Composition of Coordinated Web Services

Stefan Tai, Rania Khalaf, and Thomas Mikalsen

IBM T.J. Watson Research Center, Hawthorne, New York, USA
{stai,rkhalaf,tommi}@us.ibm.com

Abstract. The Web services architecture defines separate specifications for the composition and the coordination of Web services. BPEL is a language for creating service compositions in the form of business processes, whereas the WS-Coordination framework defines coordination protocols for distributed activities. In this paper, we investigate the combination of these two aspects to compose coordinated Web services. We argue for a policy-based approach to address this problem and introduce a new model and middleware that enables the flexible integration of diverse coordination types into (existing) process-based Web services compositions.

1 Introduction

The landscape of business technology today is shifting. Traditional integrated enterprises with centralized control are giving way to loosely-coupled networks of applications owned and managed by diverse business partners. *Service-oriented computing* is a distributed computing paradigm that treats the distributed, loosely-coupled, heterogeneous nature of this trend in a first-class manner. Its approach is centered on standards and the pervasiveness of Internet technologies.

The *Web services architecture* defines a set of specifications that provide an open XML-based platform for the description, discovery, and interoperability of distributed, heterogeneous applications as services. Included are specifications for business process management and various quality-of-service protocols supporting, for example, transactions, reliable messaging, and security [29] [15] [1].

Figure 1 illustrates the Web services architecture. The various Web services specifications are designed to complement each other, serving as building blocks that can be combined to provide interoperability at different software layers, from low-level transport protocols to high-level application interactions. The combined usage of some specifications is well-understood, such as WSDL [31] for description, SOAP [30] bindings in the WSDL for interaction, and UDDI [25] registries holding WSDL descriptions. However, this is not the case for all specifications, in particular, where the integration of respective middleware implementations supporting the individual Web services specifications is required.

In this paper, we look at the combined use of the Web services specifications for service composition and service coordination: the *Business Process Execution Language (BPEL)* for Web Services [27], and the specifications that use the *Web Services Coordination (WS-C)* framework [19]. These include *Web Services Atomic Transaction (WS-AT)* [20] and *Web Services Business Activity (WS-BA)* [21].

These specifications can be used in combination to support production workflows for Web services. In [6], we provided an overview of these specifications and a high level view of how they conceptually fit together. However, we did not propose a full

H.-A. Jacobsen (Ed.): Middleware 2004, LNCS 3231, pp. 294–310, 2004.
© IFIP International Federation for Information Processing 2004

solution or detailed strategy for how this may be done. In this paper, we argue for the use of declarative policies to address this problem, and introduce an approach that utilizes the *Web Services Policy Framework (WS-Policy)* [12].

Thus, the paper has two main objectives.

1. To provide a process-based Web services composition model that supports the integration of a variety of coordination protocols; current approaches to Web services composition have limited or no support for the external coordination of Web services.
2. To achieve the composition of coordinated activities using existing Web Services specifications. Rather than proposing a new service composition language, we define WS-Policy-based assertions that integrate the existing BPEL language and the WS-C framework (specifically, WS-AT and WS-BA).

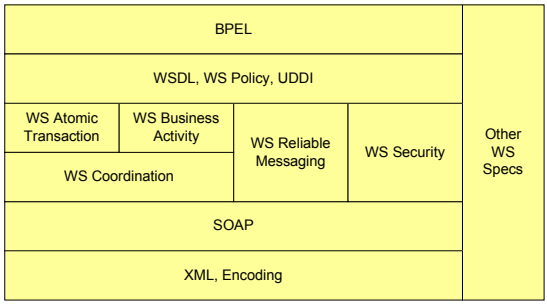

Fig. 1. Web Services Architecture.

2 Background

Web services are software applications that support open, Internet and XML standards-based interfaces and protocols [29]. In this section, we provide a brief summary of the Web services specifications relevant to our discussion. We refer the reader to the published specifications and diverse Web services literature for further details [1] [14] [24].

The functional description of a Web service is provided by the *Web Services Description Language (WSDL)* [31]. WSDL separates the abstract functionality of a service from its mappings to available deployed implementations. The abstract description consists of the operations supported by a service and the definition of their input and output messages. A *portType* groups a set of operations.

The concrete aspects of a WSDL definition include *bindings* that map operations and messages of a portType to specific protocol and data encoding formats (such as SOAP), *ports* that provide the location of physical *endpoints* implementing a specific portType using a specific binding, and *services* definitions as collections of ports.

The *Web services Addressing (WS-Addressing)* specification has been developed to provide transport-neutral mechanisms to identify Web services endpoints through *endpoint references* and to address messages through *message information headers* [3].

The *Business Process Execution Language (BPEL)* is a language for creating compositions of Web services in the form of business processes [27]. Compositions are created by defining control semantics around a set of interactions with the services

being composed. The BPEL composition model is recursive: a BPEL process, like any Web service, supports a set of WSDL interfaces that enable it to be exposed and invoked as a regular Web service.

A BPEL process contains a set of typed connectors known as *partnerLinks*, each specifying the portType required from the party being connected along that link and the portType provided by the process to that party in return. The composition model explicitly stays away from binding these to actual service endpoints, leaving the door open for flexible binding schemes and selection algorithms. The "activity" is the unit of composition. Primitive activities provide such actions as Web services invocations, waiting, and throwing faults. Structured activities impose predefined control semantics on the activities nested within them, such as sequence or parallel execution. Additional control may also be defined using explicit conditional control links. We mention one BPEL activity that is of particular interest to this paper: the *scope*. BPEL scopes contain a set of (nested) activities and provide the unit of data, fault, and compensation handling.

The *Web Services Coordination (WS-C)* specification provides an extensible framework for the definition of protocols that coordinate distributed activities [19]. The framework can be used to support different coordination types, including atomic transactions and long-running business transactions. WS-C enables the creation of coordination contexts for propagation among coordination participants, and the registration of participants for particular coordination protocols of a given coordination type. Further details are provided in Section 4 of this paper.

Examples of specific coordination types are *Web Services Atomic Transaction (WS-AT)* [20] and *Web Services Business Activity (WS-BA)* [21]. These specifications define agreement coordination protocols, such as a durable two-phase commit protocol (WS-AT) or a participant-driven completion protocol for business transactions (WS-BA). Other coordination types and protocols can be defined using WS-C.

The *Web Services Policy Framework (WS-Policy)* defines a general-purpose model and syntax for expressing functional or non-functional properties of a Web service in a declarative manner [12]. A policy is an XML-expression that logically combines one or more assertions which specify concrete or abstract service characteristics such as a required security authentication scheme or a desired quality of service. Policies can flexibly be attached to various Web services definitions, including WSDL type definitions, as described in the *Web Services Policy Attachment* specification [13].

3 Motivation: Composing Coordinated Activities

In service-oriented computing, services are the basic building blocks out of which new applications can be created. With a plethora of services in-place and accessible in a standardized way, composition languages such as BPEL are needed to weave those services together and subsequently expose the resulting artifact itself as a Web service.

Composition and coordination and, correspondingly, composition middleware and coordination middleware, are two complementary aspects and techniques. The schema for a service composition is an aspect that is mostly internal to the implementation of the service that composes other Web services, whereas the protocols for service coordination are required properties of the external interactions between Web services [1].

A composition of Web services may not always require additional external coordination protocols. However, in order to develop *production workflows* [22] any functional composition of a set of Web services must be combined with the non-functional (reliability, transactions, security, and other) coordination properties required for process partner interactions. Production workflows are processes that define and implement the business logic and the quality-of-service necessary to integrate distributed heterogeneous applications. The overall motivating objective for our work is to enable production workflows using Web services.

Consider a workflow process that must interact with a partner using a reliable messaging protocol (such as *Web Services Reliable Messaging, WS-RM* [9]). The workflow (BPEL process) and the service provider each need to advertise their support for a reliable messaging protocol as a capability and/or requirement for interaction. In a dynamic service-oriented computing environment, such advertisement must be part of the WSDL service descriptions. The process and partner service use such information to agree on a particular protocol, which must then be supported by the middleware implementations of both workflow client and service provider.

Another example concerns declaring a (sub-) set of activities within a business process to be atomic using the WS-AT atomic coordination type. The service partners that are part of this atomic scope must correspondingly support the required WS-AT coordination protocols.

Additionally, another set of activities within the same business process may need to be coordinated with partners using a loosely-coupled business transaction model. A transaction coordination type such as WS-BA is required here.

The requirement for different coordination types and their protocols in a single service composition is illustrated in Figure 2. Here, activity *1* interacts with Web service *A* using the WS-RM protocol. Activity *2* and activity *3* are coordinated with Web service *B* and *C* using the WS-AT coordination type. Activity *4* is coordinated with Web service *D* using the WS-BA coordination type.

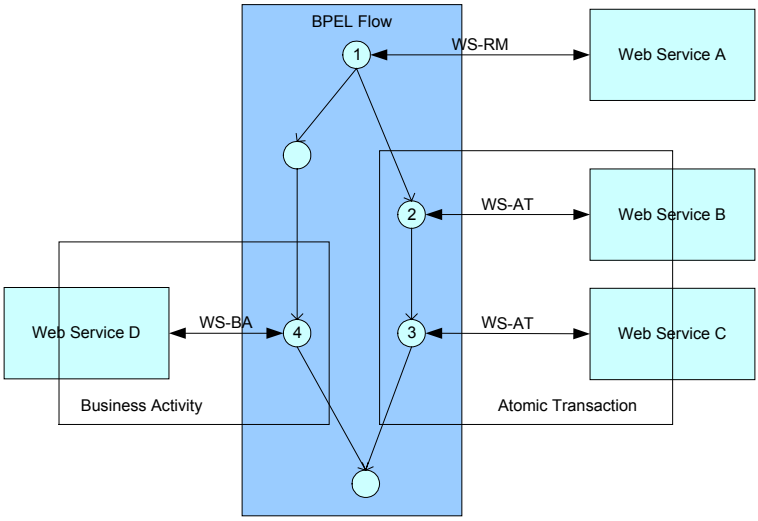

Fig. 2. Coordination requirements for service composition.

In this paper, we investigate a policy-based approach and middleware for composing coordinated activities as illustrated in Figure 2. The objective is to integrate the use of external coordination protocols (specifically, WS-AT and WS-BA) for different activities of a BPEL process composition, where BPEL defines the scope of the coordination. We propose a generic approach that can be applied for various defined coordination types.

It is not our objective to investigate the integration of all Web services interoperability protocols (like WS-RM) and of ad-hoc combinations of diverse protocols (such as a combination of WS-AT and WS-RM). We focus attention on the integration of (transactional) coordination types that are based on WS-C. Our approach aims to support any coordination type that is defined using WS-C.

4 Coordination Policies for BPEL

We propose a model for integrating WS-C coordination types and BPEL definitions using declarative policies attached to selected Web services constructs. The model does not introduce a new language or Web service specification, but integrates existing specifications through policy assertions.

4.1 Coordination Model

The WS-C specification defines three main elements that are commonly required for different kinds of coordination:

- A *coordination context*: the context that is shared and propagated among the participants in the coordinated activity
- An *activation service*: the Web service used to create a coordination context
- A *registration service*: the service used by participants to register for inclusion in specific coordination protocols

WS-C *coordination types* such as WS-AT and WS-BA extend the proposed coordination context, adapt the registration service (and optionally, the activation service) and define a set of specific *coordination protocols* and corresponding protocol Web services. The protocol services, registration service, and activation service together constitute a *coordinator* (coordination middleware).

Figure 3 illustrates the principle WS-C architecture. A coordination participant, in role of a requestor or a responder, is an application that uses a coordinator. The application interacts (locally) with the coordinator to create a coordination context (omitted in the figure). The context is propagated to any other (remote) participant(s) via an application message. The context includes the WS-Addressing endpoint reference of the registration service of the requestor's coordinator, so that the responder's coordinator ("sub-coordinator") can register for participation in a specific coordination protocol. The coordination protocol messages are then exchanged between the coordinators.

A coordination participant thus is any application that is capable of understanding WS-C contexts. In addition, a coordination participant requires a coordination middleware for WS-C protocol registration and specific (WS-AT, WS-BA, or other) protocol interactions.

Technically, a coordination participant is a set of Web services that support application-specific and coordination-middleware interfaces (port types) that may result in

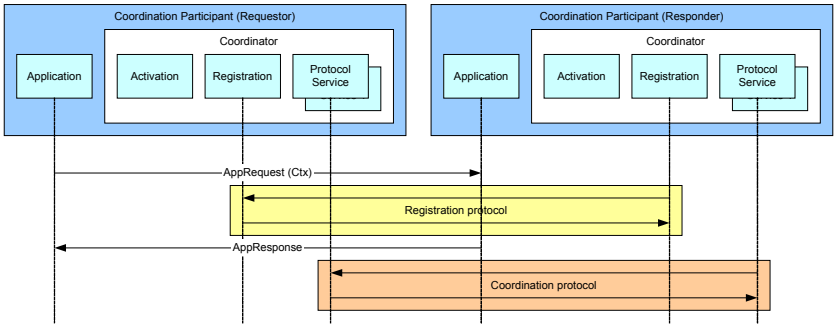

Fig. 3. WS-Coordination Architecture.

multiple endpoints at runtime. In this paper, we use both "coordinated (Web) service" and coordination participant to refer to the same concept.

4.2 Coordination Policies

The capability (of a Web service or BPEL process) to participate in WS-C coordination can easily be communicated using a declarative policy assertion. We define a *coordination policy* as an XML element referencing a WS-C coordination type and specific protocols of that type.

An example policy for the WS-AT coordination type and its durable two-phase commit protocol is given below. The policy uses the XML syntax defined in the WS-Policy framework [12] and the XML element <wsce:CoordinatedService> for coordination policies as proposed in [26].

```
<wsp:Policy wsu:Name="tns:WSATPolicy"
  <wsce:CoordinatedService
    CoordinationType=
      "http://schemas.xmlsoap.org/ws/2003/09/wsat">
    <wsce:Protocol
      ProtocolIdentifier=
        http://schemas.xmlsoap.org/ws/2003/09/wsat#Durable2PC
    />
  </wsce:CoordinatedService>
</wsp:Policy>
```

The policy references a (transaction) coordination type that is defined in a published XML schema. Authoring such policies is a matter of selecting a published WS-C coordination type and including in the <wsce:CoordinatedService> element the links that hold the XML schema.

These policies can then be flexibly and meaningfully attached to various Web services definitions [13]. To declare a coordination capability of a deployed Web service provider, we attach coordination policies to Web services port bindings. For example, a banking Web service "ABCBankService" may declare its support for WS-AT as follows.

```
<service name="ABCBankService"
  <port name="creditAccount" binding="tns:CreditBinding"
     wsp:PolicyRefs="tns:WSATPolicy">
     <soap:address location=…/>
  </port>
</service>
```

The policy attachment defines that the service supports the WS-AT coordination type and its durable two-phase commit protocol. If a client invocation, such as for debiting or crediting a customer account carries a coordination context, the invoked operation will be executed according to the WS-C coordination model.

The policy "WSATPolicy" defined above may also be attributed with a WS-Policy usage attribute such as <wsp:Required> or <wsp:Optional> [12]. The WS-Policy usage attributes, if specified for a coordination type, define the processing semantics of the policy. In this example, the WS-AT coordination type may be declared to be required or optional for any invocation on the port that the policy is attached to. If a required attribute is specified and an invocation on the port does not carry a proper coordination context, a fault will be raised.

Different coordination (and other) policies may also be combined using WS-Policy operators corresponding to the logical operators AND, OR, XOR. For example, a service that alternatively supports two coordination types would create a policy for each and combine them with an XOR. The WS-Policy operators corresponding to AND, OR, and XOR are <wsp:All/>, <wsp:OneOrMore/>, and <wsp:ExactlyOne/>, respectively.

4.3 BPEL Coordinated Partner Links and Coordinated Scopes

Through the attachment of coordination policies coordination semantics are introduced to (existing) BPEL compositions. We propose to attach coordination policies to BPEL *partner links* and to BPEL *scopes*. As noted earlier, a BPEL partner link is a typed connector along which a conversation with another party occurs. A BPEL scope is the demarcation of a group of activities of the process. Scopes are the units of fault handling and compensation in BPEL.

A partner link with an attached coordination policy is a *coordinated partner link*. Such a link describes the (abstract) requirement on any (concrete) deployed Web service that aims to provide the partner functionality at process execution time. The interpretation is similar to the attachment of a policy to a WSDL port type definition; it is a requirement for every deployed service to satisfy the policy [13].

If a coordinated partner link is used within a regular BPEL scope, for the duration of the scope the conversation with that partner is carried out using the declared coordination protocol. That is, a coordination context will be created for the conversation with that specific deployed service. Interactions with other (non-coordinated) partners in the same scope will not share the coordination context. This is illustrated in Figure 4.

In addition to partner links, however, coordination policies can also be attached to BPEL scopes to define *coordinated scopes*. The semantics of a policy attachment to a scope is that a coordination context is created for the scope by the BPEL middleware and that the context will be propagated to all the partners that are part of the scope.

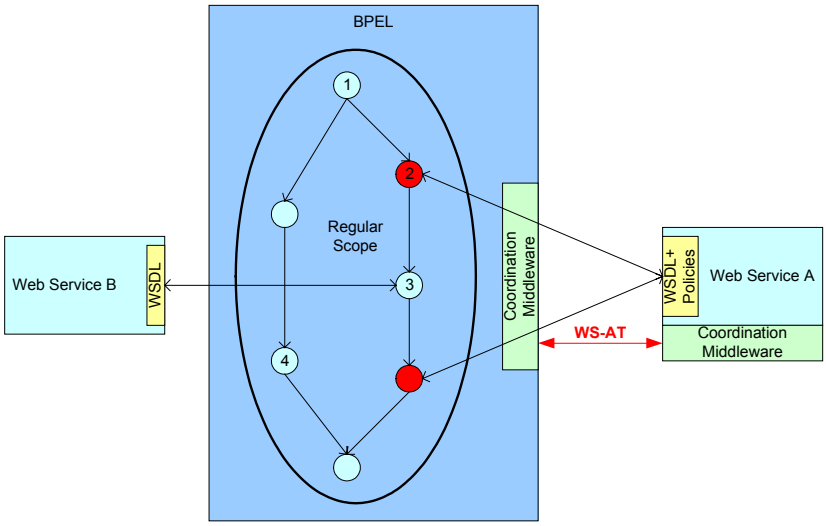

Fig. 4. Coordinated partner links in regular scopes: Attaching a WS-AT policy on a PartnerLink connecting the process with Web Service A and required coordination middleware for WS-AT protocol interactions.

Using a WS-AT coordination policy, for example, an *atomic scope* [22] can be modeled. Using a WS-BA coordination policy, a compensation-based *business activity scope* can be modeled.

WS-BA coordinated scopes are not to be confused with the concept of *compensation scopes* [22], which are the units of compensation handling that are already present in BPEL. WS-BA coordinated scopes establish a coordination context for distributed partners that engage in compensation-based business transaction protocols; WS-BA defines the messages and message exchange order for driving compensation. BPEL compensation scopes define actual compensation flows to be executed.

If coordination policies are attached to both scopes and partner links, the policies of the scope dictate the required policy for each partner. For example, if a scope is declared to be atomic using a WS-AT coordination policy, each coordinated partner in the scope must be compatible and support the WS-AT coordination type. The scope's policy defines a requirement on all partner links of the scope, and establishes a shared context for all partners in the scope. This is illustrated in Figure 5.

In either case of using a coordination context for a specific partner conversation (in a regular scope), or for conversation with multiple partners (in a coordinated scope), the BPEL scope demarcates the coordination. The context is created when entering the scope and the coordination is completed when closing the scope.

Notice that with coordinated scopes, BPEL is in the role of the initiator (requestor) of the coordination. Coordinated scopes do not model BPEL as a participant (responder) that registers with coordination contexts that have been created outside the process and propagated to the BPEL process through receive operations. Also, coordinated scopes do not allow nesting. External coordination and nesting of coordinated scopes is discussed in Section 6.3.

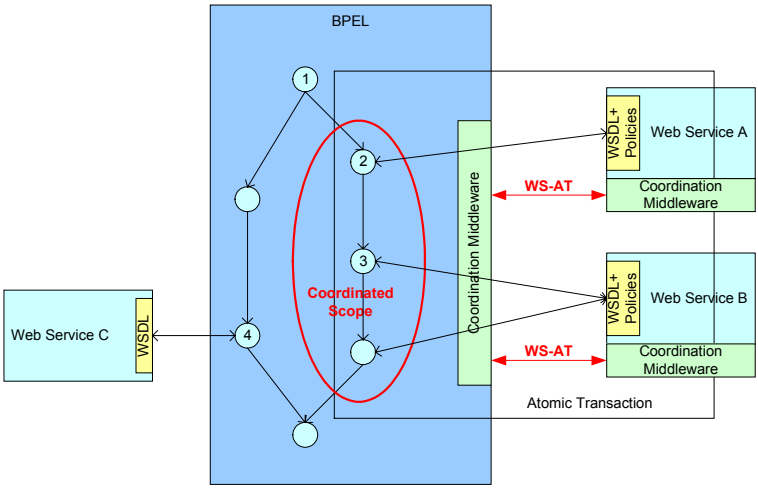

Fig. 5. Coordinated partner links in coordinated scopes: Attaching a WS-AT policy to a scope and required coordination middleware for WS-AT protocol interactions with all partners of the scope.

Using coordination policies, a scope defines a WS-C coordinator that provides the required registration and coordination type-specific protocol service port types. Coordination middleware on both sides can then engage in protocol interactions.

The programming complexity of authoring and attaching coordination policies to BPEL compares to the complexity associated with (declarative) transaction processing in general. A transaction model (for atomic transactions, long-running business transactions, or other) must be carefully selected for a given coordination problem, and the desired transaction semantics of the process must be carefully analyzed. Our policy-based approach does not simplify the task of understanding transactional semantics, but allows for a simple and effective way to extend BPEL to support (different) transaction coordination models.

4.4 Policy Matchmaking

The above proposed model of coordinated partner links and coordinated scopes introduces the need for two kinds of policy matchmaking:

- A static check on the compliance of coordinated scope and coordinated partner link policies within the BPEL definition
- Deployment-time and/or runtime policy matchmaking of coordinated partner links and deployed services

4.4.1 Static Verification

Static verification takes place to ensure that all coordinated partner links support at least the policies needed by the coordinated scope. For each coordinated scope, all BPEL constructs that make use of an abstract partner (such as an invoke statement) are verified: The coordination policy of the partner must satisfy the policies of the scope. Static verification ensures a correct BPEL process flow before instantiating the

abstract partners; it ensures the ability of a single coordination context of a particular coordination type to be shared among all partners of the scope.

4.4.2 Dynamic Matchmaking

After static verification is successfully completed, all coordinated partner links, within regular scopes or within coordinated scopes, describe valid requirements for the interaction with deployed services that aim to fulfill the partner role at process execution time.

The partner links can be instantiated at BPEL deployment time or dynamically through the exchange of endpoint references at runtime. When instantiation occurs, the coordination policies of the partner links must be matched with the coordination policies declared by the deployed services. The policies must be evaluated for compliance of their coordination type and protocols, and they must not conflict with one another.

Policy matchmaking determines if the BPEL requirements on coordination can be fulfilled by the deployed service under investigation. Matchmaking is required only once for each partner link, as long as the effective policy and the physical endpoint reference of the partner do not change.

The policy matchmaking algorithm [32] first calculates for each (potentially complex) WS-Policy expression the acceptable assertion set in terms of Boolean algebra. Each assertion, such as support for a coordination protocol, is interpreted as a unique Boolean variable. The acceptable assertion set is the set that consist of a list of assertions that when set to true reduce the entire policy to true. For each set, all assertions that are not on the list are set to false.

Next, the acceptable policy sets are compared to find matching sets that contain exactly the same list of assertions. A matching set is then selected by the middleware. If no matching set is found then the requestor and responder are incompatible and a BPEL runtime exception will be raised. (In future work, a mismatch may be resolved through dynamic policy negotiation or other application logic.)

4.5 Policy Mediation Meta-protocol

To compare policies for matchmaking, a policy mediation meta-protocol such as the GPP described in [32] can be used. The GPP proposes the steps of *policy request* to initiate a policy exchange, *policy promise* as the reply by the responder, and *policy confirmation* as the notification of a successful match. The protocol must be executed for any partner conversation for which no matching policy is in effect.

Alternatively, the recently published *Web Services Metadata Exchange (WS-MetadataExchange)* specification may be used [8]. WS-MetadataExchange defines three request-response message pairs to retrieve the policies, WSDL, or the XML schema of a Web services endpoint and/or given target namespace. WS-MetadataExchange replaces a proprietary solution like the GPP.

4.6 Programming Model

The programming model for composing coordinated activities is standard BPEL. Coordination policies, which may be defined separately, are attached to selected partner links and/or scopes. Coordinated partner links are interpreted depending on the

declaration of coordinated scopes, as described above. The required coordination policy for a partner interaction is determined using static verification. Policy matchmaking, possibly using a policy mediation meta-protocol, is performed when needed at deployment and/or runtime. Policy mediation and matchmaking, as well as all coordination protocol interactions are the responsibility of the supporting middleware.

5 Middleware Prototype

We have implemented a middleware research prototype that demonstrates and validates the approach described. This prototype provides a *BPEL compiler*, a *policy middleware*, and a *Web services transaction processing middleware (WSTPM)* that supports the WS-AT and WS-BA coordination types.

5.1 Components

Using our prototype, a BPEL definition with coordinated scopes and coordinated partner links is parsed and processed, together with the WSDL definitions of deployed partners (those available at deployment time), to generate a Java implementation. During code generation, the policy middleware is consulted for policy matchmaking.

The resulting code is then deployed as a regular Web service. For those abstract partners that did not have a concrete service at deployment time, the GPP policy mediation meta-protocol and dynamic policy matchmaking is executed at runtime. The generated BPEL Web service interacts with the policy middleware for this purpose. The generated Web service further interacts with the WSTPM to begin, end, and manage transactional coordination. Standard J2EE and proprietary APIs are used for the generated service to interact with the policy and WSTPM middleware. Figure 6 illustrates a sample deployment architecture that utilizes our policy and WSTPM middleware on all nodes.

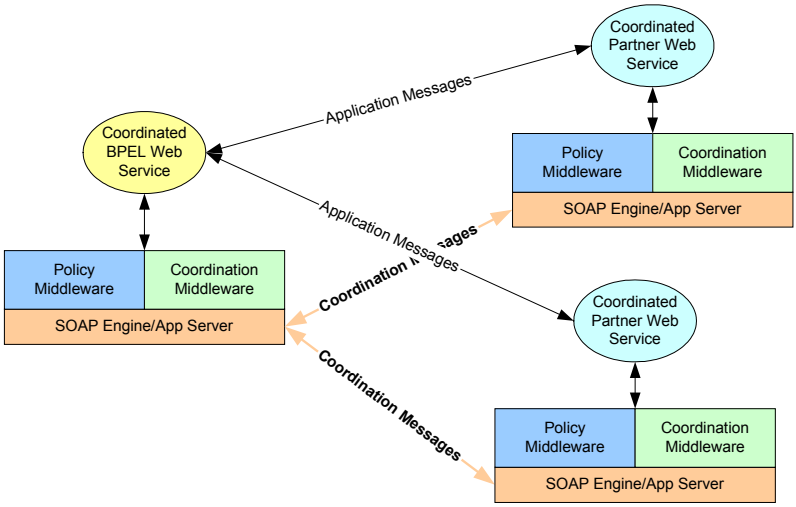

Fig. 6. Middleware prototype: Sample deployment architecture.

5.2 Process Execution

The coordination policies describe external interaction properties required for process execution. When entering a scope, a WS-C coordination context is created using the coordination middleware. In our prototype, the context is created by the generated service implementation executing the process. Alternatively, the context may be created by a BPEL middleware (a process runtime engine) or by any other coordination middleware client.

The context is then propagated to the partner as part of an application message, and the partner's coordination middleware in turn registers with the requestor's coordination middleware as previously described in Section 4. Subsequent application messages will carry the coordination context. The context is the means based on which the coordination middleware on both sides can identify the coordination to handle messages appropriately.

When the BPEL scope is about to close, the process implementation (the generated Web service or a process runtime) initiates any completion necessary according to the chosen coordination type. In case of WS-AT, a commit request is made to the local coordination middleware, which in turn will drive the two-phase commit protocol with all registered remote coordinators. In case of WS-BA, a completion request is made and the local coordinator will send WS-BA completion messages to all registered participants.

Failures can either be communicated using the defined coordination protocol messages (such as an "abort" message), in which case they will be handled by the coordination middleware. Failures may also be detected and handled by the BPEL middleware, and then communicated by the BPEL middleware to the coordination middleware.

Notice that heterogeneous middleware systems can be used on either end, as long as they support the Web services specifications. That is, (transactional) coordination can be implemented in different ways, with diverse internal options for registering (local and remote WS-C) resources and handling recovery from failures, as long as the external messages conform to the WS-C coordination types. For example, a coordination middleware may use a proprietary transaction service such as the J2EE Java transaction service. Java transaction contexts must then be mapped to WS-C XML contexts for all external interaction, and the messages and protocols defined in the WS-AT and WS-BA specifications must be understood and supported for interoperability.

6 Discussion and Related Work

In this paper, we presented an approach to composing coordinated Web services by means of declarative policy attachments to Web services definitions.

We previously reported on our work from the viewpoint of transaction processing [26] and software engineering [32], independent of the BPEL-related issues presented in this paper.

Although the use of policies for combining BPEL with WS-C coordination types has been mentioned as a desirable approach before [27] [16], we are not aware of prior work that has concretely combined the composition and coordination of Web services using policies. In this section, we discuss our approach and study related work.

6.1 Other Coordination Protocols

This paper provides a solution in which the supported coordination models for BPEL are WS-C coordination types. Other Web services transaction and coordination specifications have been proposed, however, including the Business Transaction Protocol (BTP) [5] and the Web Services Composite Application Framework (WS-CAF) [4]. Both define coordination types that are similar to WS-AT and WS-BA, but also feature subtle semantic differences.

In order to support these alternative coordination models with our approach, the coordination models would need to be represented as WS-C coordination types. Then, corresponding policies can be defined and attached as described in this paper. We do not see any reason why a definition of these alternative models as WS-C coordination types would not be possible. WS-C is a generic framework providing only the very fundamental coordination mechanisms.

Other, non-transactional coordination protocols have also been proposed. These include a service availability tracker [28], previously published as WS-Membership. Since it utilizes the same WS-C coordination model, we believe this coordination type to also be readily applicable to our approach. In this way, a BPEL process may also support group membership coordination for selected partners.

In all cases of BTP, WS-CAF, and the service tracker, however, corresponding middleware systems must also be integrated.

Non-WS-C coordination types, such as the reliable messaging protocol WS-RM, may also be applicable. This depends however on the extent that the coordination protocol interactions are (or can be) separated from the application messages. For example, message acknowledgments of receipt may be communicated via the reliable messaging middleware. The WS-RM message ordering and sequencing constructs for application messaging, however, may conflict with the BPEL message sequencing and flow definition. The composition of WS-RM and BPEL may therefore require additional integration and preference rules beyond the attachment of policies.

6.2 Other Coordination-Aware Composition Models

The most directly related approach to solving part of the problem addressed in this paper is presented in [10]. In that work, transaction management capabilities are added directly to BPEL as language extensions. New syntax is proposed to add support for a subset overlapping with specific business coordination models (WS-BA, BTP Cohesions).

In contrast, our work uses policies to non-intrusively attach coordination capabilities to parts of BPEL process definitions. Our approach enables an extensible variety of coordination protocols to be used, the potential of dynamically choosing from a set of supported protocols based on the environment, and the separation of concerns between the business process logic and the available/required coordination protocols. While extending the language directly may provide for a more integrated BPEL (middleware) implementation, we believe our approach using policy attachments to be more aligned with the dynamic nature of the service-oriented computing landscape. It is this dynamic nature that has led to the modularity of the Web services stack of specifications.

In our approach, coordination models are represented as policies that complement the workflow definition. In this way, a business process may interact with different services using the diverse coordination protocols that they require without the need to redefine the business logic or to port it to another language or system. Implementations executing the business process can then make use of modules that support the different parts of the Web service stack to make the execution happen. In this way, our approach introduces a dynamic aspect-oriented programming model for BPEL.

A large number of existing workflow systems also has built-in, proprietary support for transactional coordination. This has been a major requirement of workflow since many years. For example, [22] defines compensating actions on activities and a default coordination model for the set of compensable activities. The transaction literature has also proposed many ways to do (extended) transactions, some of which essentially are coordination-based workflows. The Sagas model, for example, defines a long-running process to consist of a set of atomic transactions and corresponding compensating transactions [11]. Some of these extended transactions models have also been adapted to support Web services, for example [18]. These approaches are similar to the approach of extending the BPEL language as discussed above in that they do not provide for an open, dynamic integration of diverse coordination models.

Even BPEL itself defines a built-in compensation mechanism that operates at the level of scopes. However, as described earlier in Section 4.3, BPEL compensation scopes differ from (WS-AT or WS-BA) coordinated scopes. BPEL and related workflow systems have considered the transactions governing the process model's activities directly and do not consider the interactions with services in a first class manner or how their coordination requirements may be composed. This is where the combination of BPEL and such specifications as WS-C comes into play. Additionally, these approaches are not adaptable to different coordination requirements of service providers.

The basic idea of attaching declarative policies to business process definitions is not new either. However, prior work on policy-based composition has been mainly used for selecting which service provider(s) to bind to at runtime, using measurable parameters. [2] devise a global planning approach to optimizing service selection during execution based on a set of measurable quality of services properties that can be objectively compared (such as price, reputation, and reliability). This is in contrast to our work on coordination interoperability protocols. Also, [23] suggests semantic annotations on BPEL processes that can be used at runtime to perform matchmaking and possible service chaining. However, our work is unique to our knowledge in that policies are used to describe and choose from a set of coordination protocols which require a middleware for runtime interoperability.

6.3 Future Work

In this paper, we address the problem of composing coordinated Web services independent of how the services are implemented. It may be that the coordinated services are themselves compositions, in which case the question of "composing coordinated compositions" arises. For example, a BPEL process coordinates services which are implemented as BPEL processes.

In this case, there are two "levels" of composition and coordination. The first level is the BPEL process that initiates coordinated activities (as presented in this paper); the second level concerns the BPEL processes that join those coordinated activities.

The first level does not need to differentiate between simple and composed services, since the composed services appear as regular Web services. This paper so far has only addressed first level composition of coordinated services.

In order to address the second level, our approach would need to be extended to allow the coordinated scopes (of the second level) to be coordinated by the first level composition. This includes the ability to accept and register with incoming coordination contexts (on receives) in addition to creating new coordination contexts as described earlier in this paper. Also, the compatibility of a coordinated scope with the coordination policy declared on the receiving partner link within that scope must be verified. The propagation of an incoming coordination context to other partners within coordinated scopes must be guided by policies.

There are also some special cases in the composition of diverse coordination protocols in BPEL that need to be addressed. The semantics of nested coordinated scopes requires careful attention. If a coordination type does not support nesting, BPEL scopes should not be used to introduce nesting into the coordination type. For example, a WS-BA coordinated scope within a WS-AT coordinated scope is undefined. One solution is to raise an error during the static process verification, and to only allow nested coordinated scopes for those coordination types that define nesting themselves.

Other future work relates to the generation of a Java implementation (to be deployed as a Web service) for executing the BPEL composition. Our current prototype does not support code generation for all possible BPEL constructs; for example, complex partner conversations using asynchronous messaging are not supported. While the prototype could be extended to fully support all BPEL constructs, the generated code may become increasingly complex and require sophisticated support mechanisms. These include support for parallelism, the conversational nature of BPEL [17], correlation and the handling of faults and conditional links/joins. The use of a first-class, separate BPEL runtime such as [7] may therefore be advantageous. The BPEL runtime however would need to be integrated with a policy and coordination middleware, as described in Section 5.

7 Conclusion

The Web services architecture intends to provide a standards-based platform for service-oriented computing. Various specifications supporting the integration of distributed heterogeneous applications as Web services are proposed. These include the Business Process Execution Language (BPEL) for service composition and the Web services Coordination (WS-C), Atomic Transaction (WS-AT), and Business Activity (WS-BA) specifications for (transactional) service coordination.

Additionally, the descriptive capabilities of WSDL are enhanced by the Web services Policy Framework (WS-Policy), which extends WSDL to allow the encoding and attachment of quality-of-service information in the form of reusable declarative policies.

In this paper, we investigated the combination of BPEL with WS-C, WS-AT and WS-BA using WS-Policy to support the definition of production workflows for Web services. We introduced coordination policies and specific BPEL coordination policy attachments to compose Web services that require coordination protocols for interaction. We defined the semantics of the proposed policy-based composition model and

discussed methods, programming model, and middleware support needed for defining and executing composed coordinated services.

Revisiting the two objectives of the paper stated in the beginning, we have (1) introduced a process-based Web services composition model that supports a flexible, dynamic integration of diverse coordination protocols, and (2) demonstrated the feasibility of combining the existing Web services specifications of BPEL and WS-C using WS-Policy.

We discussed the advantages of our approach, which include the ability to support an extensible variety of coordination types in BPEL, to dynamically choose among the types, and to allow for a clear separation of concerns between business process logic and (different) coordination protocols. The flexibility comes at the expense of a potentially more complex middleware infrastructure, which must integrate the various implementations of the individual (BPEL, policy, and coordination) specifications supported.

Acknowledgements

The research presented is based on joint work with Isabelle Rouvellou, Eric Wohlstadter, and Nirmit Desai. We gratefully acknowledge their invaluable contributions.

References

1. G. Alonso, F. Casati, H. Kuno, V. Machiraju. *Web Services*. Springer-Verlag, 2004
2. B. Benatallah, M. Dumas, Z. Maamar. Definition and Execution of Composite Web Services: The Self-Serv Project. *Data Engineering Bulletin, 25(4)*, 2002
3. D. Box, F. Curbera (Eds). *Web Services Addressing (WS-Addressing)*. Published online at http://www.ibm.com/developerworks/library/ws-add, 2004
4. D. Bunting, M. Chapman, O. Hurley, M. Little, J. Mischkinsky, E. Newcomer, J. Webber, K. Swenson. *Web Services Composite Application Framework (WS-CAF)*. Published online at http://developers.sun.com/techtopics/webservices/wscaf/, 2003
5. A. Ceponkus, S. Dalal, T. Fletcher, P. Furniss, A. Green, B. Pope. *Business Transaction Protocol*. Published online at http://www.oasis-open.org/committees/download.php/1184/2002-06-03.BTP_cttee_spec_1.0.pdf, 2003
6. F. Curbera, R. Khalaf, N. Mukhi, S. Tai, S. Weerawarana. The Next Step in Web Services. *Communications of the ACM,* 46(10):29-34, 2003
7. F. Curbera, M. Duftler, R. Khalaf, N. Mukhi, W. Nagy, S. Weerawarana. *BPWS4J*. Published online at http://www.alphaworks.ibm.com/tech/bpws4j, 2003
8. F. Curbera, J. Schlimmer (eds.) *Web Services Metadata Exchange (WS-MetadaExchange)*. Published online at http://www-106.ibm.com/developerworks/ library/specification/ws-mex/, 2004
9. C. Ferris, D. Langworthy (eds.) *Web Services Reliable Messaging (WS-ReliableMessaging)*. Published online at http://www-106.ibm.com/developerworks/ webservices/library/ws-rm/, 2004
10. T. Fletcher, P. Furniss, A. Green, R. Haugen. *BPEL and Business Transaction Management*. Published online at http://www.choreology.com/standards/BPEL.and. Busi ness.Transaction.Management.Choreology.Submission.html, 2003
11. H. Garcia-Molina, K. Salem. Sagas. *Proceedings ACM SIGMOD*, 1987
12. M. Hondo, C. Kaler. *Web Services Policy Framework (WS-Policy)*. Published online at http://www-106.ibm.com/developerworks/library/ws-polfram/, 2003

13. M. Hondo, C. Kaler. *Web Services Policy Attachment (WS-PolicyAttachment).* Published online at http://www-106.ibm.com/developerworks/library/ws-polatt/, 2003
14. IBM Corporation. http://www-136.ibm.com/developerworks/webservices/
15. R. Khalaf, F. Curbera, W. Nagy, S. Tai, N. Mukhi, M. Duftler. Understanding Web Services. In M.P. Singh (ed.), *Practical Handbook of Internet Computing*, CRC Press, 2004 (to appear)
16. R. Khalaf, F. Leymann. On Web Services Aggregation. *Proceedings of the VLDB Technologies for e-Services Workshop*, Springer LNCS, 2003
17. R. Khalaf, N. Mukhi, and S. Weerawarana. Service-Oriented Composition in BPEL4WS. *Proceedings of the 2003 World Wide Web Conference, Web Services Track*, 2003
18. N. B. Lakhal, T. Kobayashi, H. Yokota. THROWS: An Architecture for Highly Available Distributed Execution of Web Services Compositions. *Proceedings of the 14th International Workshop on Research Issues on Data Engineering Web Services for E-Commerce and E-Government Applications*, 2004
19. D. Langworthy (ed.) *Web Services Coordination (WS-Coordination).* Published online at http://www-106.ibm.com/developerworks/library/ws-coor/, 2003
20. D. Langworthy (ed.) *Web Services Atomic Transaction (WS-AtomicTransaction).* Published online at http://www-106.ibm.com/developerworks/library/ws-atomtran/, 2003
21. D. Langworthy (ed.) *Web Services Business Activity Framework (WS-BusinessActivity).* Published online at http://www-106.ibm.com/developerworks/webservices/library/ws-busact/, 2004
22. F. Leymann, D. Roller. *Production Workflow*. Prentice-Hall, 2000
23. D. J. Mandell, S. A. McIlraith. A bottom-up Approach to Automating Web Service Discovery, Customization, and Semantic Translation. *Proceedings of the 2003 WWW Conference Workshop on E-Services and the Semantic Web*, 2003
24. Microsoft Corporation. http://msdn.microsoft.com/webservices/
25. OASIS. *UDDI*. Specifications published at http://www.oasis-open.org/committees/ uddi-spec/doc/tcspecs.htm
26. S. Tai, T. Mikalsen, E. Wohlstadter, N. Desai, I. Rouvellou. Transaction Policies for Service-Oriented Computing. In *Data and Knowledge Engineering Journal*, Special Issue on Contract-based Coordination and Collaboration, 2004 (in press)
27. S. Thatte (ed.) *Business Process Execution Language for Web Services Version 1.1*. Published online at http://www-106.ibm.com/developerworks/library/ws-bpel/, 2003
28. W. Vogels. Tracking Service Availability in Long Running Business Activities. In *Proceedings 1st International Conference on Service-oriented Computing*, 2003
29. W3C. *Web Services Architecture Requirements*. Published online at http://www.w3.org/TR/wsa-reqs, 2002
30. W3C. *SOAP*. Specifications published at http://www.w3.org/2000/xp/Group/
31. W3C. *Web Services Description Language (WSDL)*. Specifications published at http://www.w3.org/2002/ws/desc/
32. E. Wohlstadter, S. Tai, T. Mikalsen, I. Rouvellou, P. Devanbu. GlueQoS: Middleware to Sweeten Quality-of-Service Policy Interactions. In *Proceedings of the 26th International Conference on Software Engineering*, 2004