AN INFRASTRUCTURE FOR SHARING AND EXECUTING CHOREOGRAPHIES

Alan Massaru Nakai[1], Carla Geovana N. Macário[1,2], Edmundo Madeira[1]
and Claudia Bauzer Medeiros[1]
[1]*Institute of Computing, University of Campinas, 13083-970, Campinas, SP, Brazil*

[2]*Embrapa Agriculture Informatics, 13083-886, Campinas, SP, Brazil*

Keywords: Web Services Choreographies, Interorganizational Business Processes.

Abstract: The main attractiveness of Web services is their capacity to provide interoperability among heterogeneous distributed systems. Increasingly, companies and organizations have adopted Web services as a way to interoperate with their business partners. In such a scenario, Web services choreography can be applied in the specification of interorganizational business processes. However, the dynamic nature of business partnerships requires mechanisms for agile designing and deploying choreographies. In this paper, we present an infrastructure that aims to address the above concern. Our approach aims to reach flexibility by providing mechanisms for sharing, finding and executing choreographies in a friendly manner for the user. We also present a prototype implementation.

1 INTRODUCTION

Increasingly, companies and organizations have adopted Web services to interoperate with their business partners. Web services are applications that are published, located and invoked over the Web. The major attractiveness of this technology is that it is based on XML standards and well known Internet protocols, providing facilities for interoperability among heterogeneous distributed systems. In the business scenario, services are building blocks that are used for the creation of interorganizational business processes.

Web services-based business processes can be specified as Web service choreographies, which are conversations among multiple services that interact in a collaborative fashion to solve a specific problem. We envision three central issues that must be faced in this approach. First, existing proposals for choreography specification languages are not directly executable. Thus, choreography representations need a specific infrastructure to execute them. Second, partners may not be known at specification time; hence, it may be necessary to discover them at execution time. Finally, the mechanisms for choreography specification and deployment must be flexible enough to accomplish the changes of very dynamic business environments. In this paper, we present an infrastructure that aims to address the questions above.

In our infrastructure, interorganizational business processes are described in WS-CDL (Web Services Choreography Description Language), and performed by a set of coordination managers that execute WS-BPEL (Business Process Execution Language for Web Services) compositions. WS-BPEL is used to represent the specific behavior of one choreography partner and to integrate its public behavior with its internal logic. The infrastructure comprises a WS-CDL to WS-BPEL translator that speeds up the generation of WS-BPEL code to be executed by coordination managers It reduces the possibility of errors and inconsistencies in this task. A coordination management mechanism facilitates the specification and deployment of choreographies, keeping choreography designers away from details like partner binding and context control.

A key feature of our infrastructure is a mechanism for sharing choreography descriptions. This mechanism allows organizations to discover choreography descriptions through the use of semantic annotations based on ontologies. We believe that choreography sharing combined with our facilities for deployment and execution of choreographies allows organizations not only to adapt friendly to partnership changes, but also to augment their range of possible partners.

The main contribution of our work is thus an infrastructure geared towards helping choreography designers (our users) in the task of choreography specification. It allows these users to quickly find a choreography that matches their necessities, considering both syntactic and semantic issues. It also supports deployment and execution of the business' portion of a process, without the need to previously know involved partners. This fosters flexibility and independence of each participant in a business environment.

This paper is organized as follows: Section 2 presents some related works; an overview of the proposed infrastructure is showed in Section 3; Sections 4, 5 and 6 detail our approach for sharing choreographies, discovery partners and coordinating the choreography execution; Section 7 presents some implementation issues; finally, in Section 8, we present conclusions and future works.

2 RELATED WORK

Several papers, e.g. (Chung et al., 2004; Caituiro-Monge and Rodrguez-Martinez, 2004; Jung et al., 2004), present solutions for executing Web service compositions. Chung et al. (Chung et al., 2004) present an architecture that uses a grammar to specify business processes. The architecture provides mechanisms to discover services, bind them to processes' taks, and coordinate them. Caituiro-Monge (Caituiro-Monge and Rodrguez-Martinez, 2004) introduces a framework for Web services collaboration in E-Government. In this framework, control data, which is attached to service requests and results, is used to control Web service invocations and message forwarding. Jung (Jung et al., 2004) proposes an interface protocol for incorporating existing workflows into interorganizational business processes and an architecture to execute them. In our solution, like in (Mendling and Hafner, 2005; Diaz et al., 2006), Web service choreographies are executed through a set of executable WS-BPEL plans generated from a WS-CDL choreography description. However, our infrastructure adds a coordination logic on top of WS-BPEL business logic in order to keep the choreography designer away from details such as partner discovery and context control.

Following the Semantic Web trend (Berners-Lee et al., 2001), the Web service community has proposed means to semantically annotate Web services, allowing their automatic discovery and composition. Examples of these efforts are SA-WSDL (Semantic Annotations for WSDL) (W3C, 2007) and OWL-S (Martin et al., 2004). SA-WSDL is a proposal for increasing the expressivity of WSDL through concepts of an ontology. OWL-S is a set of constructs, built on top of OWL, for describing properties and capabilities of Web services in an unambiguous form. In our infrastructure, we propose the use of semantic annotation at choreography description level in order to help users to find descriptions that match their necessities.

3 OVERVIEW

In our infrastructure, choreographies are represented in two levels: *global views* and *coordination plans*. A global view is a WS-CDL document that defines the role of each participant of a business process. It describes, from a global point of view, the interactions among partners, the conditions for interactions to happen and the set of data exchanged during the collaboration. Each global view is assumed to be uniquely identified by a URI. In the same way, each role that is defined in the global view is uniquely identified by a URI.

Coordination plans describe the individual behavior of one of the roles defined in a global view. A coordination plan is composed of a set of instructions that trigger requests to other partners, process requests received from other partners and control the flow of actions for that role. A coordination plan integrates the external logic defined by a global view with the internal logic of an individual partner. We have adopted WS-BPEL for representing coordination plans.

The constraints defined in a global view enable an organization to generate a coordination plan that reflects its behavior in the corresponding business process. A global view ensures that coordination plans for distinct roles, generated from the same global view, are interoperable. However, global views do not define a partner's specific internal logic. So, an organization that wants to play a role in a business process described by a global view must generate a coordination plan skeleton from that global view and then customize it, including the organization's internal logic.

Figure 1 shows our infrastructure. It is composed of four elements: the *choreography repository*, the *plan generator*, the *coordination manager*, and the *participant repository*.

The choreography repository stores and shares choreography descriptions. It allows users to publish and discover choreography descriptions that fulfill their business requirements. Any organization that wants to participate in a choreography can access a choreography repository to obtain the corresponding description. Such a description is used as the input of a plan generator that creates a coordination plan skeleton. If required, a programmer can customize the

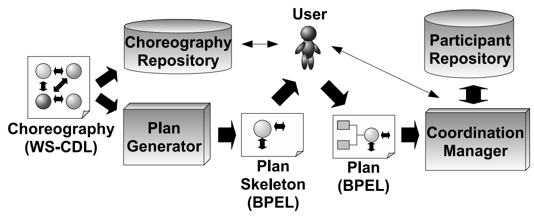

Figure 1: Infrastructure overview.

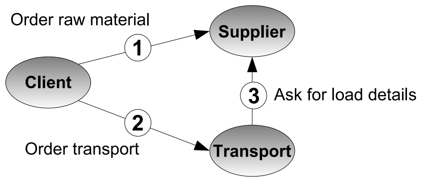

Figure 2: Example of a simplified choreography for supplying raw material.

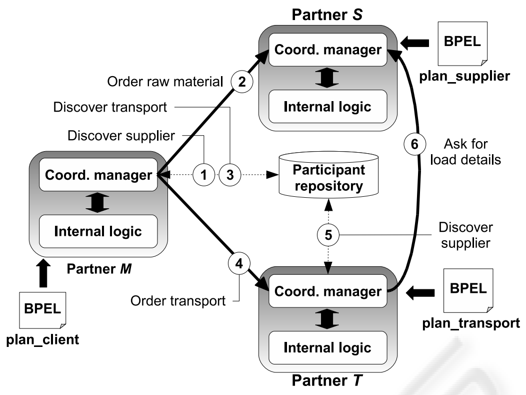

Figure 3: Example execution.

skeleton to include the organization's internal logic. Each generated plan implements the logic needed for playing a role specified by the global view.

Once the coordination plan is generated, it can be interpreted and executed by a coordination manager. The latter is also responsible for discovering choreography partners. The partner discovery involves the use of the participant repository that allows coordination managers (i) to discover a coordination manager that is able to perform a given choreography role or (ii) to resolve the endpoint reference of a coordination manager that is already performing such a role.

To illustrate the use of our infrastructure, consider a simple scenario: a manufactor *M* needs to implement a business process to order raw material. Therefore, *M* searches in a choreography repository for a global view that meets its requirements. Assume that *M* chose the choreography described in Figure 2.

The choreography example in Figure 2 defines three roles: *client*, *supplier*, and *transport*. The *client* partner (1) invokes a specific Web service operation to order raw material from *supplier*. Next, *client* (2) orders transport service from *transport*. To perform its portion of the process, *transport* needs to ask *supplier* for load details (3). From the choreography global view, *M* generates the coordination plan (*plan-client*) for role *client* and deploys this plan in its coordination manager.

To execute *plan-client*, *M*'s coordination manager accesses a participant repository to discover partners that are able to execute coordination plans relative to roles *supplier* and *transport*. After having discovered appropriate partners (consider *S* and *T* for roles *supplier* and *transport*, respectively), *M*'s coordination manager can invoke their operations, according to constraints defined by its plan. Partner *T*'s coordination manager, in its turn, needs to interact with the participant repository to obtain the endpoint reference of the specific partner that is playing the *supplier* role in the current instance of the choreography (*S*). Figure 3 shows an overview of the choreography execution.

4 SHARING AND REUSING CHOREOGRAPHIES

Our infrastructure supports choreography sharing and reuse by means of the choreography repository. This component allows business partners to publish the descriptions of choreographies they are able to participate in. Once descriptions are published, other organizations can access the repository and discover choreographies that fulfill their requirements. Thus, they can generate their coordination plans and interact with other choreography participants.

In order to help users in choreography discovery, we propose the addition of semantic annotations to the WS-CDL descriptions. Our annotations associate the roles and relationships specified by WS-CDL with concepts of an ontology. Figure 4 illustrates our semantic annotation approach. Ontology terms are in grey boxes and the choreography is specified in the WS-CDL box at the top. In the figure, the annotations denote, for example, that role "A" is played by "SupplierA", which supplies products of type "ProductA".

The semantic annotation allows users to discover choreography descriptions based on roles and relationships among these roles. Examples of queries are:

- Return choreographies that include a supplier of "ProductA" and a manufactory that produces "ProductB";

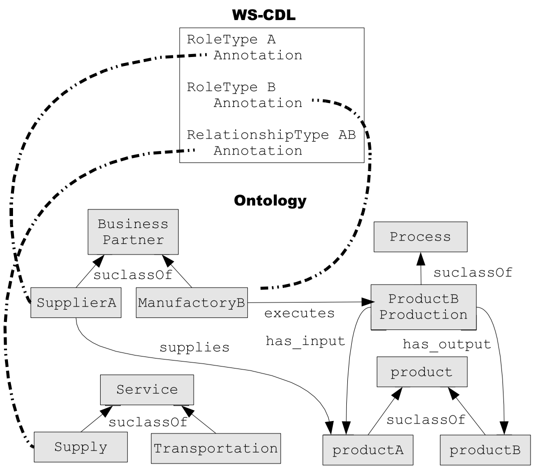

Figure 4: Semantic annotation.

- Return choreographies that include two participants related by a “Supply” service.

Our approach considers the similarity of concepts and properties provided by different ontologies. So, one can search for choreographies using as basis one’s particular ontology, without knowing the specific ontology used in WS-CDL annotations. This is done through a service developed by (Daltio and Medeiros, 2007), which provides means for ontology alignment and query.

Our semantic annotation can be complemented by the use of SA-WSDL. The semantic information added by SA-WSDL can be used for higher granularity queries involving, for example, details about the SA-WSDL operations. Moreover, the adoption of SA-WSDL may facilitate the automatic generation of coordination plans.

5 RESOLVING PARTNERS

Our infrastructure allows a choreography participant to resolve partners at execution time with little effort from choreography designers. Global views and coordination plans can be designed at a partner role level, regardless of who will play each role. The logic for resolving partner’s endpoint references is implemented by the coordination manager and is almost transparent for designers. The coordination manager can obtain a partner endpoint reference in three ways:

- Automatic partner discovery: the coordination manager searches for partners in a participant repository using the identifier of the role (*choreographyId* + *roleId*) the partner must be able to play (e.g. interactions involving partner *M* in Figure 3).
- Automatic instance discovery: the coordination manager searches in a participant repository for the partner that is already playing a role in a specific choreography instance (e.g. interaction among partners *T* and *S* in Figure 3). The key used for searching is the combination of the identifiers of the role and the choreography instance.
- Static binding: the endpoint reference of the partner is defined at deployment time of the coordination plan.

The coordination manager maintains a configuration file for each coordination plan that it is able to execute. This configuration file, which is created by the plan designer, defines how to resolve the endpoint reference for each partner.

6 COORDINATION MANAGEMENT

An important issue when executing choreographies is to preserve the context of their instances to ensure that messages arrive to the right instances of coordination plans. Coordination managers provide a management abstraction that controls context, in a level that is hidden from designers. It means that choreography designers can design global views and coordination plans without concerning themselves about context control.

Interactions between two coordination managers are preceded by a connection phase. In this phase, coordination managers agree on the choreography to be followed, the role to be played by each one, and the context identifier that distinguishes the choreography instance. To connect to a partner, the coordination manager resolves the partner endpoint reference (as shown in Section 5) and sends it a connection proposal. When a coordination manager receives a connection proposal and it is not already participating in the related choreography instance, it starts the appropriate coordination plan and registers the plan instance in the participant repository.

All application messages – those exchanged by coordination plan instances – carry, in their headers, data used in context control and message delivery. Such data includes identifiers for: choreography (*ChoreographyID*), choreography instance (CIID), sender partner (SRID - *Sender roleID*), and destination partner (DRID - *Destination roleID*). These data uniquely identify a connection between two coordination managers. Through them, the sender coordination manager can retrieve the endpoint reference of the destination one from a table of connections. In other side, the destination coordination manager can deliver the message to the right coordination plan instance.

7 IMPLEMENTATION ISSUES

In order to validate our infrastructure, we have implemented a prototype. The infrastructure components were implemented using Java language. In this section we present some issues about our implementation experience.

7.1 Choreography Repository

Our repository is to be implemented as a Web service that when it is necessary invokes the Aondê ontology service of (Daltio and Medeiros, 2007). When a user sends a request for a choreography based on semantic annotations, the system checks against all choreographies to investigate (a) which choreographies have these exact annotations, and (b) if none is found, which choreographies would have similar or equivalent annotations. The user provides a reference to an input ontology and a list of terms in this ontology that annotate choreographies. Let O be the ontology used to annotate the choreographies in a repository, and $< O_N, \{T_i\} >$ the user input – source ontology, and terms of interest in this ontology.

If $O_N = O$, then step (a) is executed, i.e., looking for all choreographies that have been annotated with some T_i. All such choreographies are returned to the user. If $O_N \neq O$, then step (b) is executed. The Aondê service is invoked with a request to align O_N and O. This request returns a new (aligned) ontology. An ontology alignment is the process of determining semantic correspondences between concepts of two ontologies, generating a third ontology to represent them without modifying the original ones. From the alignments, the system will check which terms $\{T_1, ..., T_n\}$ in O_N are equivalent with terms $\{O_1, ..., O_n\}$ in O. Choreographies annotated with terms in $\{O_1, ..., O_n\}$ are returned to the users.

7.2 Coordination Manager

Figure 5 shows the coordination manager architecture. It is composed of two main components: the execution engine and the management module. The execution engine interprets and executes coordination plans. It issues messages to the management module

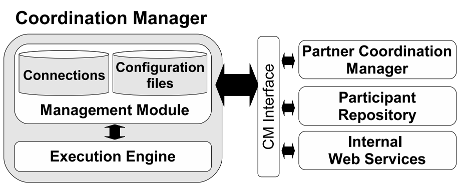

Figure 5: Coordination Manager Architecture.

and processes messages that are delivered by it, according to the plan flow control. We have adopted a conventional BPEL engine[1] in our prototype.

The management module adds the management logic to the logic defined by coordination plans. This module is responsible for the interaction with other coordination managers, participant repositories, and Web services that represent the internal logic of the organization. The management module implements the logic for resolving partner endpoint references, managing connections, forwarding messages that are generated by the BPEL engine to destination partners, and delivering messages that are received from partners.

7.3 Participant Repository

The participant repository was implemented as a simple Web service that is able to register and discover endpoint references. We have implemented only the operations that are needed to validate our automatic discovery mechanisms and test the coordination manager functionalities.

7.4 Plan Generator

We have specified a set of rules for translating WS-CDL global views to BPEL coordination plans. The central idea of this translation is to generate a coordination plan skeleton for each role defined in the global view. For each WS-CDL element that is related to a specific role, a BPEL mapping is added in the correspondent skeleton. Although our plan generator is specific to our infrastructure, the basic ideas under the translation rules are applicable to a generic WS-CDL to BPEL translation. Table 1 summarizes the main rules for mapping WS-CDL elements to BPEL elements.

8 CONCLUSIONS

We believe that an important issue in SOA environments is the flexibility to agilely adapt to business

[1]ActiveBpel, available in http://www.activebpel.org/.

Table 1: Main rules for WS-CDL/BPEL elements mapping.

WS-CDL Element	BPEL Mapping
cdl:variable	bpel:variable
cdl:getVariable	bpel:getVariableData
cdl:globalizedTrigger	Subexpression related to the specific role
cdl:sequence	bpel:sequence
cdl:parallel	bpel:flow
cdl:choice	bpel:switch/bpel:cas
cdl:workunit	Combination of bpel:while and bpel:switch/bpel:case
cdl:silentAction	bpel:empty
cdl:assign	bpel:assign
cdl:interaction	Combination of bpel:invoke and bpel:receive

changes. Our solution aims to meet such flexibility by providing mechanisms for sharing and finding choreography descriptions, based on semantic annotations, and deploying and executing them with small programming effort. The prototype implementation shows our infrastructure's feasibility.

A contribution of our infrastructure is a mechanism that allows automatic partner discovery. This feature allows business processes to be executed by distinct configurations of partners. Another contribution is the separation between business logic and coordination management. Choreography designers do not need to worry about the way context control will be done or which choreography partners are connected to the system. This transparency facilitates the choreography design.

Our infrastructure also facilitates the deployment and execution of choreographies. The plan generator speeds up the generation of coordination plans; besides, it reduces the possibility of errors and inconsistences in this task.

Issues that will be addressed as future work include: (i) the improvement of WS-CDL semantic annotations to provide more refined queries; (ii) the automatic customization of plan skeletons based on SA-WSDL semantic annotations; (iii) the improvement of the coordination manager architecture to deal with partner authentication, security and authorization issues; and (iv) the provision of fault tolerance.

ACKNOWLEDGEMENTS

The authors would like to thank FAPESP (07/56423-6), CNPq (472810/2006-5), CNPq WebMaps 2 project, and CAPES for the financial support.

REFERENCES

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The semantic web. *Scientific American*, 284(5):34–43.

Caituiro-Monge, H. and Rodrguez-Martinez, M. (2004). Net Traveler: A Framework for Autonomic Web Services Collaboration, Orchestration and Choreography in E-Government Information Systems. In *ICWS '04: Proceedings of the IEEE International Conference on Web Services (ICWS'04)*, pages 2–10, Washington, DC, USA. IEEE Computer Society.

Chung, M. J. et al. (2004). A framework for collaborative product commerce using web services. In *Proceedings of the IEEE International Conference on Web Services (ICWS'04)*, pages 52–60.

Daltio, J. and Medeiros, C. B. (2007). Aondê: An ontology web service for interoperability across biodiversity information systems. Master's thesis, Institute of Computing - Unicamp, Campinas-SP, Brazil. (in Portuguese).

Diaz, G., M. E. Cambronero, J. J. Pardo, V. V., and Cuartero, F. (2006). Automatic generation of correct web services choreographies and orchestrations with model checking techniques. In *Telecommunications, 2006. AICT-ICIW '06. International Conference on Internet and Web Applications and Services/Advanced International Conference on*, pages 186–192.

Jung, J., Hur, W., Kang, S.-H., and Kim, H. (2004). Business Process Choreography for B2B Collaboration. *IEEE Internet Computing*, 8(1):37–45.

Martin, D. et al. (2004). Owl-s: Semantic markup for web services. http://www.daml.org/services/owl-s/1.1/overview/.

Mendling, J. and Hafner, M. (2005). From inter-organizational workflows to process execution: Generating BPEL from WS-CDL. *Proceedings of OTM 2005 Workshops. Lecture Notes in Computer Science*, 3762:506–515.

W3C (2007). Semantic annotations for wsdl and xml schema - w3c recommendation. http://www.w3.org/2002/ws/sawsdl/spec/.