A CP-nets-based Design and Verification Framework for Web Services Composition

Xiaochuan Yi and Krys J. Kochut
Department of Computer Science, The University of Georgia,
Athens, GA 30602-7401 {xyi,kochut}@cs.uga.edu

Abstract

Web services aim to support efficient integration of applications over Web. Most Web services are stateful, such as services for business processes, and they converse with each other via properly ordered interactions, instead of individual unrelated invocations. In order to address efficient integration of conversational Web services, we create a unified specification model for both conversation protocol and composition; we propose methods to integrate a partner service with complex conversation protocol into a composition of Web services; assure the correctness of composition by formal verification. The mapping between our model and BPEL4WS is also discussed.

1. Introduction

Many Web services in practice are stateful, and interactions with such a service are governed by its conversation protocols. WSDL [1] as the minimal conceptual service description model does not offer techniques to specify conversation protocols. WSCL [2] allows specification of conversation protocols based on the Finite State Machine (FSM) model, which lacks the ability to express concurrency, timeouts, or events. Newer languages such as WSCI [4] and BPEL4WS [3] offer their own means to describe the conversation protocols based on workflow models. [18] indicates that sharing workflow definitions across services will enable inter-operability, but it leads to tighter coupling amongst the services. Our contention is that both WSCI and BPEL4WS lack well defined formal semantics, thus they do not offer means for formal analysis.

With the emergence of Web service process technologies such as BPEL4WS, conversation specification languages such as WSCL should be an integral part of the overall process specification. Newer standards may provide their own means of describing conversations [5]. In this paper, we propose a unified specification model for both the conversation protocol and composition based on CP-nets [9].

The verification of the Web service composition is a very important issue. Bultan [11], Fu [12], and Foster [13] use FSM based model checking to verify composition. However, all of the above verification approaches require user to capture specific composition properties with temporal logic formula, no general criteria of correctness of composition is discussed. We believe that from the point of view of efficient integration of Web services, there are two key issues: (1) a precise and intuitive way to integration conversation partner into a composition; (2) a set of general criteria of control flow correctness. On the other hand, the ability to simulate the composition is a helpful addition to formal verification. [11] considers the realizabilility of composition given conversation specifications of peers, and they give some theoretical results to show that the global behavior of composition can be quite unexpected.

This paper is organized as follows: in Section 2 we introduce a motivating scenario, in Section 3 discuss the CP-nets-based model for conversation protocols. Section 4 presents the unified model as well as the design and verification framework for Web services compositions. Finally, Section 5 contains the concluding remarks and discussion of future work.

2. A motivating scenario

Consider a task of creating a composite Travel Agent Web service process, utilizing an Airline partner service and a Credit Card partner service. A similar problem is also discussed in [4]. For simplicity, we will assume that the Credit Card partner service offers only a single operation, and therefore has a *simple* conversation protocol. However, the Airline partner service provides five related operations:

CheckSeatsAvailability must be the first operation to be invoked, followed by *ReserveSeats* but only if a

Proceedings of the IEEE International Conference on Web Services (ICWS'04)
0-7695-2167-3/04 $ 20.00 IEEE

customer has already invoked *CheckSeatsAvailability* and that the requested seats are indeed available; the reservation is held only for a certain amount of time;

BookSeats or *CancelReservation* may be invoked, but only if the seats have been reserved (by a successful invocation of *ReserveSeats*) and the reservation has not yet expired;

If neither *BookSeats* nor *CancelReservation* has been invoked by the customer within a specified amount of time, the Airline service will itself invoke *notifyExpiration* to inform the customer that the reservation has expired.

A new Travel Agent Web service should offer a value-added service to travelers by incorporating interactions with the Airline and the Credit Card services, according to the following scenario:

A traveler planning on taking a trip submits a *TripOrder* (including all pertinent information) to her travel agent, hoping to get an *Itinerary* proposal (*GetItinerary*).

After the Travel Agent finds the best itinerary, the Travel Agent invokes the Airline service to verify the availability of seats (*CheckSeatsAvailability*). In case the seats are not available, the Travel Agent notifies the traveler and waits for the traveler to submit a modified *TripOrder*.

If seats are available, the proposed *Itinerary* is sent to the traveler for confirmation, who then decides to *Reserve* the seats for the *Itinerary*.

Next, the Travel Agent interacts with the Airline service to electronically finalize the reservation (*ReserveSeats*). The Airline holds such a reservation for certain amount of time, and if a *BookReq* (Book Request) is not received within the specified time, the seats are released and the Travel Agent is notified. The Travel Agent sends a *ReserveRes* (Reserve Result) message to the traveler as an acknowledgement.

At this point, the traveler can either *Book* or *Cancel* the reservation. If she decides to book the trip, she sends a *BookReq* (Book Request) to the Travel Agent containing her credit card information. The Travel Agent then invokes the Credit Card service. If the Credit Card service approves the charge, the Travel Agent invokes the *BookSeats* operation of the Airline service to finally book the seats. As a result, the Airline service books the seats for the proposed *Itinerary*, and issues an e-ticket to the traveler. The Airline service also sends a *BookRes* (Book result which contains statement along with a detailed description of the *Itinerary)* to the traveler. Otherwise, if the credit card charge is rejected, the Travel Agent notifies the traveler and waits for alternate payment information.

The creation of the composite Travel Agent Web service described above requires the incorporation of the Airline partner service, which has a complex conversation protocol.

3. The unified model

A Petri net [6, 7, 8] is a directed bipartite graph with two types of nodes (places and transitions) and extension of tokens. Petri nets have been used as a graphical and formal modeling tool suitable for systems involving communication, concurrency, synchronization, and resource sharing. Petri nets are more expressive than FSMs [8]. The Petri nets model provides the necessary primitives for specification of process interactions. CP-nets extend Petri nets with the primitives for the definition of the data types (color) and the manipulations of data values, thus a CP-net is more concise than a Petri net modeling the same system. CP-nets can be used to overcome the weaknesses [5] of FSMs specifying conversations [15]. They allow specification of concurrency and inter-process synchronization and thus are more suitable for specification of conversation protocols. Petri nets and CP-nets have already been successfully used to model communication protocols [10, 16, 17]. Moreover, Petri nets have been successfully used for specification of workflow processes [14].

3.1. CP-nets-based model for Web service conversation protocols

We introduce a *CP-nets-based model for the specification of conversation protocols*, in which:

(1) A conversation protocol of a Web service is represented as a CP-net.

(2) A WSDL operation is represented by an op-transition (CP-net transition tagged by op-). A one-way operation has one *in-place*, which receives and buffers inbound messages of the operation; A notification operation has one *out-place*, which buffers and transmits outbound messages of the operation; A request-response or solicit-response operation has a pair of in-place and out-place, and the in-place of a request-response operation is fed a token first, while the in-place of a solicit response operation wait for a token after the token in the out-place has been fired.

(3) Messages exchanged by the service and its customers are modeled by tokens. The protocol-relevant feature of a message is captured by usually small-sized color set of the corresponding token.

(4) The synchronization rules of the conversation protocol are captured by connecting the op-transitions

with internal places, arcs, and transitions used only for control flow purposes.

For example, the conversation protocol of the Airline service is shown in Figure 1. Tokens are timestamped; transitions *timeout* and *reset* are used only for control flow purposes; each of the op-transitions such as checkSeatsAvailability etc. represents a WSDL operations. The WSDL interface of the Airline service is listed in Appendix A of [15].

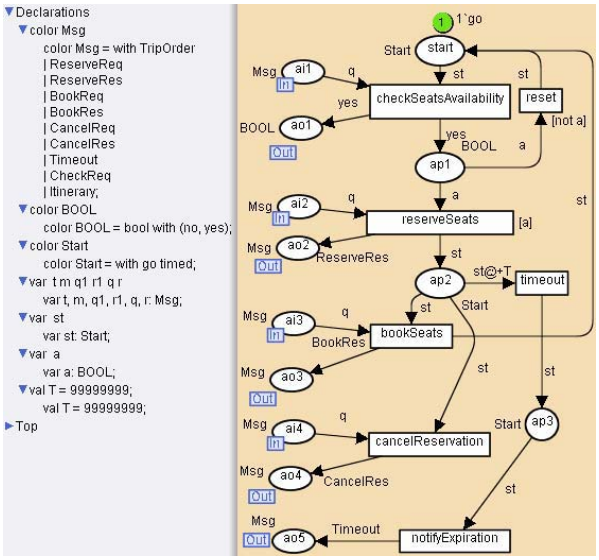

Figure 1. Conversation protocol of Airline service

3.2. CP-nets-based composition model

Our *CP-nets-based process composition model* is based on the control flow patterns of BPEL4WS and is defined as follows:

(1) The process of the composite service is represented by a CP-net *NetS*; a partner is represented with the CP-net model for its conversation protocol *NetP*; NetS interacts with *NetP* through arcs connecting the in- and out-places of *NetP*. Each arc must be labeled with a token variable that matches the colored set declared for the in-place/out-place.

(2) Messages (events) and process variables are represented by tokens. Abstract color sets are declared for the messages and variables (concrete content of the messages is not known at design time). Therefore, each color set is kept small to speed up the analysis.

(3) A Web service activity is usually mapped to a CP-nets transition. A <receive> activity is represented by a transition which has an in-place. A <reply> activity is represented by a transition which has an out-place. An <invoke> activity is represent by a pair of transitions, one of them may fire a request token to *NetP*, and the other may wait for a token from *NetP*. A structured activity is represented by a substitution transition. The control flow between activities is captured by connecting the activity-related transitions with arcs, places, and transitions purely used for control flow purpose. More refined control flow can be expressed with arc inscriptions and transition guard expressions.

A BPEL4WS process can be translated to an equivalent CP-nets model and CPN tools [10] can be used to analyze the process.

4. The CP-nets-based design and verification framework

4.1. Creation of new compositions

We represent the initial service process design in the CP-nets-based process composition model, (1) use the analysis techniques provided by CPN tools to detect process composition errors, (2) correct any errors, and repeat steps (1) and (2) until the specification of the designed process passes the verification.

The CP-nets-based process composition model also visualizes the incorporation of complex conversation protocols and helps the designers to avoid creating compositions inconsistent with protocol specification. The skeleton BPEL4WS code can be generated from the CP-nets-based process composition model after it has been thoroughly verified.

In our framework, the verification is carried out at the design time in order to detect and correct errors as early as possible. One of the verification problems we have found is *protocol conformance*, i.e. when a partner service with a complex conversation protocol is incorporated in a larger composite Web service, the protocol must be enforced properly. Protocol conformance checking is equivalent to detecting dead marking of the composition model. Many service process properties specified in design requirements correspond to the reachability of certain marking of the composition model. CP-nets analysis techniques can be used to analyze many important properties of the composition, such as boundedness, liveness, and reachability, which can be used as general criteria of correctness of composition.

According to the description of the example in Section 2, the Travel Agent service process should provide a portType for customers which contains five operations – *GetItinerary*, *Reserve*, *Book*, *Cancel*, *NotifyTimout* (the WSDL interface is listed in [15]). In our framework, the CP-nets-based process composition

model is created in a structured way in recognition of process (workflow) patterns. We have discussed the representation of these process patterns both in CP-nets and BPEL4WS constructs [15].

Since the content of most messages in the process is not known until execution time, color set *Msg* is declared as an enumeration of message types such as *TripOrder* and so on. *Start* is declared a unit color set only for flow control. Place *TA_start* indicates both the beginning and end of the process. A token removed from *TA_start* indicates the beginning of the process; a token deposited into *TA_start* indicates the termination of the process. The CP-nets models of the conversation protocols of the partner services are connected with the CP-nets model of the service process. The visualization of the incorporation of complex conversation protocols helps designers to match message types and follow the synchronization of rules of the partner service. The hierarchy constructs of CP-nets, such as the transition substitution [9] can be used to modularize the design by placing the partner service on a sub-page.

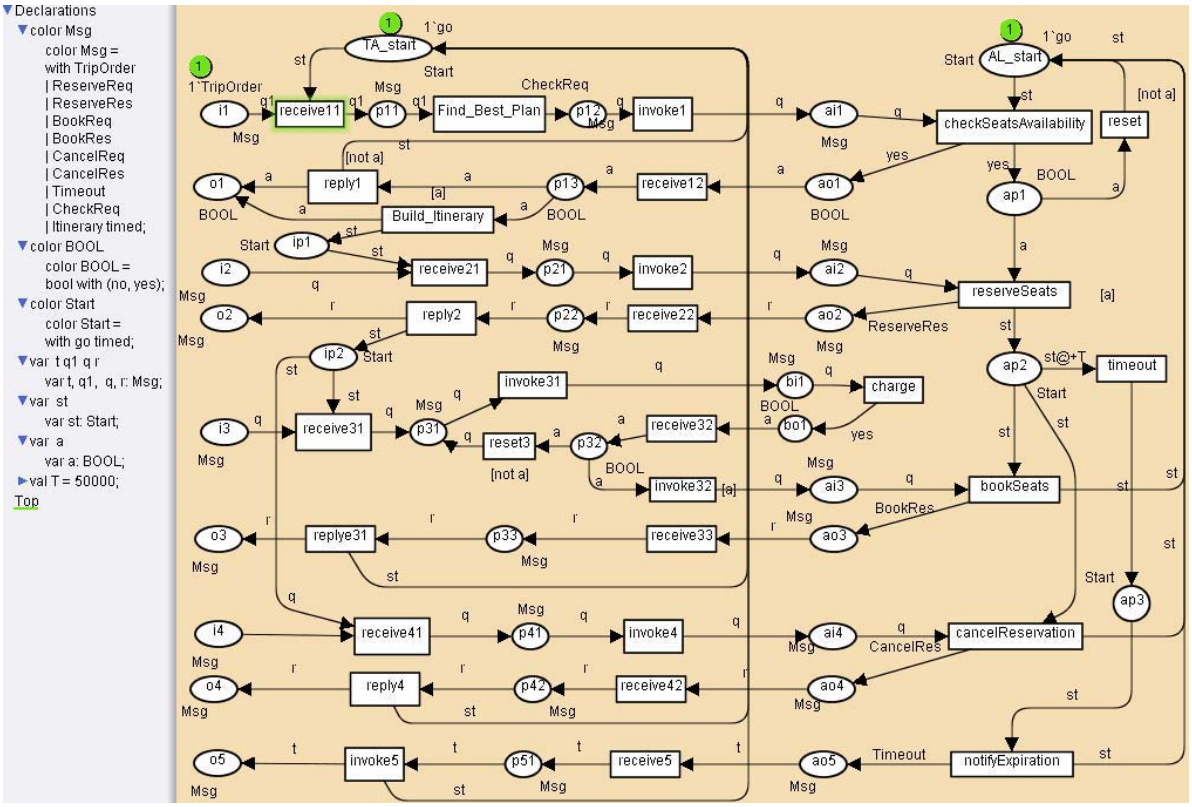

Figure 2. CP-nets-based process composition model for the initial design of the Travel Agent service

During the initial design, the designers can create certain initial markings and run the simulations with the use of CPN tools in order to check whether the service process will be executing as expected, and whether the service process is consistent with the conversation protocol of each partner service. Furthermore, state space analysis can be carried out for the CP-nets-based process composition model to formally analyze the service process. The final version of the service process model is shown in Figure 2. The BPEL4WS skeleton code can be generated from the CP-nets-based process composition model (stored in CPN XML form). A simplified version of the BPEL4WS skeleton code of the Travel Agent service process is shown in Appendix B of [15]. Further implementation details needed by the execution can be added to the thoroughly tested BPEL4WS skeleton to complete the final composition specification.

4.2. Derivation of conversation protocol

The Travel Agent service process exhibits a fairly complex conversation protocol. In order for customers to use the new service correctly, the conversation protocol of the Travel Agent service must be specified. The conversation protocol specifies the ordering rules among WSDL operations. It can be derived from the

corresponding CP-nets-based process composition model. The internal process details (e.g. private data manipulation, and interaction with other partners) should be hidden in the conversation protocol. The conversation protocol should only reveal the externally visible behavior of the composite service process, i.e. its operations and their sequencing rules. The protocol derivation can be conducted automatically, based on a series of occurrence graph construction of CP-nets. The algorithm is detailed in [15].

The algorithm is applied to the CP-net in Figure 2, and it produces the CP-net model for the conversation protocol shown in Figure 3.

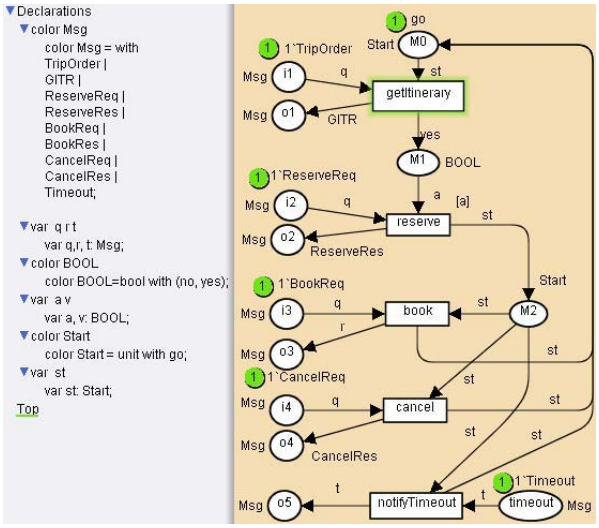

Figure 3. Travel Agent Conversation protocol

5. Conclusions and future directions

In this paper, we have specified a unified model for the specification of the Web services conversation protocols and process compositions based on CP-nets.

We have also proposed a design and verification framework, which can be used to visualize and verify existing BPEL4WS processes. The framework can be used to support creating new BPEL4WS processes, as it enables the incorporation of a partner service with complex conversation protocol into the overall composition. The framework enables the use of many verification techniques at the design time to detect errors as early as possible. The skeleton of the BPEL4WS process specification is generated from the thoroughly verified process model. Finally, we have presented a method of automatically deriving the conversation protocol of a new composite service.

We plan to develop a graphical designer to support the creation of Web service compositions, with BPEL4WS code view as well as the CP-net view. The conversation protocol derivation program will be implemented, as well. We also plan to propose an extension of WSDL to include the conversation protocol specification.

6. References

[1] "Web Services Description Language (WSDL) 1.1", *W3C Note*, 15 March 2001. http://www.w3.org/TR/wsdl

[2] "Web Services Conversation Language 1.0 ",*W3C Note*, 14 March 2002. http://www.w3.org/TR/wscl10/

[3] "Specification: Business Process Execution Language for Web Services Version 1.1", 05 May 2003. http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/

[4] "Web Service Choreography Interface (WSCI) 1.0", *W3C Note*, 8 August 2002. http://www.w3.org/TR/wsci/

[5] S. Chatterjee, J. Webber, "Developing Enterprise Web Services An Architect's Guide", Prentice Hall, 2004

[6] C.A. Petri, "Communication with Automata", *Griffiss Air Force Base Tech. Report RADC-TR-65-377*, vol. 1, Suppl. 1, 1966

[7]T. Murata, "Petri nets: Properties, analysis and applications", *Proceedings of IEEE*, vol 77, no 4, April 1989

[8] J. L. Peterson, "Petri Net Theory and the Modeling of Systems", Prentice-Hall International, 1981.

[9] K. Jensen, "Colored Petri Nets Basic Concepts, Analysis Methods and Practical Use", Volume 1, 2 and 3, second edition, 1996

[10] CPN tools. http://www.daimi.au.dk/CPNtools/

[11] T. Bultan, X. Fu, R. Hull, and J.Su, "Conversation Specification: A New Approach to Design and Analysis of E-Service Composition", *Proc. of Twelfth International World Wide Web Conference (WWW2003)*, 2003.

[12] Xiang Fu, Tevfik Bultan, and Jianwen Su, "Analysis of Interacting BPEL Web Services", *Proc. of the Thirteenth International World Wide Web Conference (WWW 2004)*.

[13] H. Foster; S. Uchitel, J. Kramer, and J. Magee, "Model-Based Verification of Web Service Compositions", *Automated Software Engineering (ASE) Conference 2003*, Montreal, Canada, October 2003.

[14] W.M.P. van der Aalst, "The Application of Petri Nets to Workflow Management", *Journal of Circuits, Systems and Computers,* 8(1):21-66, 1998.

[15] X. Yi, and Krys J. Kochut, "A Framework for Web Services Composition Design and Verification", *UGA Computer Science Technical Report*, UGA-CS-TR-04-001. http://www.cs.uga.edu/~xyi/tr04001.pdf

[16] T. Suzuki, S. M. Shatz, and T. A. Murata, "Protocol Modeling and Verification Approach Based on a Specification Language and Petri Nets", *TSE* 16(5): 523-536, 1990.

[17] J. Billington, M. Diaz, G. Rozenberg (eds.), "Application of Petri nets to communication networks: advances in Petri nets", *Lecture Notes in Computer Science*, Springer, 1999.

[18] S. Frølund, K. Govindarajan, cl: A Language for Formally Defining Web Services Interactions, HP Laboratories Palo Alto, *HPL-2003-208*, October 1st, 2003