2009

From Service Conversation Models to WS-CDL

Karthikeyan Umapathy
University of North Florida, k.umapathy@unf.edu

From Service Conversation Models to WS-CDL

Karthikeyan Umapathy
University of North Florida
k.umapathy@unf.edu

ABSTRACT

Changing business environments are forcing organizations to develop flexible and adaptable enterprise systems. To accomplish this and to solve associated systems integration issues, many are moving towards web service technology. Two key ingredients of web services based solution are service composition and service choreography. While there has been lot of advancement in respect to service composition, service choreography rather largely remains an open problem. WS-CDL specification is considered to be a candidate standard for service choreography; however, consensus on support mechanisms to develop conversation models depicting peer-to-peer interactions are yet to be reached. In this paper, we develop an approach as well required heuristics for identifying service interaction patterns from business process models and using them to develop conversation models. We provide detailed discussion on heuristics, illustrate our approach through an example, as well as indicate how these conversation models can be used for generating WS-CDL specifications.

Keywords

Service choreographies, service conversations, conversation models, service interaction patterns, business process models, WS-CDL.

INTRODUCTION

In order to effectively participate in the competitive global market, the organizations need to be agile and flexible (Agarwal and Sambamurthy 2002). A key strategy that organizations adopt towards this goal is integration of their various disparate information systems (Markus and Tanis 2000). Task of integrating multiple information systems to enable data transfer among them and to support common business processes is called systems integration (Lee et al. 2003). In order to develop complete integration solution, designers have to develop business processes that reflect requirements of the integrated system and develop integration solutions that can be implemented (Umapathy et al. 2008).

With respect to implementation platform, web service is the current most preferred technology for implementing integration solutions (Iyer et al. 2003), because it permits integration of disparate systems irrespective of their development platform. Realization of systems integration solutions, then, requires an approach that will translate integration solutions into mechanisms that can be implemented using web services technology platform. In order to provide complete solution, such an approach must generate web service specifications such as service compositions and service choreographies (also referred as service conversations), given that each system is represented as a web service. Service composition specifications provides protocols for describing order in which services would be executed, while service choreography specifications provides protocols for describing how to maintain long-running conversations during execution of composite services.

Web service composition specifications such as WS-BPEL (Web Services Business Process Execution Language) (WS-BPEL 2007) provides protocol for aggregating individual services and executing them in a logical order to achieve certain business goals. With respect to generating web service composition specification, designers can utilize various commercial tools such as ActiveBPEL Designer, Intalio Designer, and Oracle BPEL Process Manager. For this purpose, designers can also use relevant research work on generating service composition specifications based on business processes (Ouyang et al. 2007).

Web service choreography specifications such as WS-CDL (Web Services Choreography Description Language) (WS-CDL 2005) provides a standardized way for describing interactions among services as an ordered sequence of message exchanges geared towards a business goal. It is critical for web services to support long-running conversations that are crucial for performance and coordination of business processes that involves multiple participants and/or cross organizational boundaries (Papazoglou 2002). Service choreography specifications supports long-running conversations through mechanisms that provide a loosely coupled, peer-to-peer interaction model where services exchange several correlated messages (Ardissono et al. 2003; Peltz 2003). Thus, service choreography describes conversation model that participating services must adhere to and keep track of "who is allowed (or expected) to send messages to whom and in what order" (Hohpe 2007). Complete and comprehensive service choreographies are critical for achieving communication, coordination,

and collaboration among services for successful execution of composite services, particularly, in the context of cross-organizational processes which involves long-running interactions.

While importance of service choreographies is well established, it is not highly concentrated in web service research domain.WS-CDL specification is only a candidate standard for service choreography, there are several factors inhibit their progress to become the standard (Barros et al. 2005b). Key issues inhibit advancement of service choreography are lack of formal grounding to link between Web Service Definition Language (WSDL), WS-BPEL, and WS-CDL; support mechanisms to develop conversation models; and lack of adequate tools and subsequent evaluation demonstrating their utility. Above issues indicate that there are several open research questions in the context of service choreographies.

In this paper, we address the following: *How to develop conversation model that represents interaction logic for multi-party services involved in a composite service?* Anticipated benefit of solutions to above question would be mechanisms that can capture series of ordered message exchanges among participating services to achieve their intended goals. Such mechanisms can be helpful for tool developers to guide their users in developing WS-CDL specifications that are coherent to a given problem. Therefore, the objective of this research is to address the above research question by developing an approach that would provide support for generating service choreography specifications adhering to interaction logic specified in the business process.

CONVERSATION MODELS

Service choreography specifications describes multi-step exchange of correlated messages that need to be performed by participating services in order to achieve a business goal (Ardissono et al. 2003). Services exchange series of messages within an explicit conversational context that is established on initial contact, maintained for the duration of the conversation, and discarded at the end (Hanson et al. 2002). Each new message in a conversation is interpreted in relation to the messages previously exchanged in that conversation. Therefore, conversations among services can be represented as a "conversation model" that describes the reactive behavior of web services based on a "conversation context" (Benatallah et al. 2003; Hanson et al. 2002), which essentially forms a state-transition model. For each conversation state, the model describes set of services that are part of an interaction and messages that need to be exchanged among them. Therefore, a prerequisite for generating web service choreography specifications is development of comprehensive conversation model depicting peer-to-peer interactions among participating services. However, developing such a comprehensive conversation model is not a trivial task.

One possible strategy is to identify interactions among participants in the business process and assemble them in the specified order. An interaction can be defined as a sequence of adjacent tasks that implicates two or more participants (Aakhus 2004), including the roles 'initiator' and 'responder'. Thus, the heuristic change in participant is useful for identifying an 'interaction' in the business process. Moreover, each interaction would provide details on messages to be exchanged by participating services. Each message exchange in the indentified interactions can be represented as a state-transition in the conversation model. Thus, assemblage of identified interactions in accordance to order specified in the business processes, thus, would form a conversation model. Figure 1 shows an example conversation model for loan request process depicted as a state-transition model. Business process description of the loan request process is provided later in the illustration section.

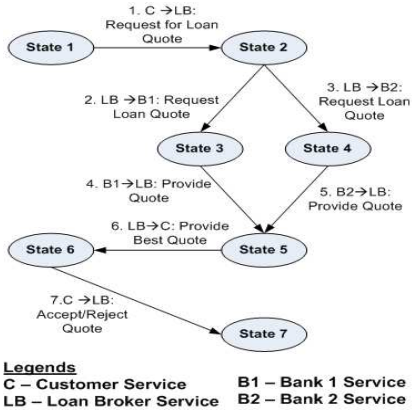

Figure 1. Conversation model and interactions model for loan request process

Interactions identified from business process models would only provide bilateral interaction details such as whether there is one-to-one, one-to-many, or many-to-one associations between participating services. These interactions do not provide

details of multilateral, concurrent, and interrelated interactions that typically would arise when services engage in conversation. Therefore, interactions identified from business process models may not necessarily reflect peer-to-peer interactions between services. This can be ensured by comparing interaction indentified from business process model against to service interaction patterns (Barros et al. 2005a). In the section we provide brief discussion on service interaction pattern and in the subsequent section, we provide details on how interactions identified from business process models can be compared against to service interaction patterns.

SERVICE INTERACTION PATTERNS

Barros et al. (2005a) describe a collection of patterns of service interactions for the context multi-party collaborative environments. They developed these patterns primarily to be applied for development of web service specifications. They derive these patterns based on prior works, real-world cross-organization transaction processes, use case scenarios relevant to service composition and service choreographies, and generic scenarios from other industry standards such as RosettaNet Partner Interface Protocols. They present 13 service interaction patterns: send, receive, send/receive, racing incoming messages, one-to-many send, one-from-many receive, one-to-many send/receive, multi-responses, contingent requests, atomic multicast notification, request with referral, relayed request, and dynamic routing. Detailed description of these service interaction patterns are available elsewhere (Barros et al. 2005a). In the next section, we develop heuristics for comparing interactions identified from business process models against to service interaction patterns.

HEURISTICS FOR IDENTIFYING SERVICE INTERACTION PATTERNS

Business process models provide details on control flow of business tasks, set of rules for task execution, and process events. Business Process Modeling Notation (BPMN) is considered to be standard for describing business processes. Each BPMN elements used in the business process models provides rich information about the tasks involved in the interactions. Thus, we use combination of elementary elements of interactions, BPMN notations of tasks involved in the interaction, and service interaction pattern descriptions to develop heuristics for determining whether an interaction identified from business process model resembles either of the service interaction patterns. Below, we provide discussion on heuristics for identifying service interaction patterns along with suggested state-transition models to depict their peer-to-peer interaction model (see figure 2). In the next section, we illustrate utility of the heuristics by applying on loan request business process.

Send

Send pattern represents one-to-one interaction between two services, where in, first service sends a message to the second service. Typical scenario where send pattern can be applied would involve interactions with initiator task represented using start or intermediate message event BPMN element. However, send pattern can be applied to any one-to-one interaction, where initiator of the interaction sends a message to responder. Thus, send interaction pattern can be identified using below heuristics:

> Interaction *ind* with one-to-one association between initiator *X* and responder *Y*

Receive

Receive pattern represents one-to-one interaction between two services, where in, second service receives a message from the first service. Typical scenario where receive pattern can be applied would involve interactions with responder task represented using end or intermediate message event BPMN element. However, receive pattern can be applied to any one-to-one interaction, where responder of the interaction receives a message from initiator. Thus, both send and receive pattern contain peer-to-peer interaction model, where in, one service sends a message and second service receives. Thus, heuristics and state-transition model of send and receive pattern are same.

Send/receive

Send/receive pattern represents two-way one-to-one interaction between two services, where in, first service sends a message to second service and second service subsequently sends a response message to first service. Typical scenario where send/receive pattern can be applied would involve interactions with request/response message exchanges. Thus, send/receive interaction pattern can be identified using below heuristics:

> Interaction *ind1* with one-to-one association between initiator *X1* and responder *Y1* AND Interaction *ind2* with one-to-one association between initiator *X2* and responder *Y2* AND Initiator *X1* of *ind1* and responder *Y2* of *ind2* are same AND Responders *Y1* of *ind1* and initiator *X2* of *ind2* are same.

Racing incoming messages

Racing incoming messages pattern represents many-to-one interaction, where in, N number of services sends a message (may be structurally different to each other) to a service. Typical scenario where racing incoming message pattern can be applied would involve interactions where only one of N initiators is expect to send a message to the responder. Thus, responder task in the interaction should be represented using XOR Gateway BPMN element. Thus, racing incoming messages interaction pattern can be identified using below heuristics:

> Interaction *ind* with many-to-one association between N number initiator(s) and responder *Y* AND BPMN element used for representing responder *Y* is "XOR Gateway."

One-to-many send

One-to-many send pattern represents one-to-many interaction, where in, a service sends a message to N different services simultaneously. Typical scenario where one-to-many send pattern can be applied would involve interactions with publish/subscribe message exchanges where initiator sends a message to all N number of responders at same time. One may or not use parallel gateway to represent parallel flow scenario in business process model as they are optional; therefore, our heuristics does not use Parallel Gateway as a condition for this pattern. Thus, one-to-many send interaction pattern can be identified using below heuristics:

> Interaction *ind* with one-to-many association between initiator *X* and N number of responders.

One-from-many receive

One-from-many receive pattern represents many-to-one interaction, where in, a service receives logically related messages from N number of services. Typical scenario where one-from-many receive pattern can be applied would involve interactions with multilateral transmission message to a single service, which begins its processing on the messages only after it receives all N messages from N services. In respect to business process models, appropriate BPMN element to represent scenario of synchronizing all parallel task is Parallel Gateway. Thus, one-to-many receive interaction pattern can be identified using below heuristics:

> Interaction *ind* with many-to-one association between N number initiators and responder *Y* AND BPMN element used for representing responder *Y* is "Parallel Gateway."

One-to-many send/receive

One-to-many send/receive pattern involves a service sending a request message to N number services and it expects to receive response message from them. Typical scenario where one-to-many send/receive pattern can be applied would involve two multilateral interactions. Specifically one-to-many send interaction pattern immediately followed by one-from-many receive interaction pattern, where in, initiator of the first interaction is the responder of the second interaction and N responders of the first interaction are the initiators of the second interaction. Thus, one-to-many send/receive interaction pattern can be identified using below heuristics:

> Interaction *ind1* with one-to-many association between initiator *X1* and N number of responders AND interaction *ind2* with many-to-one association between N number of initiators and responder *Y2* AND Initiator *X1* of *ind1* and responder *Y2* of *ind2* are same AND N responders of *ind1* and N initiators *ind2* are same AND BPMN element used for representing responder *Y2* is "Parallel Gateway."

Multi-responses

Multi-responses pattern involves a service A which sends a request message to another service B and subsequently receives N number of response messages from service B. Typical scenario where multi-responses pattern can be applied would involve combination of two interactions. Specifically a send pattern subsequently followed by one-from-many pattern, where in, all initiators is same. Unlike one-from-many pattern, N messages are not expected to be received or send synchronously. Thus, multi-responses interaction pattern can be identified using below heuristics:

> Interaction *ind1* with one-to-one association between initiator *X1* and responder *Y1* AND Interaction *ind2* with many-to-one association between N number of initiators and responder *Y2* AND Initiator *X1* of *ind1* and responder *Y1* of *ind2* are same AND All N initiators of *ind2* and responder Y1 of *ind1* are same.

Contingent requests

Contingent requests pattern involves a service A which sends a request message to service B; if service B does not respond within a timeframe then service A sends the request message to another service and so on. Typical scenario where contingent requests pattern is applied would involve two multilateral interactions. First interaction involves one-to-many association where the initiator service prepares an ordered list of interested services, and sends a request message to the first service in the order. Second interaction involves many-to-one association where the contacted service sends response message. If the service fails to respond in timeframe then second service in the ordered list is contacted. Thus, any type of gateway element can be used for representing task of initiator of the first interaction while timer element would be used for representing responders who need to send response message within a timeframe. Thus, contingent requests interaction pattern can be identified using below heuristics:

> Interaction *ind1* with one-to-many association between initiator *X1* and N number of responders AND None of N responders of *ind1* are same AND BPMN element used for representing initiator *X1* is "Gateway" AND For all N responders of *ind1*, BPMN element used is "Timer" AND Interaction *ind2* with many-to-one association between N number of initiators and responder *Y2* AND Initiator *X1* of *ind1* and responder *Y1* of *ind2* are same AND N initiators of *ind2* and N responders of *ind1* are same.

Atomic multicast notification

Atomic multicast notification pattern involves a service A which sends a message containing certain commands to N number of services which must send their acceptance message within a timeframe. Typical scenario where atomic multicast notification pattern is applied would involve two multilateral interactions. First interaction involves one-to-many association where initiator sends a message to N responders, where in, a minimum number of responses are expected to be received in the second interaction which involves many-to-one association. Thus, responders of the first interaction would be represented using timer BPMN element. Thus, atomic multicast notification requests interaction pattern can be identified using below heuristics:

> Interaction *ind1* with one-to-many association between initiator *X1* and N number of responders AND None of N responders of *ind1* are same AND For all N responders of *ind1*, BPMN element used is "Timer" AND Interaction *ind2* with many-to-one association between N number of initiators and responder *Y2* AND Initiator *X1* of *ind1* and responder *Y1* of *ind2* are same AND All N initiators of *ind2* and N responders of *ind1* are same.

Request with referral

Request with referral pattern involves a service A which sends a request message to service B which must send response message to N other services. Typical scenario where request with referral pattern is applied would involve interactions where initiator sends a request message to responder indicating that follow-up response should be addressed to N other services. Thus, this pattern involves two interactions, first one as send pattern and second as one-to-many send pattern, where in, responder of the first interaction is the initiator of the second interaction. Number of N responder services to be contacted can be decided based on certain evaluation conditions. A gateway element would be used for representing initiator of the second interaction. Thus, request with referral interaction pattern can be identified using below heuristics:

> Interaction *ind1* with one-to-one association between initiator *X1* and responder *Y1* AND Interaction *ind2* with one-to-many association between initiator *X2* and N number of responders AND Responder *Y1* of *ind1* and initiator *X2* of *ind2* are same AND BPMN element used for representing initiator *X2* of *ind2* is "Gateway."

Relayed request

Relayed request pattern involves a service A which sends a request message to service B which in turn delegates N other services to perform the requested task. Relayed request pattern is similar to request with referral pattern, however, with relayed request pattern subsequent N number of services is determined by service B. Thus, relayed request interaction pattern can be identified using below heuristics:

> Interaction *ind1* with one-to-one association between initiator *X1* and responder *Y1* AND Interaction *ind2* with one-to-many association between initiator *X2* and N number of responders AND Responder *Y1* of *ind1* and initiator *X2* of *ind2* are same.

Dynamic routing

Dynamic routing pattern involves a service which sends a message to N other services based on certain dynamic condition. Typical scenario where dynamic routing pattern is applied involves interaction with one-to-many associations where initiator service decides dynamically based on runtime data on number of services to which it needs to send a message. Complex gateway would be appropriate BPMN element to represent initiator in these scenarios. Thus, dynamic routing interaction pattern can be identified using below heuristics:

> Interaction *ind* with one-to-many association between initiator *X* and N number of responders AND BPMN element used for representing initiator *X* of *ind* is "Complex Gateway."

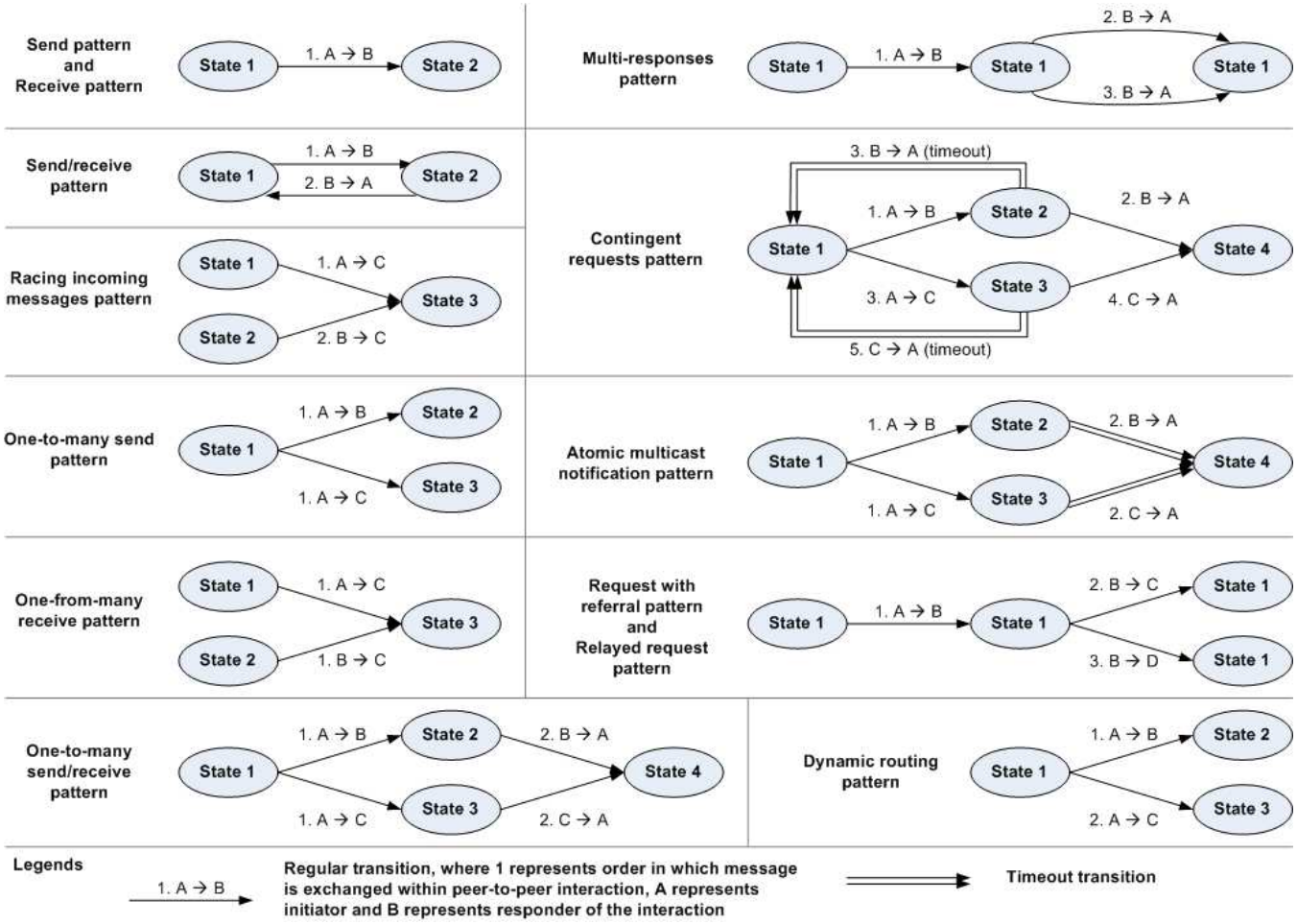

Figure 2. State-transition models for service interaction patterns

ILLUSTRATION

We use the loan request processing business process adapted from Hohpe and Woolf (Hohpe and Woolf 2004). In this process, a customer goes online to the loan broker's website and enters several pieces of information including name, contact details, and desired loan amount. The loan broker receives the request and forwards it to two different banks to obtain their quotes. Once the loan broker receives quote from all banks it selects the best quote and sends the best quote to the customer. Customer then makes decision to accept or reject the loan quote and sends an accept/reject message to loan broker. Figure 3 shows the loan request process drawn using BPMN notation.

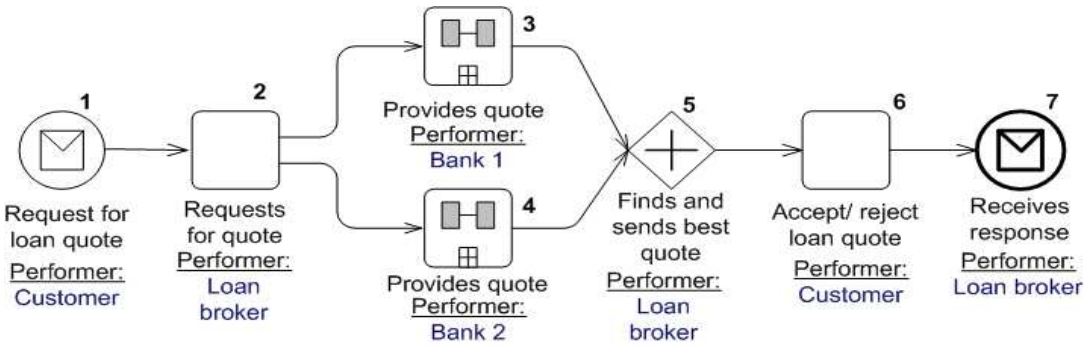

Figure 3. Loan request business process

In this process, 5 interactions can be identified. The tasks are marked in the figure and the interactions are listed in the table 1. First interaction involves one-to-one association between Customer (Service C) and Loan Broker (Service LB), where in, process is initiated when Service C submits a request loan quote. Thus, first interaction closely resembles send service interaction pattern. Second interaction has one-to-many association between Service LB and two Banks (Service B1 and Service B2); where in, Service LB sends a request for loan quote message to two other services. Thus, for now, second interaction closely resembles one-to-many send pattern. Third interaction involves many-to-one association between Service B1, Service B2, and Service LB, where in, Service LB receives quote response messages from the two bank services. Thus, for now, third interaction closely resembles one-from-many receive pattern. However, when both second and third interactions are consider together, they closely resemble one-to-many send/receive pattern. Fourth interaction involves one-to-one association between Service LB and Service C, where in, Service LB sends the best quote message to Service C. Thus, fourth interaction closely resembles send pattern. Fifth interaction involves one-to-one association between Service C and Service LB, where in, process is terminated upon when Service LB receives message from Service C. Thus, fifth interaction closely resembles receive pattern. However, when both fourth and fifth interactions are considered together, they closely resemble send/receive pattern. Table 1 provides summary of appropriate service interaction patterns for each identified interactions in the loan request process. Thus, loan request process contains only 3 peer-to-peer service interaction models.

Interactions	Tasks	Performer	Role	Service Interaction Pattern
1	1	Customer	Initiator	Send pattern
	2	Loan broker	Responder	
2	2	Loan broker	Initiator	One-to-many send/receive pattern
	3	Bank 1	Responder	
	4	Bank 2	Responder	
3	3	Bank 1	Initiator	
	4	Bank 2	Initiator	
	5	Loan broker	Responder	
4	5	Loan broker	Initiator	Send/receive pattern
	6	Customer	Responder	
5	6	Customer	Initiator	
	7	Loan broker	Responder	

Table 1. Service interaction patterns for loan request process

Service interaction patterns identified through application of heuristics can be arranged in the interaction order specified in the business process model to form conversation model. Figure 4 provides conversation model for loan request process model after applying heuristics on the identified interactions. Notable differences in comparison to figure 1 (which reflects conversation model prior to applying heuristics) are changes in order of message exchanges due to use of one-to-many send/receive pattern and one less state to be managed due to use of send/receive pattern.

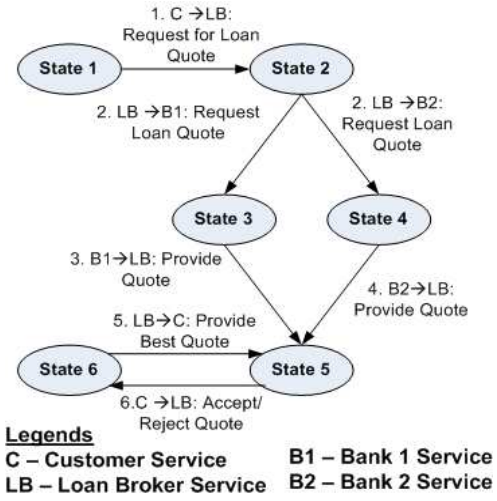

Figure 4. Conversation model for loan request process after applying heuristics

GENERATING WEB SERVICE CHOREOGRAPHY (WS-CDL) SPECIFICATION

Developing conversation model based on business process models is only half of the solution. In order to develop complete solution, conversation model should be converted into WS-CDL specification document. For that purpose, first we store conversation model using conversation policy XML (cpXML) which is an XML schema used for describing conversation models for web services (Hanson et al. 2002). cpXML follows state-transition model to describe conversation model, includes required schema elements for conversation model in the context web services such as specifying state information, transition information, details on messages exchanged, timeout information, order of message exchanges, and provisions for handling sub-conversations. After storing conversation model using cpXML, WS-CDL specification can be generated using it. Even though both cpXML and WS-CDL are XML documents, we cannot use simple XML style sheet transformations to generate WS-CDL specification. WS-CDL schema contains complex entities which can be primarily derived from cpXML and may need some information from business process models depending upon the problem context. Due to space limitations, we cannot provide detailed discussion on how elements of cpXML can be used for deriving required information for generating WS-CDL specification. However, we provide schema mapping for cpXML and WS-CDL specifications in figure 4 to indicate potential sources for deriving required information.

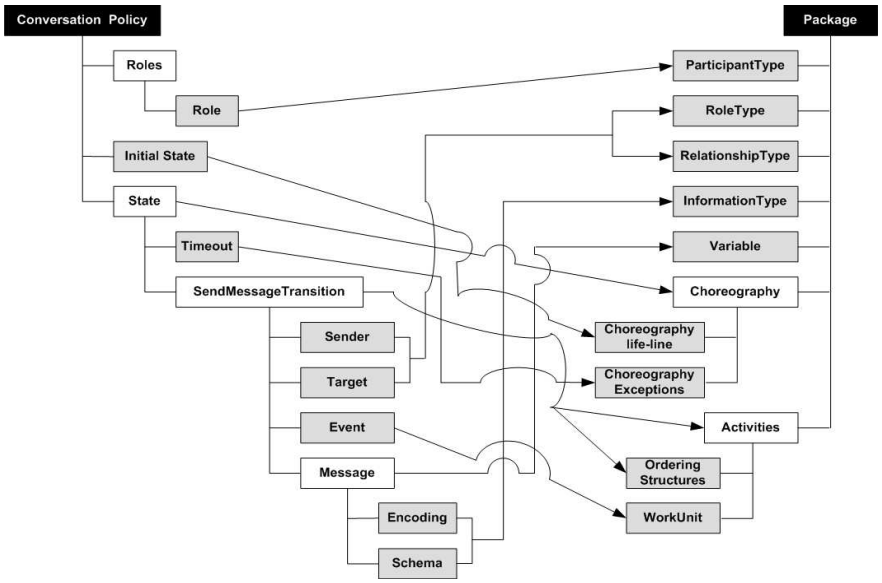

Figure 5. Schema mapping of cpXML and WS-CDL

CONCLUSION

Advancement of service choreographies is stalled due to various reasons including lack of linkages to other relevant specifications, lack of adequate tool support, and lack of mechanisms to develop conversation models depicting coherent ordered message exchanges. In this research, we focus on providing support for developing conversation models that captures peer-to-peer interaction logic among participating services required for generation of WS-CDL specifications. We argue that such support mechanisms can be helpful for tool developers to generate WS-CDL specifications that are complete and comprehensive. Better tools would increase adoption as well as help in progression of WS-CDL as the recommended standard.

In this paper, we provide an approach to develop WS-CDL specification by taking advantage of service interaction patterns and business process models represented using BPMN. We, first, indicate importance of conversation models for developing service choreographies. Even though business process models is a good starting point to develop conversation model, interactions identified from business process do not necessarily reflect peer-to-peer interaction model required for service conversation models. Service interaction pattern provides list of peer-to-peer interaction models that can be applied in the context of conversations among services. We develop heuristics using interactions from business process and BPMN elements used to represent interactions to identify appropriate service interaction patterns. Identified service interaction patterns are then arranged in sequence described in the business process models to develop conversation model. cpXML schema is used to store conversation model, which is later used for generating WS-CDL specification. We illustrate application of our heuristics through loan request business process by applying it to develop conversation model.

The approach described in this paper, is being implemented as a software tool with following modules: BPMN editor with drag and drop facility to develop business process models, identifying interactions from business process models, applying heuristics to identify appropriate service interaction patterns, developing conversation model as cpXML, and generating WS-CDL specifications. Software tool also uses a forward-chain reasoner for identifying service interaction patterns using described heuristics, where in, interactions and business tasks identified from business process models are declared as facts and heuristics are declared as rules. Software tool is being developed in Java platform along with help of XMLBeams.

There are some associated caveats with this research, as it assumes that business process models developed are complete, which may not necessarily be true. Research from workflow patterns (van der Aalst et al. 2003) can be useful in determining completeness of business process. Approach described in this paper, following typical (systems integration) requirement scenarios, considers business process as only form of input for developing conversation model; however, requirements can be in multiple forms. Conversation model developed by this approach would be relevant to context of web service, but not necessarily comprehensive, as it may not be able to handle unforeseen exceptions that are not captured in business process. Thus, as a part of future work, above noted problems would be addressed. Also following design science methodology (Hevner et al. 2004), software artifact that implements the approach would be evaluated to demonstrate its utility.

REFERENCES

1. Aakhus, M. "Felicity conditions and genre: Linking act and conversation in LAP style conversation analysis," International Working Conference on the Language-Action Perspective on Communication Modelling, 2004.
2. Agarwal, R., and Sambamurthy, V. "Principles and Models for Organizing the IT Function," *MIS Quarterly Executive* (1:1) 2002, pp. 1-16.
3. Ardissono, L., Goy, A., and Petrone, G. "Enabling conversations with Web Services," Autonomous Agents and MultiAgent System (AAMAS), Melbourne, Australia, 2003.
4. Barros, A., Dumas, M., and Hofstede, A.H.M.t. "Service Interaction Patterns," Business Process Management, Springer-Verlag, Nancy, France, 2005a, pp. 302-318.
5. Barros, A., Dumas, M., and Oaks, P. "A Critical Overview of the Web Services Choreography Description Language (WS-CDL)," BPTrends, 2005b. http://www.bptrends.com/publicationfiles/03-05%20WP%20WS-CDL%20Barros%20et%20al.pdf
6. Benatallah, B., Casati, F., Toumani, F., and Hamadi, R. "Conceptual Modeling of Web Service Conversations," International Conference on Advanced Information Systems Engineering (CAiSE), Lecture Notes in Computer Science, Springer, 2003, pp. 449-467.
7. Hanson, J.E., Nandi, P., and Kumaran, S. "Conversation support for Business Process Integration," *IEEE International Enterprise Distributed Object Computing Conference (EDOC)*) 2002, pp. 65-74.
8. Hevner, A.R., March, S.T., Park, J., and Ram, S. "Design Science in Information Systems Research," *MIS Quarterly* (28:1) 2004, pp. 75-105.
9. Hohpe, G. "Let's Have a Conversation," *IEEE Internet Computing* (11:3) 2007, pp. 78-81.

10. Hohpe, G., and Woolf, B. *Enterprise Integration Patterns* Addison-Wesley, Boston, USA, 2004.
11. Iyer, B., Freedman, J., Gaynor, M., and Wyner, G. "Web Services: Enabling Dynamic Business Networks," *Communications of the Association for Information Systems* (11) 2003, pp. 525-554.
12. Lee, J., Siau, K., and Hong, S. "Enterprise integration with ERP and EAI," *Communications of the ACM* (46:2) 2003, pp. 54-60.
13. Markus, M.L., and Tanis, C. "The Enterprise Systems Experience–From Adoption to Success," in: *Framing the Domains of IT Management Research: Glimpsing the Future through the Past,* R.W. Zmud (ed.), Pinnaflex Educational Resources, Cincinnati, OH, 2000, pp. 173-207.
14. Ouyang, C., Dumas, M., Hofstede, A.H.M.t., and Aalst, W.M.P.v.d. "Pattern-based Translation of BPMN Process Models to BPEL Web Services," *International Journal of Web Services Research (JWSR)* (5:1) 2007, pp. 42-62.
15. Papazoglou, M.P. "The World of e-Business: Web-Services, Workflows, and Business Transactions," International Workshop on Web Services, E-Business, and the Semantic Web, Springer-Verlag, Toronto, Canada, 2002, pp. 153-173.
16. Peltz, C. "Web Services Orchestration- a Review of Emerging Technologies, Tools, and Standards," Hewlett Packard Labs Technical Paper.
17. Umapathy, K., Purao, S., and Barton, R.R. "Designing enterprise integration solutions: effectively," *European Journal of Information Systems* (17:5) 2008, pp. 518-527.
18. van der Aalst, W.M.P., Hofstede, A.H.M.t., Kiepuszewski, B., and Barros, A.P. "Workflow Patterns," *Distributed and Parallel Databases* (14:1), July 2003, pp. 5-51.
19. WS-BPEL "Web Services Business Process Execution Language (WS-BPEL)," OASIS, 2007. http://www.oasis-open.org/committees/download.php/23964/wsbpel-v2.0-primer.htm
20. WS-CDL "Web Services Choreography Description Language (WS-CDL)," W3C, 2005. http://www.w3.org/TR/ws-cdl-10/