$\mathcal{CP}4\mathcal{WS}$ - A METHOD FOR DESIGNING AND DEVELOPING SYSTEMS BASED ON WEB SERVICES

Zakaria Maamar[β], Djamal Benslimane[γ] and Chirine Ghedira[γ]

[β]*Zayed University, Dubai, United Arab Emirates*
[γ]*Claude Bernard Lyon 1 University, Lyon, France*

Keywords: Design, Development, Method, Web Service.

Abstract: This paper presents $\mathcal{CP}4\mathcal{WS}$ (standing for $\mathcal{C}$ontext and $\mathcal{P}$olicy for $\mathcal{W}$eb $\mathcal{S}$ervices), which is a context-based and policy-driven method for the design and development of Web services-based systems. Although Web services constitute an active area of research, very little has been achieved for the benefit of those who are responsible for modeling and developing such systems. To address this lack of support, we developed $\mathcal{CP}4\mathcal{WS}$ that consists of several steps ranging from user needs identification to Web services behavior specification. A running scenario that illustrates the use of $\mathcal{CP}4\mathcal{WS}$ is also discussed in the paper.

1 INTRODUCTION

Overview. For the W3C, a Web service *is a software application identified by a URI, whose interfaces and binding are capable of being defined, described, and discovered by XML artifacts and supports direct interactions with other software applications using XML-based messages via Internet-based applications*. Web services R&D have to a certain extent revolved around the following aspects: description, discovery, and composition. Several standards related to Web services definition, discovery, etc., have been developed, and several projects related to Web services composition, personalization, etc., have been kicked-off.

In this paper, we shed the light on another aspect that concerns **design and development methods**. The objective, here, is to assist people in designing and developing Web services-based systems. Nowadays, designers and developers are put on the front line of satisfying the promise of Web services' providers to deliver a new generation of B2B systems. Simply put, a method comprises a set of steps to carry out and comply with according to a certain chronology. Each step has usually a representation formalism that facilitates discussions and validation exercises among the design team's members and with end-users, respectively. Our proposed method, $\mathcal{CP}4\mathcal{WS}$ for $\mathcal{C}$ontext and $\mathcal{P}$olicy for $\mathcal{W}$eb $\mathcal{S}$ervices, is built upon our previous research on Web services (Maamar et al., 2006a; Maamar et al., 2005). $\mathcal{CP}4\mathcal{WS}$ uses two major concepts: policy and context. First, policies handle various aspects related to Web services including participation in composition, semantic mediation, and adjustment due to changes in the environment. Second, context provides the necessary information that enables for instance to trigger the appropriate policies and to regulate the interactions between Web services according to the current state of the environment. In $\mathcal{CP}4\mathcal{WS}$, an extra element namely resource is part of the design and development exercise. A resource is a computing means upon which a Web service operates. Because resources need to schedule the execution requests of Web services, these latter have to be constantly aware of the capabilities and limitations of the resources they require.

Definitions and motivation. Policies are defined as *information which can be used to modify the behavior of a system* (Lupu and Sloman, 1999). Our definition goes beyond behavior modification and considers policies as external, dynamically modifiable rules and parameters that are used as input to a system. This permits to the system to adjust to administrative decisions and changes in the execution environment (Maamar et al., 2006b). In the field of Web services, policies specify different aspects of the behavior of a Web service so, this one can align its capabili-

Maamar Z., Benslimane D. and Ghedira C. (2007).
C P 4W S - A METHOD FOR DESIGNING AND DEVELOPING SYSTEMS BASED ON WEB SERVICES.
In *Proceedings of the Ninth International Conference on Enterprise Information Systems - ISAS*, pages 453-458
DOI: 10.5220/0002349804530458

ties to users' requirements and resources' constraints.

Context "*... is not simply the state of a predefined environment with a fixed set of interaction resources. It is part of a process of interacting with an ever-changing environment composed of reconfigurable, migratory, distributed, and multiscale resources*" (Coutaz et al., 2005). In the field of Web services, context supports the development of adaptable Web services so, these latter become aware of the environment in which they operate. Several Web services' standards are now enriched with contextual details (Keidl and Kemper, 2004).

We argue the value-added of context and policies to $\mathcal{CP}4\mathcal{WS}$ for supporting Web services composition with the following points:

1. Context provides useful information concerning the environment wherein the composition of Web services occurs. This information tracks the composition process by enabling for instance to trigger the appropriate policies and to regulate the interactions between Web services according to the current state of the environment and the current constraints such as performance, security, etc.
2. Policies separate guidelines for conducting composition from guidelines for defining Web services. Guidelines for composition concern among other things how to integrate user preferences into Web services, how to guarantee the satisfaction of these preferences subject to resource availability, and how to track the execution progress of Web services? Guidelines for Web services concern among other things how to exchange comprehensible information between Web services, how to deal with a Web service non-reliability, and how to suspend a Web service execution due to risks of behavior alteration or information interception?

It is made clear at this stage of the paper that our motivation do not aim at suggesting any new composition approach of Web services. Our aim is to suggest a set of steps that designers follow while designing and developing Web services-based systems. It will also be shown later that the various available standards like WSDL and BPEL can easily be integrated into these steps.

Paper organization. Section 1 motivates the role of design and development methods in Web services and defines some basic concepts. Section 2 presents the different steps that constitute $\mathcal{CP}4\mathcal{WS}$. For illustration purposes, a motivating scenario is suggested in this section. Section 3 reviews the prototype that implements the motivating scenario. Related work and concluding remarks are presented and drawn in Section 4 and Section 5, respectively.

2 DESIGN STEPS IN $\mathcal{CP}4\mathcal{WS}$

2.1 Motivating Scenario

Our motivating scenario concerns Amin who is visiting Melissa in Oslo. Amin and Melissa agree on a certain day to meet in a coffee shop, not far from Melissa's office since she finishes work late on this day. Amin has two options to reach the meeting place: by taxi or by bus. For illustration purposes we identify some potential Web services that will take over the implementation of this scenario.

At his hotel, Amin browses some Web sites about transport in Oslo. A site has `Itinerary WS` that proposes routes between for example Amin's hotel and the coffee shop. The proposed routes are subject to weather forecasts and traffic jams. Cold weather results in recommending taxis, otherwise public transport like tramways and buses are recommended. Parallel to consulting `Weather WS` about weather forecasts, `Itinerary WS` requests details about the origin and destination places using `Location WS`. In case `Weather WS` returns bad weather, a taxi booking is made using `Taxi WS` upon Amin's approval. Otherwise, Amin uses public transport. The location of both Amin's hotel and coffee shop are submitted to `Bus Schedule WS`, which returns the buses' numbers Amin has to ride. Potential traffic jams in Oslo force `Bus Schedule WS` to regularly interact with `Traffic WS` that monitors the state of the traffic network. This state is submitted to `Bus Schedule WS` so, changes in buses' numbers are made.

Amin scenario yields insight into the multiple challenges that designers of Web services-based systems face. Some challenges include: how to identify the relevant Web services, how to orchestrate the Web services as part of their composition, how to make the Web services context-aware, how to use policies to define the behavior of the Web services, and how to run the Web services subject to resource availability?

2.2 $\mathcal{CP}4\mathcal{WS}$ Steps

This section details the steps that make up $\mathcal{CP}4\mathcal{WS}$ and illustrates them based on Amin scenario.

Step 1 - User needs identification and specification. It consists of identifying and specifying user needs. Traditional techniques that permit identifying user needs and requirements are used in $\mathcal{CP}4\mathcal{WS}$ such as interviews with end-users, information collection on similar applications, and literature review. Regarding the specification technique of user needs, $\mathcal{CP}4\mathcal{WS}$ adopts UML use-cases. This has several advantages: most designers are familiar with use-cases,

identified use-cases could be mapped onto potential Web services, and interactions between use-cases are an excellent indicator of the composition-based collaboration between future Web services.

Fig. 1 presents a part of the use-case model we developed for Amin scenario. Several actors are shown like Amin, Melissa, weather forecast agency, and transport agency. In addition, several use-cases are identified like itinerary development, weather forecast establishment, and traffic monitoring. The same figure shows how some use-cases interact with each other, e.g., itinerary development use-case "uses" traffic monitoring use-case.

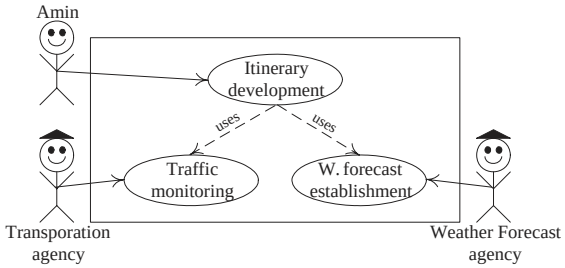

Figure 1: Part of Amin scenario's UML use-cases.

Step 2 - Web services orchestration. It consists of specifying the orchestration of the Web services that will constitute a composite Web service. The types (in term of functionality) of the required Web services are already identified based on the outcome of user needs identification and specification step. We recall that a use-case is a potential candidate to be mapped onto a Web service, although the mapping is not always one-to-one.

In $\mathcal{CP}4\mathcal{WS}$, the Web services orchestration step relies on the concept of service chart diagram. This diagram enhances an UML state chart diagram (Booch, G. and Rumbaugh, J. and Jacobson, I., 1998) with emphasis on the elements that surround the execution of a Web service rather than only on the states that a Web service takes. To this purpose, the states of a Web service are wrapped into five perspectives: state, flow, business, information, and performance. Additional details on service chart diagrams are given in (Maamar et al., 2005).

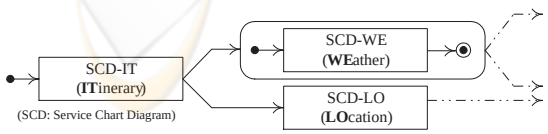

Figure 2: Orchestration of Web services in Amin scenario.

Because a composite Web service is made up of several component Web services, the process model that underlies the composite Web service orchestration is specified as an UML state chart diagram. In this diagram, states are associated with service chart diagrams, and transitions are labeled with events, conditions, and variable assignment operations. Fig. 2 shows the specification of the composite Web service, i.e., Web services orchestration, we developed for Amin scenario. Three component Web services are listed: ITineray (IT), WEather (WE), and LOcation (LO). It should be noted that service chart diagrams and state chart diagrams are used at the design level. At the implementation level, both diagrams could for example be mapped onto a BPEL specification.

Step 3 - Web services contextualization. It consists of structuring and specifying the contexts of the multiple participants that take part in Web services composition. In addition to Web services as default participant, $\mathcal{CP}4\mathcal{WS}$ considers the following additional participants (Maamar et al., 2005): composite Web service, resource, and user. Each participant has a context that is labeled as follows: $\mathcal{W}$-context for context of $\mathcal{W}$eb service, $\mathcal{C}$-context for context of $\mathcal{C}$omposite Web service, $\mathcal{R}$-context for context of $\mathcal{R}$esource, and $\mathcal{U}$-context for context of $\mathcal{U}$ser. The specification of each type of context means defining the arguments that will populate the context structure.

In $\mathcal{CP}4\mathcal{WS}$ context returns details of various levels of granularity. These details are for example on Web services like execution order, on composite Web services like forthcoming executions of Web services, on users like current locations, and on resources like next periods of unavailability. It will be shown in the next step in $\mathcal{CP}4\mathcal{WS}$ how all these contextual details affect the process of loading and firing policies. To keep the paper self-contained, we only list the arguments per type of context. These arguments will be later instantiated according to the application domain.

$\mathcal{W}$-context encompasses the following arguments: label, maximum number of active participations, number of active participations, next possibility of participation, resource&state per active participation, previous Web services per active participation, current Web services per active participation, next Web services per active participation, regular actions, reasons of failure per active participation, corrective actions per failure type and per active participation, and date.

$\mathcal{C}$-context is built upon the $\mathcal{W}$-contexts of the component Web services and consists of the following arguments: label, previous component Web services, current component Web services, next component Web services, status per component Web service, time, and date. It should be noted that status per com-

ponent Web service argument is service-dependent since this argument's value is collected from status argument of $\mathcal{W}$-context.

$\mathcal{R}$-context consists of the following arguments: label, maximum number of component Web services, number of active component Web services, next acceptance of component Web services, previous component Web services per active composition, current component Web services per active composition, consumption&state per component Web service and per active composition, next component Web services per active composition, and date

$\mathcal{U}$-context consists of the following arguments: previous locations/component services, current location/component services, next locations/component services, previous periods of time/component services, current period of time/component services, next periods of time/component services, and date.

Step 4 - Web services behaviors specification. It consists of specifying the behavior of the Web services that were identified in Step 2. A behavior is first, exposed with a set of attributes and second, specified with a set of policies. $\mathcal{CP}4\mathcal{WS}$ uses the following attributes to define the behavior of a Web service: permission, restriction, dispensation, and obligation. $\mathcal{CP}4\mathcal{WS}$ suggests, also, the use of the Web Services Policy Language (WSPL) for policy specification (Anderson, 2004). The syntax of WSPL is strictly based on the OASIS eXtensible Access Control Markup Language (XACML) standard. Other policy specification languages like Ponder and WS-Policy could also be used.

In the following, we describe first, how the attributes that define a Web service's behavior are interpreted and second, how a Web service binds a certain behavior as a result of policy triggering.

- Permission: a Web service accepts the invitation of participation in a composite Web service upon validation of its current engagements in other concurrent composite Web services. This acceptance is backed by executing the appropriate policies.
- Restriction: a Web service is not allowed to connect some peers of a composite Web service due to non-compliance of these peers with this Web service's requirements.
- Dispensation: a Web service breaks some policies related to either permission or restriction. In case of permission, the Web service will not participate in a composition despite the positive permission of participation. In case of restriction, the Web service will be connected to some peers despite the existence of restrictions.
- Obligation: this is a strong permission for a Web service to participate in a composite Web service despite the negative permission of participation or the existence of restrictions from participation. Obligations override dispensations of types permission and restriction.

To keep again the paper self-contained, we show how permission policy is defined in $\mathcal{CP}4\mathcal{WS}$ using WSPL. Permission authorizes a Web service to be part of a composite Web service. It states that a Web service participates in a composition subject to evaluating <Condition> to true. This condition refers to some $\mathcal{W}$-context's arguments like number of current active participations of the Web service in compositions and next possibility of participation of the Web service in additional compositions. <True/FalseConclusion> shows the permission/rejection of participation, "CrtNbrPar" stands for current number of participation, and "MaxNbrPar" stands for maximum number of participation.

```
Policy (Aspect="Permission") {
<Rule xmlns="..." RuleId="PermissionWS">
<Condition>
 <Apply FunctionId="and">
 <Apply FunctionId="integer-less-than" DataType="boolean">
 <SubjectAttributeDesignator AttributeId="CrtNbrPar".../>
 <SubjectAttributeDesignator Attributeid="MaxNbrPar".../>
 </Apply>
 <SubjectAttributeDesignator AttributeId="Availability".../>
 </Apply>
</Condition>
<Conclusions>
 <TrueConclusion Permission = "Permit"/>
 <FalseConclusion Permission = "Deny"/>
</Conclusions> </Rule>}
```

Step 5 - Web services deployment. It consists of managing the performance of Web services on top of computing resources. The rationale of this step is to satisfy the needs of providers who have interests and/or obligations in strengthening or restricting the execution of Web services on their resources. For instance, a Web service is not accepted for execution on a resource because of this Web service's no-compliance with the resource's requirements such as agreed execution-time.

$\mathcal{CP}4\mathcal{WS}$ suggests managing the deployment of Web services on resources with policies. For standardization purposes with the policies of Web services' behaviors, WSPL is also adopted to specify these policies. A deployment policy for permission is about a Web service that receives the necessary authorizations from a resource to be executed over this resource. The authorizations are based on the state of the resource, which is reflected using its respective $\mathcal{R}$-context. The following illustrates a deployment policy for permission in WSPL. It states

that a resource accepts the execution request of a Web service subject to evaluating <Condition> to true. This condition refers to some arguments like number of active component Web services that the resource supports their execution and next acceptance of the resource for additional component Web services. In the policy, <True/FalseConclusion> shows the positive/negative permission of execution, "NbrActWS" stands for number of active Web services, "MaxWS" stands for maximum number of Web services, and "NxtAcc" stands for next acceptance of Web services. In case of positive permission of execution, yes-permission-deployment procedure is executed, which results in updating the following arguments: resource&state per active participation of $\mathcal{W}$-context of the Web service and number of active component Web services of $\mathcal{R}$-context of the resource.

```
Policy (Aspect="PermissionDeployment") {
<Rule xmlns=" ...  RuleId="PermissionDeploymentWS">
<Condition>
<Apply FunctionId="and">
<Apply FunctionId="integer-less-than" DataType="boolean">
 <SubjectAttributeDesignator AttributeId="NbrActWS" ...
 <SubjectAttributeDesignator AttributeId="MaxWS"...
</Apply>
 <SubjectAttributeDesignator AttributeId="NxtAcc" ...
</Apply>
</Condition>
<Conclusions>
<TrueConclusion>
<proc:do> yes-permission-deployment </proc:do>
</TrueConclusion>
<FalseConclusion>
<proc:do> no-permission-deployment </proc:do>
</FalseConclusion>
</Conclusions>
</Rule>}
```

3 IMPLEMENTATION OF AMIN SCENARIO

We discuss the work we carried out as part of the implementation of Amin scenario using $\mathcal{CP}4\mathcal{WS}$. For compatibility purposes, we used Sun Microsystems's tools namely J2EE 1.4 to develop Web services and XACML Open Source to develop policies.

The prototype architecture comprises 4 types of managers. The `specification manager` supports designers during the specification of composite Web services. This requires identifying the appropriate component Web services. The specification work is carried out through a `composition environment`, which is a set of integrated tools that assist designers in creating and editing new and existing specifications of composite Web services, respectively. We use a composition environment that was developed in a previous project (Maamar et al., 2005). This environment, also, supports translating composite Web service specifications, like the one shown in Fig. 2, into BPEL specification. The `selection manager` is responsible for identifying the component Web services that satisfy user needs. This manager is trigged upon user's request, who identifies the appropriate composite Web service specification. In the current system, the selection is not only driven by the resulting functionality of the composition the user needs (e.g., to reach a meeting place by taxi or by bus according to weather conditions). It also considers Web services' QoS parameters that affect the selection process like response time, performance, and throughput. These constraints are expressed with WSPL policies. The `policy manager` makes Web services bind appropriate behaviors according to a composition progress.

Finally, the `context manager` keeps track of the contexts of users, Web services, and resources. Contexts' arguments are of different types and their values change over time. Therefore the context manager is supported with a real-time triggering mechanism that feeds context parameters with fresh values. Details of contexts are structured as XML files. Before sending the selected Web services' addresses to the user for invocation, the `policy manager` ensures that these Web services comply with the different policies. Upon approval of the `policy manager`, the `selection manager` initiates the search of the resources on which the Web services will operate.

4 RELATED WORK

Web services are a very active area of R&D. However, to our knowledge, few projects have aimed at suggesting design and development methods for Web services based on context and driven by policies. We present in the following some projects that helped shape $\mathcal{CP}4\mathcal{WS}$. These projects are related to either context or policies for Web services.

In $\mathcal{CP}4\mathcal{WS}$, context is part of the modeling exercise of a system based on Web services. Other projects such as (Breener and Schiffers, 2003) use Web services for managing context provisioning. Breener and Schiffers envision that context information will typically be provided by autonomous organizations (or context providers), which means heterogeneity and distribution challenges to deal with. Additional challenges are cited in (Breener and Schiffers, 2003) including what is the optimal sequence for

gathering and combining the required context information, how to secure the whole context provisioning process, and how is the cooperation between the providers of context achieved, and even enforced?

Barkhuus and Dey identified three levels of interactivity for context-aware applications (Barkhuus and Dey, 2003): personalization, active context-awareness, and passive context-awareness. For both authors, personalization is motivated by the diversity and dynamics featuring nowadays applications. For active context-awareness, it concerns applications that, on the basis of sensor data, change their content autonomously. In passive context-awareness applications present the updated context to the user and let the user specify how the application should change.

In (Baresi et al., 2005), Baresi et al. proposed a policy-based approach to monitor Web services' functional (e.g., constraints on exchanged data) and non-functional (e.g., security, reliability) requirements. In this approach the authors report on the different types of policies that can be defined along the life cycle of a Web service (Mukhi et al., 2004). These policy types are service policies, server policies, supported policies, and requested policies. The combination of requested and supported policies results in effective policies.

5 CONCLUSION

In this paper, we presented $\mathcal{CP}4\mathcal{WS}$ method, which is one step towards offering the appropriate support to those who are responsible for the design and development of Web services-based systems. Composition of Web services addresses the situation of a user's request that cannot be satisfied by any single, available Web service, whereas a composite Web service obtained by combining a set of available Web services might be used. The core concepts of $\mathcal{CP}4\mathcal{WS}$ are context, policy, service chart diagram, state chart diagram, and resource. By developing context-based, policy-driven Web services, it would be possible to handle the aspects of the environment in which these Web services will operate. These aspects are multiple and can be related to users (e.g., stationary, mobile), time of day (e.g., in the afternoon, in the morning), etc.

ACKNOWLEDGEMENTS

$\mathcal{CP}4\mathcal{WS}$ development benefited from the discussions the authors had with A. Anderson, G. Kouadri-Mostéfaoui, S. Sattanathan, and Ph. Thiran.

REFERENCES

Anderson, A. H. (2004). An Introduction to The Web Services Policy Language (SWPL). In *Proceedings of The 5th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'2004)*, New-York, USA.

Baresi, L., Guinea, S., and Plebani, P. (2005). WS-Policy for Service Monitoring. In *Proceedings of The 6th Workshop on Technologies for E-Services (TES'2005) held in conjunction with The 31st International Conference on Very Large Data Bases (VLDB'2005)*, Throndeim, Norway.

Barkhuus, L. and Dey, A. (2003). Is Context-Aware Computing taking Control away from The User? Three levels of Interactivity Examined. In *Proceedings of The Fifth International Conference on Ubiquitous Computing (UbiComp'2003)*, Seattle, Washington, USA.

Booch, G. and Rumbaugh, J. and Jacobson, I. (1998)). *The Unified Modeling Language User Guide* . Addison-Wesley Professional.

Breener, M. and Schiffers, M. (2003). Applying Web Services Technologies to the Management of Context Provisioning. In *Proceedings of The 10th Workshop of the OpenView University Association (OVUA2003)*, Geneva, Switzerland.

Coutaz, J., Crowley, J. L., Dobson, S., and Garlan, D. (2005). Context is Key. *Communications of the ACM*, 48(3).

Keidl, M. and Kemper, A. (2004). A Framework for Context-Aware Adaptable Web Services. In *Proceedings of The 9th International Conference on Extending Database Technology (EDBT'2004)*, Heraklion, Crete.

Lupu, E. and Sloman, M. (1999). Conflicts in Policy-Based Distributed Systems Management. *IEEE Transactions on Software Engineering*, 25(6).

Maamar, Z., Benslimane, B., Mrissa, M., and Ghedira, C. (2006a). $\mathcal{C}$oo$\mathcal{PS}$ - Towards a Method for Coordinating Personalized Services. *Software and System Modeling Journal Special Section on "Service-Based Software and Systems Engineering", Springer Verlag*, 5(2).

Maamar, Z., Benslimane, D., and Anderson, A. (2006b). Using Policies to Manage Composite Web Services. *IEEE IT Professional*, 8(5).

Maamar, Z., Kouadri Mostéfaoui, S., and Yahyaoui, H. (2005). Towards an Agent-based and Context-oriented Approach for Web Services Composition. *IEEE Transactions on Knowledge and Data Engineering*, 17(5).

Mukhi, N., Plebani, P., Silva-Lepe, I., and Mikalsen, T. (2004). Supporting Policy-Driven Behaviors in Web Services: Experiences and Issues. In *Proceedings of The 2nd International Conference on Service Oriented Computing (ICSOC'2004)*, New York City, NY, USA.