unfortunately, there is no standardized way to transform core components to XML Schema. As a possible solution to this problem, the UBL effort creates schemas in XML Schema for core components which have been derived from existing standard document types. UBL schemas can be directly used in both ebXML-based and WSDL-based protocol specifications.

4.2 Documents Layer

Documents are containers for information items and are used to carry information in the workflow within and across businesses. Only the informational aspect is supported in the specification of document types.

In the WSDL-based approach, document types are not supported at all, meaning that message types are defined directly based on information items. In the ebXML-based approach, there is some support for document types addressing the informational aspect. A document type is specified in terms of a name and the information item contained in the document (bpss:BusinessDocument). The information item may be further restricted allowing for application-specific restrictions of standard information items, e.g., the status value must be "accept" (bpss:ConditionExpression).

4.3 Interactions Layer

An interaction is a basic exchange of messages between business partners having a certain effect. Interaction types are specified in a declarative way by means of predefined constructs supporting common interaction patterns. A typical interaction pattern is request/response, e.g., one partner sends a purchase order request, and the other responds with an acknowledgement, meaning that the order

is in effect. Another typical pattern is the oneway interaction, e.g., one business partner sends a shipment notification. Several workflow aspects are supported in the specification of an interaction type, namely the functional, informational, behavioral, security, transactional, and operational aspects.

In the WSDL-based approach, interactions have the semantics of remote procedure calls (wsdl:operation), i.e., the initiating partner requests some function and the responding partner performs that function and returns the result. Two kinds of interaction types are supported, namely request/response and oneway.

In the ebXML-based approach, the guiding principle behind interactions is the so-called business transaction (bpss:BusinessTransaction), i.e., a request/response kind of interaction which may create or resolve a commitment between the two involved partners. Specifically, the ebXML-based approach adheres to the metamodel for business transactions defined by UMM (UN/CEFACT Modeling Methodology, cf. [19]). UMM also defines a set of so-called analysis patterns for business transactions such as Commercial Transaction and Query/Response, which can be directly used in ebXML.

The Functional Aspect defines the intention of an interaction, i.e., its goal.

In the WSDL-based approach, the functionality of an interaction is specified only in terms of a name and whether the interaction delivers a result (request/response pattern) or not (oneway pattern).

In the ebXML-based approach, an interaction is specified in terms of a name, informal pre- and postconditions, and whether it delivers a result. Furthermore, an interaction is decomposed into the functionality at the initiating role (bpss:RequestingBusinessActivity) and at the responding role (bpss:RespondingBusinessActivity), each of which is specified by a name.

The Informational Aspect refers to the messages exchanged during an interaction. Messages are defined by a message type, which in turn specifies the documents to be included in the message.

In the WSDL-based approach, oneway interaction types comprise only one message (wsdl:input), whereas request/response interaction types comprise a request message (wsdl:input), and a number of alternative response messages (wsdl:output or one out of a set of named wsdl:fault messages). Message types are named reusable entities (wsdl:message), defined in terms of a list of named information items (wsdl:part).

In the ebXML-based approach, interaction types comprise one requesting message and optionally a number of alternative responding messages. A message type (bpss:DocumentEnvelope) is defined by a name, one primary document, which is defined by a document type, and additionally any number of named attachments, which can be defined either by a document type, by a MIME type, or left unspecified.

The Behavioral Aspect addresses the control flow during an interaction in terms of ordering message exchanges and of defining initiating and responding

roles. Furthermore, timing and exceptions need to be considered. Typically, the behavior, i.e., the control flow of interactions is predefined and only limited means of customization are possible.

In the WSDL-based approach, the behavior of the oneway interaction type is asynchronous, i.e., the initiator sends a message to the receiver. Neither timing nor exceptions are relevant at this level of abstraction. The request/response interaction type is synchronous, i.e., the initiator sends a request message to the responder, who responds after processing the message with either a normal response or an exception message. It is not possible to specify any timing parameters.

In the ebXML-based approach, the behavior of interactions follows the UMM metamodel for business transactions, which defines the control flow as an enhanced request/response model. Basically, the initiator sends a message to the responder, the responder processes the request, and optionally sends back a response message thereby indicating success or failure of the interaction (bpss:isPositiveResponse). This basic model is enhanced with optional acknowledgement signals indicating receipt of the request and response messages, respectively, or indicating acceptance of the request message. Signals indicate either success or exceptional termination. A receipt acknowledgement may inform about successful schema validation (bpss:isIntelligibleCheckRequired), an acceptance acknowledgement informs about some further validation of the request message's content. Timeout values can be specified for both signalling and request processing (bpss:timeToPerform).

The Security Aspect addresses security properties of interactions, namely integrity, authenticity, and confidentiality of messages, as well as authorization and non-repudiation.

In the WSDL-based approach, only integrity, authenticity, and confidentiality of individual messages are addressed. In particular, a message type can have an attached security policy (wssp:Policy), which can specify integrity and authenticity (wssp:Integrity) and confidentiality (wssp:Confidentiality) requirements of selected parts of a message. Selecting parts of a message is done using XPath, which is very expressive but requires knowledge about the SOAP message format and processing model.

In the ebXML-based approach, the following security requirements can be specified for each document which is part of a message: integrity (bpss:isTamperProof), authenticity (bpss:isAuthenticated), and confidentiality (bpss:isConfidential). Authorization (bpss:isAuthorizationRequired) and non-repudiation (bpss:isNonRepudiationRequired and bpss:isNonRepudiationOfReceiptRequired, respectively) can be specified for the initiating role and the responding role.

The Transactional Aspect considers transactional properties of an interaction, such as atomicity and consistency, which are of particular importance in a distributed system without central control.

In the WSDL-based approach, transactional properties of an interaction cannot be specified explicitly. Although transaction support is addressed by several proposed protocols such as WS-Transaction[4] and BTP[5], the means for including them in a WSDL-based protocol specification have yet to be defined.

In the ebXML-based approach, at least the atomicity property of transactions is supported in that each interaction is considered atomic, meaning that an interaction has a defined end and in that both parties have a consistent knowledge of whether it has succeeded or failed. If it has failed, it doesn't create any commitments. It has to be mentioned that the behavior of an interaction is not sufficient to guarantee a consistent understanding about an interaction's final state, therefore a separate interaction may be necessary to notify the responder about a failure at the initiator. A detailed analysis of the differences between the ebXML behavior and two-phase distributed transaction protocols such as BTP can be found in [8]. Besides atomicity, it can be specified that an interaction must be conducted using a reliable means of communication (bpss:isGuaranteedDeliveryRequired), essentially regarding message exchanges as sub-transactions of an interaction.

The Operational Aspect considers how interactions are performed, i.e., the particular implementation-level protocols to be used for message transport and encoding, security, and transaction coordination. While in either approach some of these decisions are fix, some options can be selected in the protocol specification.

In the WSDL-based approach, several options for message transport and message encoding are available (wsdl:binding), e.g., SOAP over HTTP. Message security is realized according to WS-Security[6].

The ebXML-based approach defines its own messaging protocol [14]. It builds on SOAP and provides extensions addressing reliable messaging, message security, and others. The underlying transport protocols and message encoding can be flexibly defined, supporting both synchronous and asynchronous bindings. Message security is realized using S/MIME and XML Signature. Non-repudiation and transaction coordination are realized on top of the ebXML interaction behavior utilizing the the request/response messages and the corresponding acknowledgement signals.

4.4 Services Layer

A service is the work done by a business (the service provider) that benefits another (the service consumer). For the specification of a service type, the supported workflow aspects are the functional, organizational, behavioral, informational, and transactional ones, as discussed below.

[4] http://www.ibm.com/developerworks/library/ws-transpec/

[5] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=business-transaction

[6] http://www-106.ibm.com/developerworks/webservices/library/ws-secure/

In the WSDL-based approach, service types are *unilateral*, i.e., the service functionality, behavior, etc. are specified from the service provider's point of view. In particular, service types are specified using BPEL, which builds upon WSDL interaction types and provides for the specification of so-called abstract processes (bpel:process). BPEL also provides concepts for the specification of so-called executable processes, which define the workflow realizing a service type, however, these are out of scope of a B2B protocol.

In the ebXML-based approach, two kinds of service types are distinguished. *Bilateral* service types are restricted to exactly two roles, i.e., service provider and service consumer (bpss:BinaryCollaboration). *Multilateral* service types involving many roles can be specified as a composition of bilateral service types (bpss:MultiPartyCollaboration). Each of these two kinds of service types address all but the informational workflow aspect.

The Functional Aspect is concerned with the work provided by the service type and its functional decomposition. Regarding decomposition, a service type can be decomposed into sub-services and ultimately into interactions as defined in the interactions layer.

In the WSDL-based approach, the functionality of a service type is specified in terms of a name and the decomposition into interactions (bpel:invoke and bpel:receive/bpel:reply for used and provided functionality, respectively). Through appropriate combination of these constructs, execution dependencies between interactions can be defined in flexible ways.

In the ebXML-based approach, bilateral service types are specified in terms of a name, informal pre- and postconditions, and the decomposition into bilateral sub-services (bpss:CollaborationActivity) and interactions (bpss:BusinessTransactionActivity). Multilateral service types are specified in terms of a name and the decomposition into bilateral sub-services. Execution dependencies between interactions can be defined in both service types, in particular nesting of interactions is supported (bpss:onInitiation).

The Organizational Aspect addresses the roles of businesses involved in a service type, and the authorizations and obligations of each role. Furthermore, a role's agent selection policy defines how a particular business playing that role is identified in an actual service instance.

In the WSDL-based approach, a service type specifies one primary role (bpel:process) and a number of secondary roles (bpel:partner). Only the primary role's functional, behavioral, and informational properties can be specified explicitly, whereas the corresponding properties of secondary roles are left undefined except for the compatibility requirements imposed by their relationship with the primary role. Agent selection policies are supported by means of a specific data type (bpel:serviceReference) which can be used in conjunction with a data flow specification (bpel:assign) to define possible businesses playing a certain role, allowing to determine it dynamically at run time.

In the ebXML-based approach, bilateral service types define two roles (bpss:-Role), which are associated with the initiating and responding roles of the interactions that constitute the service (bpss:fromRole and bpss:toRole, respectively), thereby implying the authorizations and obligations of each of the two roles in terms of functional, behavioral, informational, and transactional aspects. Multilateral service types define multiple roles (bpss:BusinessPartnerRole), whereby the relationship between each pair of roles is defined in terms of a bilateral subservice. A multilateral service type can furthermore specify the coordination obligations of a role which is involved in multiple bilateral sub-services using nesting of interactions (see functional aspect). Agent selection policies are not supported in the ebXML-based approach.

The Informational Aspect is concerned with protocol relevant data used in a service type, which is defined by variables, their data types, the data flow, and message correlation, i.e., the association of messages to service instances.

In the WSDL-based approach, data used in a service type is defined local to the primary role, in terms of variables (bpel:variable), data types (wsdl:message), the data flow between variables (bpel:assign), and the data flow between variables and interactions (bpel:inputVariable and bpel:outputVariable). Protocol relevant data is explicitly identified using XPath expressions applied to the contents of variables (bpel:property). The data flow of protocol relevant data must be completely specified, whereas the flow of other application data can be specified only partially. Message correlation is defined based on a subset of the protocol relevant data which identifies a service instance in the context of an interaction (bpel:correlationSet).

In the ebXML-based approach neither variables nor data flow can be specified. Furthermore, message correlation does not need to be defined explicitly in the service type as it is handled by the underlying run time infrastructure.

The Behavioral Aspect describes the dynamics of a service type in terms of states and the control flow between them, including conditions, timing, and exception handling.

In the WSDL-based approach, behavior of a service type is described as local to the primary role, and only the states and control flow of the primary role are explicitly defined. Behavior is specified primarily in a block-structured way using atomic and composite states. Atomic states can represent interactions and service-internal data flow. Composite states support control flow constructs such as sequence, parallelism, interruptible and triggered behavior, and cascading exception handling.

In the ebXML-based approach, the behavior of bilateral service types is defined in terms of the states and the control flow of the relationship as a whole, i.e., without taking into account that each of the role playing actors must manage its own state and control flow. The concepts provided for behavior specification are similar to those provided by UML 1.x activity diagrams, whereby the activities are defined as either interactions or sub-services. The behavior of multilateral

service types is specified differently in that no global synchronized state is assumed. The behavior interrelating different sub-services is specified local to the role involved in these sub-services. In particular, the control flow between certain states of interrelated sub-services can be specified as sequential or nested (bpss:Transition, bpss:onInitiation).

The Transactional Aspect considers the transactional properties of a service type, such as atomicity, and the specific means to achieve them, such as compensation handlers.

In the WSDL-based approach, the transactional aspect is considered to some extent in that a mechanism supporting the specification of transaction compensation in open nested transactions is provided (bpel:compensationHandler). It is, however, not possible to explicitly specify the transactional properties of a service type.

In the ebXML-based approach, a simple solution based on the atomicity property of interactions is provided in that also all bilateral service types are defined as being atomic units of work, but multilateral service types do not exhibit transactional properties. Regarding compensation, no specific concepts are supported, therefore compensating behavior must be specified using control flow mechanisms. Besides atomicity, it can be specified that an interaction has legal consequences if completed successfully (bpss:isLegallyBinding), which is, in some sense, a transactional property.

5 Summary and Outlook

We have introduced a conceptual framework for the analysis of B2B protocols, based on existing frameworks from the areas of B2B protocol specification and workflow management, respectively. Using this framework, two major approaches for specifying B2B protocols, the WSDL-based approach and the ebXML-based approach, have been analyzed and compared.

The results of the comparison show that the difference between the two approaches at the base layers of the eCo framework, i.e., information items and documents, are quite small, whereas at the higher layers, i.e., interactions and services, the approaches provide different concepts.

Regarding the interactions layer, the interaction types provided by the ebXML-based approach are basically a superset of the ones provided by the WSDL-based approach. The latter ones are simple and generic, suitable for many domains and supported by common middleware technology. The interaction types provided by the ebXML-based approach, on the contrary, support declarative specification of many interaction characteristics specifically relevant to business transactions, such as timing constraints, authorization and non-repudiation, messaging reliability, and interaction atomicity. Additionally, there are operational differences as the ebXML-based approach defines its own SOAP extensions.

In the services layer there is a fundamental difference in that the WSDL-based approach primarily supports specifying the interface of a software component supporting a service, meaning that service types are specified from the point of view of individual roles, whereas the ebXML-based approach is closer related to agreements or contracts between collaborating partners, in that service types basically specify binary relationships. As a consequence, behavior specification in the ebXML-based approach is much simpler than in the WSDL-based approach since there is no need to explicitly specify the synchronization of interacting roles in terms of message send and corresponding receipt. But the WSDL-based approach, on the other hand, provides higher expressiveness and flexibility. Furthermore, the WSDL-based approach supports the specification of data flow, which is an important prerequisite for the specification of executable processes. Finally, the ebXML-based approach offers a complete but simple solution to transaction specification, whereas in the WSDL-based approach transactional properties cannot be specified as such.

Resulting from the differences, there is no direct interoperability between the WSDL-based approach and the ebXML-based approach, neither conceptually nor operationally. Considering interoperability at the interactions layer, it is basically possible to express interaction types from the WSDL-based approach in terms of the ebXML-based approach, not taking into account the implied atomicity of ebXML interactions and the operational differences. Vice versa, ebXML interaction types could be expressed using existing and/or forthcoming behavior, security, and transaction features of the WSDL-based approach (cf. Services Layer). This approach allows the combination of features in very flexible ways, e.g. one could realize a transaction which involves several interactions and spans multiple business partners, which is not possible in the ebXML-based approach. The downside is that the resulting specification is much more complex and that the semantics of ebXML's business transactions is not captured explicitly. Besides conceptual interoperability, operational interoperability could be achieved by adaptation mechanisms to translate between the different technologies, however, having conceptual interoperability as a prerequisite. Regarding interoperability at the services layer, in general it is not possible to express WSDL-based service specifications in terms of ebXML-based ones due to ebXML's lack of expressive power in the informational and behavioral aspects. Translating ebXML-based service specifications to WSDL-based ones is possible to some extent. The binary and multiparty relationships as well as the semantics of transactionality and business transactions, however, cannot be translated.

Finally, answering the question of which of the approaches to use, the ebXML-based approach is favorable for its closer alignment with the B2B domain, because it provides more specific concepts and is therefore much simpler to use while being expressive enough to cover typical B2B applications. The WSDL-based approach, on the other hand, is favorable for its stronger vendor support and tool availability. Furthermore, it is closer aligned with existing software components which need to be integrated in the implementation of a B2B protocol in an organization.

To get best of both approaches, one could use a conceptual modeling language such as UML for the design of B2B protocols independent of the idiosyncrasies of particular specification languages, and automatically generate WSDL-based and/or ebXML-based specifications out of the UML models following the model-driven architecture approach[7]. This would require to define a UML profile for B2B protocol specification and corresponding mappings to WSDL and ebXML. Such a UML profile would likely be based on the conceptual framework used in this paper, and on already existing work such as the UMM [19] and OMG's UML Profile for Enterprise Distributed Object Computing [15]. Elaborating the feasibility of these ideas, as well as completing the conceptual framework with the business and market layers of the eCo framework, is subject of ongoing work.

References

1. BEA, IBM, Microsoft, SAP, and Siebel. Business Process Execution Language for Web Services, Version 1.1. `http://ifr.sap.com/bpel4ws/BPELV1-1May52003Final.pdf`, May 2003.
2. M. Bernauer, G. Kappel, G. Kramler, and W. Retschitzegger. Specification of Interorganizational Workflows – A Comparison of Approaches. In *Proceedings of the 7th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2003)*, 2003.
3. C. Bussler. B2B Protocol Standards and their Role in Semantic B2B Integration Engines. In *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, volume 24, pages 3–11. IEEE, 2001.
4. C. Bussler. Modeling and Executing Semantic B2B Integration. In *Proceedings of the 12th Int'l Workshop on Research Issues in Data Engineering: Enginering e-Commerce/e-Business Systems (RIDE'02)*. IEEE, 2002.
5. Q. Chen, U. Dayal, and M. Hsu. Conceptual Modeling for Collaborative E-business Processes. In *Conceptual Modeling – ER 2001*, volume 2224 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2001.
6. Q. Chen and M. Hsu. CPM Revisited – An Architecture Comparison. In *On the Move to Meaningful Internet Systems 2002: CoopIS, DOA, and ODBASE*, volume 2519 of *Lecture Notes in Computer Science*, pages 72–90. Springer, 2002.
7. eCo Working Group. eCo Architecture for Electronic Commerce Interoperability. `http://eco.commerce.net/rsrc/eCoSpec.pdf`, 1999.
8. B. Haugen and T. Fletcher. Multi-Party Electronic Business Transactions, Version 1.1. `http://www.supplychainlinks.com/MultiPartyBusinessTransactions.PDF`, 2002.
9. IBM, Microsoft, RSA, and VeriSign. Web Services Security Policy Language (WS-SecurityPolicy). `http://www.verisign.com/wss/WS-SecurityPolicy.pdf`, 2002.
10. D. E. Jenz. The 'big boys' unite forces – What does it mean for you? `http://www.webservices.org/index.php/article/articleview/633/1/4/`, 2002.
11. G. Kramler. B2B Protocol Specification by Example Using WSDL and ebXML. Technical Report, `http://www.big.tuwien.ac.at/research/publications/2003/0703.pdf`, 2003.

[7] `http://www.omg.org/mda/`

12. H. Kreger. Web Services Conceptual Architecture (WSCA 1.0). http://www.ibm.com/software/solutions/webservices/pdf/WSCA.pdf, May 2001.
13. OASIS. ebXML Collaboration-Protocol Profile and Agreement Specification, Version 2.0. http://www.oasis-open.org/committees/download.php/204/ebcpp-2.0.pdf, 2002.
14. OASIS. ebXML Message Service Specification, Version 2.0. http://www.ebxml.org/specs/ebMS2.pdf, 2002.
15. OMG. UML Profile for Enterprise Distributed Object Computing Specification. OMG Adopted Specification ptc/2002-02-05, http://www.omg.org/cgi-bin/doc?ptc/2002-02-05, February 2002.
16. C. Peltz. Web services orchestration – A review of emerging technologies, tools, and standards. http://devresource.hp.com/drc/technical_white_papers/WSOrch/WS-Orchestration.pdf, 2003.
17. S. Rausch-Schott. *TriGS$_{flow}$ – Workflow Management Based on Active Object-Oriented Database Systems and Extended Transaction Mechanisms*. PhD thesis, University at Linz, 1997.
18. R. Shapiro. A Comparison of XPDL, BPML, and BPEL4WS. http://www.ebpml.org/A_Comparison_of_XPDL_and_BPML_BPEL.doc, 2002.
19. UN/CEFACT. UN/CEFACT Modeling Methodology (N090 of TMWG). http://www.unece.org/cefact/docum/download/01bp_n090.zip, 2001.
20. UN/CEFACT. ebXML Core Components Technical Specification, Version 1.90. http://xml.coverpages.org/CCTSv190-2002.pdf, 2002.
21. UN/CEFACT and OASIS. ebXML Business Process Specification Schema, Version 1.01. http://www.ebxml.org/specs/ebBPSS.pdf, May 2001.
22. W. M. P. van der Aalst. Don't go with the flow: Web services composition standards exposed. In *IEEE Intelligent Systems*. IEEE, 2003.
23. W. M. P. van der Aalst and M. Weske. The P2P Approach to Interorganizational Workflows. In *Advanced Information Systems Engineering (CAiSE 2001)*, volume 2068 of *Lecture Notes in Computer Science*. Springer, 2001.
24. W3C. Web Services Description Language (WSDL) 1.1, W3C Note. http://www.w3.org/TR/2001/NOTE-wsdl-20010315, 2001.
25. W3C. XML Schema Part 1: Structures, W3C Recommendation. http://www.w3.org/TR/xmlschema-1/, May 2001.
26. W3C. XML Schema Part 2: Datatypes, W3C Recommendation. http://www.w3.org/TR/xmlschema-2/, May 2001.