SPECIFICATION AND VERIFICATION OF VIEWS OVER COMPOSITE WEB SERVICES USING HIGH LEVEL PETRI-NETS

Khouloud Boukadi[α], Chirine Ghedira[β], Zakaria Maamar[γ] and Djamal Benslimane[β]
[α]*Division for Industrial Engineering and Computer Sciences, ENSM, Saint-Etienne, France*
[β] *LIRIS Laboratory, Claude Bernard Lyon 1 University, Lyon, France*
[γ]*College of Information Technology, Zayed University, Dubai, U.A.E*

Keywords: Web service, Composition, Context-aware, high level Petri-Net, Views.

Abstract: This paper presents a high level Petri-Net approach for specifying and verifying views over composite Web service. High level Petri-Nets have the capacity of formally modelling and verifying complex systems. A view is mainly used for tracking purposes as it permits representing a contextual snapshot of a composite Web service specification. The use of the proposed high level Petri-Net approach is illustrated with a running example that shows how Web services composition satisfies users' needs. A proof-of-concept of this approach is also presented in the paper.

1 INTRODUCTION

Web services (WS) have given Web applications a new shape, from content display to service supplier. The capacity of defining composite Web services is an advantage that currently backs the widespread use of Web services. Businesses and academia have shown a significant interest in WS composition (Daniel 2005). Despite the large body of research on Web services, much work still needs to be done to tie informal methods, e.g., Petri-Nets, with specification languages for composite Web services. Few initiatives have looked into the use of Petri-Nets in WS including (Yang 2005). Indeed, there is no guarantee that the specification of a composite Web service is error-free. Conflicts like concurrent accept or reject and deadlocks may occur during the specification execution. Fixing errors at run-time is time-consuming and requires another round of low-level programming, which could be expensive, and error-prone. An attractive solution would consist of allowing developers to detect and fix issues prior to WS deployment and to formally verify the business processes underlying composite WS against some desired properties.

Composition does not only make WS bind to each other, but emphasizes the cornerstone of handling users' preferences and constraints as part of the process of meeting personalization requirements. Personalization is tightly related to the features of the environment in which WS will operate after triggering. These features can be related to users (e.g., state, location), computing resources (e.g., fixed device, mobile device), time periods (e.g., in the afternoon, in the morning), physical places (e.g., mall, cafeteria), etc. Sensing, gathering, and refining the features and changes in an environment contribute towards the definition of what is known as context. Context *is the information that characterizes the interactions between humans, applications, and the surrounding environment* (Medjahed 2003). Embedding WS with context-awareness mechanisms has several advantages. To be aware of which part of the specification of the composite Web service has to be adjusted because of changes in the user environment, an assessment of what-was-previously-expected and what-is-effectively-happening is deemed appropriate. This specific part of the composite Web service specification is referred to as view. A view is *a dynamic snapshot over the specification of a composite Web service according to a certain context* (Benslimane et al. 2005).

In this paper, we aim at discussing the value-added of Petri-Nets to the specification of firstly, the composite WS and secondly, the views that run over those ones. We emphasize the use of high level Petri-Nets, particularly Colored-Petri-Nets (CPN) and Hierarchical-Colored-Petri-Nets (HCPN) (Jensen 1997). CPN and HCPN have the capacity to specify and analyze concurrent systems (Petri 1962). Our contributions in this paper are as follows: a definition of a composite Web service using high level Petri-Nets, an approach for checking the correctness of a composite Web service, a specification of a view based on high level Petri-Nets, and finally, automatic mechanisms for extracting and showing up views over composite WS.

The rest of this paper is organized as follows. Section 2 presents a scenario, lists some Petri-Nets' advantages, and suggests a list of related projects. Section 3 discusses the use of Petri-Nets in modelling WS and composite WS. Section 4 describes the concept of view as a means for tracking the execution of a composite Web service specification. Section 5 presents the prototype that was developed as a proof-of-concept of our use of Petri-Nets in WS. Concluding remarks are drawn in Section 6.

2 BACKGROUND

2.1 Motivating Scenario

Our motivation scenario concerns Anatole, a 60-years old patient who has a Portable ECG Monitor (PEM), which is used to detect and manage any cardiac event. An electrocardiogram (ECG) is a test that records the heart's electrical activity. When Anatole feels a chest pain, he turns on the PEM so his ECG is recorded. The PEM starts with a serial analysis of this record and compares it with the referenced ECG. The PEM can suspect any cardiac problems and send an alert to a call center, if needed. The alert triggers a Web service whose role is to find a first-aid medical-center close from Anatole's current location. Processing both the recorded and referenced ECG, the selected medical center identifies two types of alarm: severe or minor. In case of a minor alarm, **LookforDoctor** and **TreatmentTransmission** WS are triggered. When a doctor is assigned to Anatole, he gets access to his medical records and checks the referenced and recorded ECG. Afterwards, he diagnoses the case and prescribes an adequate treatment for Anatole. **TreatmentTransmission** WS takes care of notifying the treatment, as an SMS message, to Anatole's mobile phone. The language of the message is set according to Anatole's preferred-language. In case of a severe alarm, **LookforEmergency** WS is concurrently triggered with other separate WS that upload Anatole's medical records and identify Anatole's location, respectively. Finally, **ContactMobileCare** WS is activated in case an ambulance is needed. This motivating scenario will illustrate our approach, and makes clear the use of CPN for contextual WS composition.

2.2 Rationale of Views

In the three-level ANSI-SPARC architecture, a view corresponds to the organization of the database as it appears to a particular user. In the relational model, a view is defined as a virtual relation that is dynamically derived from one or more other base relations. The concept of view is also used in the workflow community. Workflow view was suggested as a support mechanism for the interoperability of workflows across multiple businesses (Chiu et al., 2004).

The use of the view concept is backed by our previous work and is motivated by several reasons.
First, the view mechanism grants a powerful and flexible approach by hiding the whole specification of a composite service from users and the process responsible for adjusting this specification. Only the significant parts of a specification are presented. Second, multiple views over a specification can be obtained at different levels of granularity ranging from the dependency between WS and the execution preferences coupled with WS to the corrective measures that WS use. In (Benslimane et al. 2005), we proposed a formal specification of the view concept based on state-chart diagrams, and a set of colored graph procedures to manage the transition according to the context and the dynamic nature of the view. This proposal motivated the study of the correlation and the value-added of the colored Petri-Nets.

2.3 Rationale of Colored Petri-Nets

Jensen formulates CPN as a formally founded graphically-oriented modeling language (Jensen 1997). CPN have got their name because they use different colors to be associated with tokens, which carry data values. This is in contrast to low level Petri-Nets' tokens, which by default are black. On the one hand, Petri-Nets provide the necessary mechanisms for specifying synchronization of concurrent processes. On the other hand, any programming language provides the primitives that are needed for defining and manipulating data types.

Compared to CPN, HCPN includes additional features such as substitution transitions.

Mapping CPN and HCPN concepts into WS and composite WS is to a certain extent straightforward. First, a WS's behavior is basically a partially ordered set of operations. Therefore, it is possible to represent it with a Petri-Net. WS's operations are modeled by transitions and the states of the WS are modeled by places (Benatallah 2003). In addition, the use of colored tokens permits modeling contexts of WS and users by specifying places used to model these contexts. Moreover, the hierarchy concept of the HCPN shows the components of a composite WS at a higher level with no mention to their internal details. This is really useful for running contextual views over composite WS.

2.4 Related Work

Benatallah et al. propose a Petri-Net-based algebra for composing WS (Benatallah 2003). In this study, context is just ignored, which does not permit capturing the changes in a WS composition process. Yang et al. concentrate in verifying and analyzing composition specification of WS once these specifications get translated into a CPN (Yang 2005). Yang suggests also to model BPEL processes as CPN. However, the transformation process is ambiguous and no formal definition of how to translate a BPEL specification into a CPN is given. Xiaochuan et al. propose a model of a simplified Travel Reservation system using WS (Xiaochuan 2004). In (Bing 2006), the author addresses the shortfalls of Xiaochuan et al.'s work like incomplete conversation between WS, removal of some major interactions within WS, and modeling of unnecessary components that make the graphical representation complex.

All these proposals mainly focus on WS composition modeling with no-emphasis on contexts of WS and users. The transformation process for example from a BPEL specification towards a CPN is still ambiguous and a formal definition would be highly appreciated. In order to react in a proper way to the detected changes in user and WS environment, context needs to be handled during the development of specifications of WS.

3 MODELING WEB SERVICES COMPOSITION USING HIGH LEVEL PETRI-NETS

In this section, we define a WS and a composite WS using CPN and HCPN, respectively. To this purpose, we comply with Jensen's work (Jensen 1997). A composite WS is defined by an HCPN to be called as Composition Net (CN). Moreover, each component WS in the CN has a page modeled by a CPN to be called as Service Colored Net (SCN).

3.1 Service Colored-net Definition

As aforementioned, we define a WS as a **SCN=<Σ,P,T,L,A,N,C,E,G>** where:

Σ is a finite set of types also called color sets.
P is a finite set of places that model the state of a system. A WS's states consist of distributing a data value, i.e., token, on the SCN's places. Two types of places exist:

- **Message Places (MP)** contain messages exchanged between component WS.
- **Context Places (CP)** containing the execution context of WS.

T is a finite set of transitions. Each operation in a WS is captured by a CPN transition. We can distinguish two types of transitions: $\mathbf{T}_{\mathbf{gu}}$ is a finite set of transitions with guard condition and $\mathbf{T}_{\overline{gu}}$ is a finite set of transitions without guard condition.
A is a set of directed arcs. An arc connects a place to a transition and *vice-versa*. In fact, an arc represents a causality relation between places and transitions.
L is a labeling function for each operation in a WS.
N is function that links each arc going from a place to a transition and *vice-versa*.
C is a color function that assigns a unique color to each place p. The color of a place is denoted as C(p). Therefore, each token in a place p must have a color, i.e., data value, from C(p).
E is a function that describes arcs using a set of variables. These variables determine the token's variables (i.e., a token has a set of variables) that are either consumed or produced during operation.
G is a guard function that checks the logical conditions in a transition.

Hence, the SCN is defined, we can focus on how we get over transitions inside a SCN of a composite WS. Indeed, the evolution of a SCN consists in crossing its transitions, this task is based on two types of rules:

1. Firing rule for a transition with guard condition: In order to get over a transition with guard condition, we must consider the types of places, whether message or context, and thus the conditions of these places. In case of a message place, four conditions should be verified before a transition can be passed. The first condition deals with the color of the place. The second condition verifies that the variable set of the arc has the same type as the place connected with this arc. In addition, the values of the variables on an arc must match the expected data types such as integer. The last condition checks if the guard condition returns true assuming that every type of variables of a guard belong to the color sets of the service colored net. We use Is-enabled (T_{gu}) as the function that checks the four conditions for each message place that is connected to transition T_{gu}. In case of a context place, only the three first conditions must be verified.

2. Firing rule for a transition without guard condition: this transition is independent of the type of its connected places. Is-enabled ($\mathbf{T}_{\overline{gu}}$) function verifies only the first three conditions. Once each SCN of the WS participating in a composite WS defined, we can elaborate its Composition Net.

3.2 Definition of the Composition Net

A composite WS is a HCPN that is defined as follows: **CN=<S, ST, SA, PP, PT, PA, FP>** where:
S is a set of pages that represent the atomic WS. Each page $s \in S$ is a **:**

SCN=<∑s,Ps,Ts,As,Ns,Cs,Es,Gs>

ST is a set of substitution transitions. A substitution transition identifies a WS without any internal details on how it is performed.
SA is a function that assigns a WS to a composite WS. Indeed, each ST corresponds to SCN **(SA: ST $\rightarrow$ SCN)**. We assume that firing a substitution transition depends on the firing of all the transitions that are present in the SCN**.**
PP $\subseteq$ P is the set of port places. Each SCN contains places that are tagged with either in, out, or i/o. These places are named port places and permit the communication of a SCN with its peers. As mentioned before, each substitution transition is related to a SCN. This is achieved by providing a port assignment, which describes how the port places of the SCN are related to the socket places of the substitution transition.
PT defines the type of the port, PT: PP $\rightarrow$ {in, out, i/o}.
PA is a port assignment function that describes how the port places of the SCN related to the socket places of the substitution transition.
FP is the first page of the CN, i.e., it represents the composite WS. For each substitution transition in the first page, a SCN is obtained.

3.3 Verifying a Composite Web Service

The use of high level Petri-Nets permits increasing the reliability level of composite WS. The associated CN could be subject to analysis using different techniques and computer tools for CPN. The most important one known as a state space method consists of designing a graph that has a node for each reachable marking, as well as an arc for each occurring binding element. We suggest the definitions of four properties that can be checked using CPN: **Reachability** that determines whether it is possible for a composition to achieve the desired results; **Boundness** that determines the minimal and the maximal number of tokens in the different places; **Dead transition** that determines the number of transitions which will never be enabled and **Dead marking** which is a marking with no enabled transitions.

3.4 Illustration with Anatole Scenario

The composition net of Anatole scenario is shown in Fig.1. It consists of seven WS designed as substitution transitions. The substitutions transitions are: **{ LookforCenter**, **LookforDoctor**, **LookforEmergency**, **TreatmentTransmission**, **UpdatePatientRecord**, **Localization**, and **ContactMobileCare }**.
The boxes that are next to each substitution transition specify the SCN that contains the detailed description of the activity represented by the corresponding substitution transition. For example, the page modelling **LookforCenter** WS is modeled by the substitution transition named LookforCenter. The substitution transition for LookforCenter WS is broken up into three context places (CP1, CP2, CP3), as an input and a single message place (PTLFC). These places are the input socket places for this substitution transition. For illustration purposes, the following assumptions are made regarding the context and message places: **CP1** contains the required memory, e.g., 1'128 in Megabits, for the execution. **CP2** contains the time-slot availabilities of the WS for execution. **CP3** contains French language using 1'french.

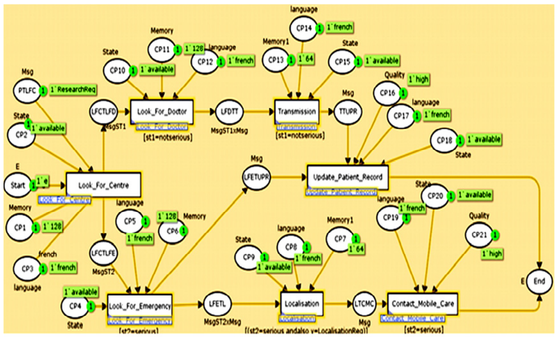

Figure 1: The CN for Anatole scenario.

PTLFC models the message between the PEM and **LookforCenter** WS. The two output socket places for this substitution transition are: LFCTLFD models the message between **LookforCenter** and **LookforDoctor** WS; LFCTLFE models the message between **LookforCenter** and **LookforEmergency** WS.

Now we will focus on a component WS of this scenario: the LookForCenter WS. Let us consider the sub-page of Fig. 2, which is about the detailed description of the activity that **LookforCenter** WS carries out. The sub-page shows two operations captured by two transitions. We also consider **EvaluationState** transition in order to observe how the firing rules get initiated. **EvaluationState** transition is fired iff the following conditions are satisfied: (1) The color set of each place connected to EvaluationState transition is included in the color set of this transition; (2) The arc's variables type matches the color of P1; (3) Having a Boolean color, the ok variable can only receive true or false. Besides, this variable must have a type already defined in the color set of SCN_{LFC} and (4) EvaluationState transition is enabled if the ok variable in the guard condition evaluates to true. In our case, the last condition depends on the value that is randomly assigned to the ok variable since all others conditions are satisfied.

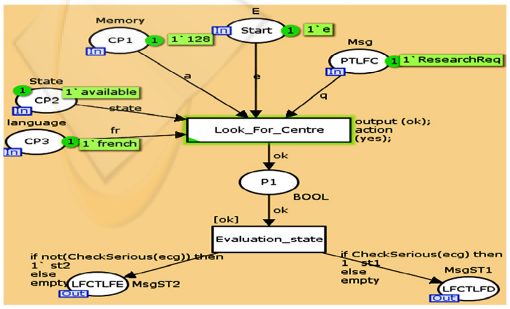

Figure 2: SCN for LookforCenter Web Service.

To apply HCPN modelling to a WS composition according to a specific context, we introduce in the next section, the formal specification of the view concept.

4 THE CONCEPT OF VIEW

4.1 Formal Definition

We recall that a view is a dynamic snapshot over the specification of a composite WS according to a certain context. We suggest below that a view is extracted out of the specification of a composite WS using high level Petri-Nets. We provide the definitions below.

Initial composition net definition. An **ICN** is defined as the following triplet: **ICN=<S,$ST_{\overline{gu}}$,ST_{gu}>** where: **S** is the set of pages that are included in the CN where $\forall s \in S$, s is an SCN, and $ST_{\overline{gu}}$ and ST_{gu} are like previously defined.

Context template definition. CT is the formal model of the corresponding context during view extraction. The CT includes two types of context: user (U-context) and WS (W-context), **CT= {U-context $\cup$ W-context**} detailed in (Ghedira 2006).

Derived composition net definition. The extraction of a view according to a certain context over an initial or derived composition specification permits obtaining a **DCN. DCN** is defined with the following triplet: a **DCN=<S',S'$T_{\overline{gu}}$,S'T_{gu}>** where: **S'**: is the derived specification that does not accept any additional WS through their pages; **S'$T_{\overline{gu}}$ = {st'|$\exists$st$\in$$ST_{\overline{gu}}$ $\wedge$ Is-enabled (st)=*true*}** is the new set of substitution transitions without guard conditions; and **ST_{gu} = {st'|$\exists$st$\in$$ST_{gu}$$\wedge$ Is-enabled (st)=*true*}** is the new set of substitution transitions with guard conditions.

4.2 Application to Anatole Scenario

Let us assume that Anatole's contexts returns details on his physical state and localization.
SU-context={Identity="Anatole", Age="60", Gender="Male"};
DU-context={PsychologicalState="stressed", PhysicalState="serious", Localization="Fourvière Cathedral"}

An example of WS context is the context of **ContactMobileCare** WS: W-context={SW-context $\cup$ DW-context};
SW-context={Name="ContactMobileCare", Memory="128", language= "French"};
DW-context={availability="no"}
Fig. 3 shows the derived composition net that is extracted out of the composition net of Fig. 1 according to the defined context template.

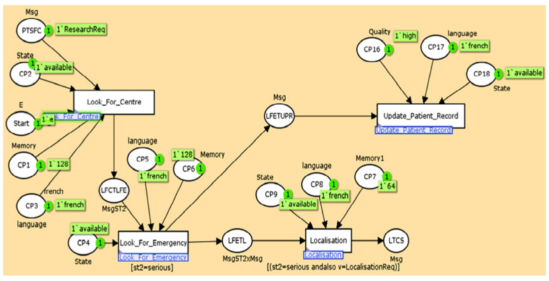

Figure 3: DCN for Anatole scenario.

Prototype: A prototype is fully operational. We used Java to implement the needed functionalities for context collection and generation as well as for view extraction. The architecture of the prototype comprises two modules that a Java program orchestrates. The first module is about the context generator and the second is the view extractor. The context generator provides, upon request, several contextual details related to users and WS. To this purpose, two XML files are delivered by the context generator. Both files are then submitted to the view extraction module. We used CPN Tools, which is a tool for editing, simulating and analyzing CPN. The extraction of a view consists of comparing the expected contextual elements that are associated with this specification to the current contextual details that are obtained out of the context generator. The result of the comparison is an XML file that corresponds to the view that can now be visualized as a Petri-Net using the CPN Tools and verified using the various properties we listed in Section 3.4.

5 CONCLUSION

In this paper, we presented a high level Petri-Net approach for the specification and verification of composite WS. Our literature review has shown that building reliable composite WS calls for formal verification. Our literature review has also shown that no much has been done to cater for context in composite services. Therefore, we proposed a high level Petri-Net approach that integrates context during specification, maps this specification onto a Petri-Net. Furthermore, we discussed in this paper how the execution of a composite WS is tracked using view. We illustrated and prototyped the dual use of Petri-Nets and views with a patient-related scenario. Although this scenario was simple, it revealed the challenges that need to be taken up when deploying WS in critical domains such as healthcare. Our next work aims at proposing extensions for BPEL with user and WS contexts included. In addition, we aim at developing a tool that converts an extended BPEL specification into a CPN for automatic verification purposes.

REFERENCES

Benatallah, B. and Rachid, H., 2003. A Petri net-based model for web service composition, in *Proceedings of the Fourteenth Australasian database conference on Database technologies*. Adelaide, Australia.

Benslimane et al, 2005. A View-based approach for tracking composite Web services. *In Proceedings of the European Conference on Web Services*, IEEE Computer Society. Växjö, Sweden.

Bing, H., 2006. Choreography Modeling and Analysis of a Travel Reservation Web Service, in *Proceedings of The Fifth International Joint Conference on Autonomous Agents & Multi-Agent Systems*. Hakodate, Japan.

Chiu, D. K. W., S. C. Cheung, S. Till, K. Karlapalem, Q. Li, and E. Kafeza, 2004. Workflow View Driven Cross-Organizational Interoperability in a Web Service Environment. *Information Technology and Management Journal*, Vol. 5, No. 3/4:221-250.

Daniel, F. and Pernici, B. , 2005. Insights into Web Service Orchestration and Choreography, *International Journal of E-Business Research, The Idea Group Inc.*, vol. 1, pp. 58 - 77.

Ghedira, C. and Mezni, H., 2006. Through Personalized Web Service Composition Specification: From BPEL to C-BPEL, *Electronic Notes in Theoretical Computer Science*, vol. 146, pp. 117-132.

Jensen, K., 1997. *Colored Petri Nets: Basic Concepts, Analysis Methods, and Practical Use*, 2nd ed. Berlin; New York: Springer.

Medjahed, B. et al., 2003. Composing Web services on the Semantic Web, *International Journal on Very Large Data Bases*, vol. 12(4), pp. 333-351.

Petri, C., 1962. *Kommunikation mit Automaten*. Schriften des IIM Nr. 2, Institut fur Instrumentelle Mathematik. Germany: University of Bonn.

Xiaochuan, Y. and Krys JK., 2004. Process Composition of Web Services with Complex Conversation Protocols: a colored Petri Nets Based Approach, in *Proceedings of The Design, Analysis and Simulation of Distributed Systems Conference*.

Yang, Y. et al., 2005. Transformation BPEL to CP-nets for verifying Web services composition, in *Proceedings of The International Conference on Next Generation Web Services Practices (NWeSP'05)*. Seoul, Korea.