A K-Based Specification of Web Services

Manel Amel DJENOUHAT
LIRE Laboratory,
University of Constantine 2,
Algeria
CEDRIC Laboratory,
CNAM,Paris,France
djenouhat.manel@gmail.com

Faïza BELALA
LIRE Laboratory,
University of Constantine 2,
Algeria
belalafaiza@hotmail.com

Kamel BARKAOUI
CEDRIC Laboratory,
CNAM,Paris,
France
barkaoui@cnam.fr

ABSTRACT

The Web service description language WSDL has been the subject of numerous researches and known a constant evolution. Nevertheless, it is almost impossible to define a semantics allowing a flawless interoperability. Indeed, in spite of its success, WSDL doesn't ensure conveniently its role in the service interaction process due to its lack of operational semantics which often leads to a loss of behavior control and creates a dichotomy in the composition process.

Therefore, various approaches have been conducted in order to formalize the semantics of service description, but while coupling more than two formalisms. To cope with this problem we propose the provision of a semantic support to Web service description language by using Rewriting logic.

This article sets out to study an extension of the interface grammar of WSDL by endowing it with a rewrite rules-based semantics using the K framework. Our contribution is twofold; on the one hand, it gives a rewrite operational semantics to the WSDL language and on the other hand, it provides a transparent WSDL-Maude translation. Thus, the description stemming from this translation is executable and analyzable under Maude system taking advantage from its environment as well as its tools.

Categories and Subject Descriptors

D.2.4. [**Software Engineering**]: Formal methods,

D.2.11. [**Software Engineering**]: Software Architectures,

D.2.12. [**Software Engineering**]: Interface definition languages.

General Terms

Design, Languages.

Keywords Service Oriented Architecture (SOA), K-Framework, Maude, WSDL.

1. INTRODUCTION

The rise of New Information and Communication Technologies ("NICT") allows glimpsing new solutions for the development of distributed architectures which bodes a real revolution in the industrial and commercial domains. Indeed, the maturity of these architectures, the use of Web technologies and the pressure of the world economy lead to a frantic race towards the conception and the distribution of autonomous physical components on Web. Defined as "Web services", these interacting components are described and located in a specific working environment based on a set of standards currently represented by a three-layer model of the overall operating process.

This model results in: a standard offering a communication protocol allowing structuring the exchanged messages between services **SOAP** [1], a second describing their interfaces **WSDL** [2] and a third giving a specification of their publication and location **UDDI** [3].

WDSL plays a paramount role in this process; in addition to giving the service interface description, it must ensure that this description guarantees its interworking with other services considering the universality vocation which is associated to them.

However, if the simple Web services admit an abstract description, where, at this level the interoperability is wide in view of the standardization of the semantics offered, it is not the case for more complex components. Indeed, the WSDL description, in this case, becomes inappropriate because these services require more complex exchange models.

This problem thus imposes the search of more effective and rigorous means to associate a formal semantics to this language type.

In this work, we propose a solution to mitigate this problem by the definition of a suitable method for Web service description based on the transformation of the WSDL language into an equivalent rewriting model more flexible and usable.

Our contribution is centered on:

1-The definition of the operational semantics of the WSDL description.

2-The use of the executable K semantic framework [4] based on the Maude language as formalism for defining and transforming this semantics into a rewrite theory.

3- The exploitation of the proposed model in the Web services dynamic invocation context.

The remainder of this paper is organized as follows. Section 2 looks at the related work of the method used .In section 3, we present the WSDL basic concepts and give an overview on the K tool, especially its fundamental elements and notations. We dedicate the section 4 to our contribution where we give our general K-based WSDL definition approach ; then we illustrate step by step, our WSDL-Maude translation Section 5 deals with the summary of results obtained and Section 6 concludes by evoking some problems that will be studied and corrected in a near future.

2. RELATED WORK

Although Web services are accessed and executed via the Web, their descriptions are only semi-formal. The consensus established on the mechanical interactions formats is not sufficient to enable them to interact unequivocally. The automation of discovery and composition of services is a key element of scalability; explicit semantics definition must be injected in their description.

Many studies have been conducted in order to add "meaning" to service descriptions. The idea is to link a semantics to the already existing WSDL description. In this context, we distinguish two initiatives of researches:

2.1 Ontology-Based Semantics

In [5] authors have allowed the addition of semantic annotations in service (service call method, exchanged messages, etc.) and also in an XML schema (for any element of the parameter). Therefore, it is possible to link virtually every element of the service definition to an ontological class, i.e., to add meaning to every element of syntactic description. By the same way, authors in [6] proposed a complete development framework platform to simplify the creation of semantic applications. This particular platform defines mediation methodology for semantic service orchestration. We note also that OWL-S language introduced in [7] adds a semantic description to a service or a set of services ontology. A service is defined through three parts: Profile, Process Model and Grounding Model. The Profile describes the service capabilities in order to inform the consumer of his service. The Process Model specifies the behavior of the service. If the service is a composite service, it will define its composition with the task model. Finally, Grounding Model specifies the technical aspects of the service including the necessary information's to its calling. Often this is a link to the WSDL file of the service.

However, most of these approaches have shortcomings; their semantic enrichment mechanisms proceed either by semantic descriptions of the standard syntactic annotations, either by extending these description languages. Thus, no hierarchical or relational semantic classification of services is supplied.

2.2 Mathematical Formalism-Based Semantics

Several formal semantics have already been introduced to define services description languages. For instance, a precise but informal operational semantics was given in [8]; we consider this as the standard against which the success of any formal operational semantics should be measured. A formal asynchronous operational semantics has been proposed in [9]; but since this asynchronous semantics allows some undesirable behaviors, a refinement of this semantics into a synchronous one, distinguishing between internal and external actions was also given in [9].

Various denotational semantics are well-suited for reasoning about identities and algebraic laws in the language rather than for describing the operational behavior of programs, Moreover, a denotational semantics was given in [10], it used labeled event structures to analyze dependencies in program execution. Furthermore, in [11] authors gave encodings of orchestration in Petri nets and the join calculus that reveal some of the subtleties of the semantics of the language. Ian Wehrman et al. in [12] proposed a relative-time operational semantics of orchestration by extending the asynchronous semantics relation of [9] to timed events and time-shifted expressions.

Most recently, Latreche Fateh et al. introduced in [13] a rewrite-logic based semantics for analyzing dynamic Web services.

In the same context, our work, has similarities with the various semantics that have been studied and particularly, with the approach given in [13]. Indeed, we propose an approach based on rewriting logic, which provides an adequate formal semantic framework for WSDL and attempts to evade some known problems due to the application of a method stemming from an hybridization of the various formalisms cited above. Further, our approach is implemented under K benefitting not only from the inherent theoretical aspects of rewriting logic but also from the K framework that insures a transformation of the WSDL language to Maude in a totally transparent way.

3. FUNDAMENTAL CONCEPTS

In this section, we introduce some basic concepts of the WSDL and K languages.

For a better understanding, we divide the WSDL document structure in different fragments represented by the figures 1 and 2. The K tool role meanwhile, is described by its architecture.

3.1 WSDL Overview

Before invoking a Web service, the user must first locate it and makes sure that it satisfies his needs. This correspondence is established by using the WSDL description of this service.

WSDL(Web Service Description Language) is a description language which supplies a model and an XML format [14] to describe a Web service contract.

The structure of a WSDL document consists of seven essential elements, which can be divided, into structural (see figure1) and behavioral (see figure 2) categories.

In figure 1, we give as example the UC2.dz Web service, written in WSDL. This service aims to manage students registration to a university training, it allows a student to seek a university training and then to enroll in it.

<u>**StudentRegistrationService**</u>

Messages :
```
<wsdl:message name="TrainingsList">
                    <wsdl:part name="TrainingInfos"
 element="xsd:List"/>
</wsdl:message>
 <wsdl:message name="addstudent">
                    <wsdl:part name="Infosstudent"
                       element="tns:student"/>
</wsdl:message>
<wsdl:message name="registrationPayment">
                   <wsdl:part name="RegistrationFees"
                    element="tns:StudentAccount"/>
</wsdl:message>
<wsdl:message name="RegistrationConfirmation">
                  <wsdl:part name="NumCardstudent"
                   element="xsd:String"/>
</wsdl:message>
```
Types:
```
<wsdl:types>
      <xsd:schema
targetNamespace="http://www.uc2.dz/student.xsd"
       xmlns: xsd = "http://www.w3.org/2001/XMLSchema">
      <xsd:element name ="student">
          <xsd:complexType>
            <xsd:sequence>
               <xsd:element name="Stadress" type="xsd:string" />
                <xsd:element name="StName" type="xsd:string" />
            <xsd:element name="StFirstName" type="xsd:string" />
                     </xsd:sequence>
             </xsd:complexType>
           </xsd:element>
    </xsd:schema>
    <xsd:schema
</wsdl:types>
```
Port types:
```
<wsdl:portType name ="NewStudentAccountPortType">
```
 Operations
```
       <wsdl:operation name="createNewStudentAccount">
            <wsdl:input message="AddStudent"/>
            <wsdl:input message="RegistrationPayment"/>
           <wsdl:output message="TrainingsList"/>
           <wsdl:output message="RegistrationConfirmation"/>
          </wsdl:operation>
</wsdl:portType>
```

Figure 1. WSDL Structural Category

The description of the abstract category of a WSDL document is given in figure 1. The **messages** concept describes the data exchanged between the Web service provider **UC2.dz** and its consumer **Student**. Before adding a student to the database, the **UC2.dz** service gives first the trainings list, once added, the student is supplied to pay his registration fees to get a confirmation.

InfosStudent,**RegistrationFees**,**NumCardStudent** are respectively the parameters of each message cited above. Each parameter is associated with a concrete data type. We distinguish in this example two complex data types: **student** and **studentAccount** and a standard type **String**.

A Web service needs to define its inputs and outputs and how they are mapped into and out of services. The WSDL types, in this example, take care of defining the **Student**, **StudentAccount** and **String** data types that are used by the **UC2.dz** Web service. The **PortType** combines multiple messages to form a complete one way or round-trip operation.

AddStudent,**RegistrationPayment** represent the input operations and **trainingsList**, **registrationConfirmation** the output ones.

Binding :
```
<wsdl:binding name="NewStudentAccountBinding"
                type="NewStudentAccountPortType">
 <soap:binding style="rpc"
        transport="http://schemas.xmlsoap.org/soap/http"/>
```

```
  <wsdl:operation name="CreateNewStudentAccount">
    <soap:operation

soapAction="http://wwww.UC2.dz/”CreateNewStudentAccount"/>
     <wsdl:input>
        <soap:body
             use="encoded"
namespace="http://wwww.UC2.dz/Student"

encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
     </wsdl:input>
     <wsdl:output>
        <soap:body
           use="encoded"namespace
            ="http://wwww.UC2.dz/”CreateNewStudentAccount “

encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
Port:
<wsdl:port  binding=" NewStudentAccountBinding "
                 name=" NewStudentAccountPort">
             <soap:address

location="http://www.UC2.dz:8856/soap/servlet/rpcrouter">
</wsdl:port>
Service :
<wsdl:service name="NewStudentAccountService">
                 <documentation>add a new Student</documentation>
          <wsdl:port binding=" NewStudentAccountBinding"
                         name="NewStudentAccountPort">
             <soap:address

location="http://www.UC2.dz:8856/soap/servlet/rpcrouter">
         </wsdl:port>
</wsdl:service>
```

Figure 2. WSDL Behavioral Category

The figure 2 represents the WSDL behavioral category. The **binding** concept provides concrete information on the protocol used to transfer the **PortType** operations. The **UC2.dz** service is represented by its **CreateNew-AccountStudentBinding** binding which provides all information about the used SOAP protocol describing the different interactions between services.

The port **NewStudentAccountPort** defines an individual endpoint by specifying a single address **http://www.UC2.dz:8856/soap/servlet**

/rpcrouter for the **NewStudentAccountBinding** binding. The **service** concept defines all the ports: **NewStudentAcco-untPort**, bindings:

CreateNewAccountStudentBinding and addresses supported by the **UC2.dz** Web service.

3.2 K Tool Presentation

The scope for using formal methods in the area of Web services is much wider, and the challenges raised by this new area can offer opportunities for applying these formal techniques to model more structured and complex services. Generally, most existing works focusing on semantic specification of Web service use hybrid models. The objective of the suggested work is to integrate a universal Web services description language in the rewriting logic framework and its formal language Maude in a transparent manner using K tool.

K was introduced in 2010 by Gigore Rosu [15], it represents an executable formal semantic framework based on the rewriting language Maude. It was initially conceived for the definition of programming languages, computing systems as well as type systems or tools of formal analysis.

The main objective of K is to prove that a formal specification language can be at the same time simple, comprehensive, analyzable and executable [15].

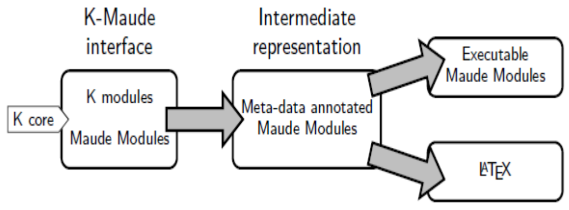


Figure 3. The K-Tool Architecture

In the architecture of K presented in figure 3 above, Core groups the elements which constitute K techniques added to Maude's platform. The interface manages K modules which are interpreted in intermediate Maude modules including the K notations in the form of meta data. Then, they are transformed into executable Maude modules or in Latex files for documentation purpose. In K system, a language is specified by its syntax and its operational semantics, the notion of module in K is inspired from Maude modules.

The syntax of a language in K is defined according to the BNF (Backus-Naur Form) notion [16]. On the other hand, the specification of semantics is constituted of three parts:

1-**Strictness strategies**: to represent the link between the syntax and the semantics of the language. They specify in which order the arguments of a syntactic construction must be evaluated.

2-**Configuration:** to describe the state of the system during execution.

3- **K-rules:** to specify semantics of transformation from a term to another one. These rules are similar to rewrite rules in rewrite logic.

The K tool presents a lot of advantages ,it supplies a simple and effective means for :

- extending the syntax of an existing language by the possibility of enriching it with new concepts and elements in answer to susceptible appearing needs.
- making a specification executable and consequently allow the concrete exploitation of the model.
- Offering a high level of abstraction by the definition of a meta-model including all the language concepts.
- Analyzing and verifying systems properties in a formal way by the use of the various mechanisms of analysis and check offered by the Maude language (K is implemented on the top of Maude).

4. K-BASED SEMANTICS FOR WSDL

Our contribution in this work is twofold. On the one hand, we define a sound semantic model based on rewriting logic for Web services description, and on the other hand, inferring semantics to this services behavior will become possible thanks to K-Maude tool. We illustrate the interests of our formalization approach on the invocation operation.

4.1 Principle

For that, we were inspired from the WSDL language intended basically to describe the Web service interface, this standard language in spite of its use by the majority of the Web services community has some syntactic and especially semantic gaps. Our purpose is to develop a specification model for this language allowing to enrich its grammar at the syntactic level, then, at the semantic one, by inferring it meaning.

4.1.1 Step 1:Abstract Syntax

In keeping with already existing works based on K, particularly, those adapted to the programming languages, we use this formal framework to define an equivalent syntax to that of the WSDL language.

WSDL being based on the XML language requires a first necessary transformation of its syntax according to the BNF (Backus-Naur Form) in order to be exploitable in K. The resulting definition integrated as a module represents a formal meta-model defining all generic aspects presented previously in section 3-1.

```
Module KWSDL-SYNTAX
Syntax K-WSDL ::= "KWSDL" "ServiceName"ServiceId ":"
"{" DescriptionPart "}"
syntax DescriptionPart ::= "messages" ":" {Message"}"
                              | "types" "=" "{"Type"}"
                              | "ports" "=" "{"Port"}"
                              | "bindings" "=" "{"Binding"}"
                              | "service" "=" "{"Service"}"
                            >DescriptionPart DescriptionPart [left]
syntax Message ::= MsgId  TypeMsg ":" Msg
syntax TypeMsg ::= "request" |"response"
syntax Msg ::=String
syntax Type ::= TypeId ":" DataType
syntax Port ::= PortId ":" PortType TypeMsg ":" Msg
syntax PortType ::= "Input" |"Output"
syntax Binding ::= BindingId PortId Protocol Style "=" Body
syntax Service ::= ServiceId BindingId PortId  Location
syntax Protocol ::="SOAP" |"SMPT"|"HTTP"
```

Figure 4. The K-Syntax of WSDL

The figure 4 describes a part of the WSDL syntax in KWSDL-SYNTAX module.

A KWSDL description starts with the keywords **KWSDL** and **ServiceName** followed by a set of sub-descriptions **DescriptionPart**.

Each description part refers to a K representation of a WSDL standard concept.

For example, the equivalent K syntax to that of a WSDL message concept takes the form:

Message ::= MsgId TypeMsg ":" Msg and where respectively:

MsgId gives the message name, **TypeMsg** specifies if this one is used for a request or a response and **Msg** represents the conveyed message.

Through the presented K modules of this section, we achieved a modular and legible specification of Web service. The expressiveness and generality of K syntax allow us declaration of both user defined operators. By the same way, this specification can be easily enriched, particularly; we can add some elements to specify behavior features. Another important fact is that each deduced Maude module (see figure 3) specifies not just a theory, but a precise high-level mathematical model. Hence, this model is executable and will serve to the formal checking of any Web services based system.

4.1.2 Step 2: Operational Semantics

The WSDL standard presents some limits related to the low expressiveness-level of its syntactic description, often limited to the enumeration of operations list and their associated input/output types parameters. It doesn't characterize the semantics of the functionality performed by the service. For instance, we notice that a student can use the service **UC2.dz** written in WSDL (section 3-1), just for search purposes, to get information about available trainings, without making registration. Syntactically, the "search" term differ from the "registration" one, however, the two concepts are semantically related, the first being a pre-requisite of the second. A search by keywords of a service allowing to find a university training does not return services which allow the student to register. The student will not be informed about all the potentially relevant services. This kind of subtlety can be by-passed when the service will have a more elaborate semantic description. To remedy to this conflict problem annoying the interoperability between Web services, we propose operational semantics for WSDL language that we integrate in K framework while importing the syntactic module described in the previous step, including also the configuration and the rewriting rules. This additional structure gives a well semantics to syntactical declarations of WSDL.

```
configuration
<state color="yellow">
    <k color="green"> $PGM:WSDL </k>
<definitions color="cyan" >
<message color="orange">.Map</message>
<portType color="red" multiplicity ="*">
            <PortName> "port" </PortName>
             <operation>.Map </operation>
 </portType>
 <binding color="Orchid" multiplicity ="*">
             <bindingName>"binding"</bindingName>
             <bindingPortName>"bindingPortName"
              </bindingPortName>
               <protocol>"protocol"</protocol>
               <style>"style"</style>
<inputOperations>.Map</inputOperations>
       <outputOperations>.Map</outputOperations>
 </binding>
<service color="green" multiplicity ="*">
       <serviceName>"nameservice"</serviceName>
      <bindingName>"binding"</bindingName>
      <bindingPortName>"bindingPortName"
</bindingPortName>
       <location>.Map </location>
 </service>
<Exchange color="red">.Map </Exchange>
</definitions>
…</state>
```

Figure 5. The K-Configuration of a WSDL file

The configuration that represents the global service state is described in figure 5.

The <state> cell represents the state of the service and it includes all the other cells for sub-states. The <k> cell contains the KWSDL description from which the K tool begins its execution.

For instance, the <service> cell describes each service state.

```
Module KWSDL
imports Module KWSDL-SYNTAX // defined in figure 4
rule <k>...port X  = {O} => X ~> O ...</k>
(. =><portType>
<PortName> X </PortName>
 <operation>.</operation>   </portType> )
                                    -a-
rule <k>... X ~> TP:Type T:TypeMsg : MS:Msg  =>          X  ~>  .
...</k>
<portType>
<PortName> X  </PortName>
               <operation>Rho:Map       (.    =>    TP    T    |->    MS
)</operation>
 </portType>
rule <k>... service S ~>  U:Loc : AS:AdresseService  => S
~> . ...</k>
                                    -b-
<service> …<serviceName> S </serviceName>
                 <bindingName>B</bindingName>
                <bindingPortName>BP</bindingPortName>
                <location>Rho:Map(. => U |-> AS )</location>
</service>                          -c-
```

Figure 6. Semantics of Syntactic Declarations

The WSDL semantics in K is defined by a set of rewriting rules acting on the system state after being specifically transformed into internal structures. In figure 6 above, we give the definition of k-rules that allow associating to the various service description parts, their internal structures to be manipulated by the K tool. For example, the transformation of a port P having a set of operations O is achieved by the first k-rule of figure 6a. The result can be found in the internal structure described by the <portType> cell.

```
rule <services>…
<serviceName>X</serviceName>
 <PortName>P </PortName>
<Operations>...Out P :PortId |->(RequestMsg :String=>.)
…</Operations>
<binding>..
<BindingName>Y</BindingName>
<Binding>… In B:BindingPort |->
(Request:String=>Resquest+ResquestMsg)…</Binding>

<exchange > Rho:Map</exchange>
 When $hasMapping(Rho,X.P)
andBool (Y.B==K Rho:Map (X.P))
andBooL Request==String ""
…
rule
<BindingName>Y</BindingName>
<Binding>… In B:BindingPort |->
Request Out P:BindingPort  |->
ResquestMsg=> ResquestMsg +String Request)…</Binding>
When RequestMsg==String  "" andBool  Request=/=String ""

rule <services>…
<serviceName>X</serviceName>
 <PortName>P </PortName>
<Operations>...In P :PortId
|->(RequestMsg :String=> Resquest+ResquestMsg)
…</Operations>
<binding>..
<BindingName>Y</BindingName>
<Binding>… Out B:BindingPort |->
(Request:String=>.</Binding>

<exchange > Rho:Map</exchange>
 When $hasMapping(Rho,X.P)
andBool (Y.B==K Rho:Map (X.P))
andBooL RequestMsg==String "" …
```

Figure 7. K-Operational Semantics

The definition of the executable operational semantics of WSDL consist of providing a set of rewriting rules which model the state change of the system. These rules specify in this case, the exchange of messages between services via ports and bindings. Figure 7 shows through these rules the modeling of the service behavior, where a request R is send from an output port P (of a service X) to an input Binding port J (of a Binding B) . In the same way, we define the conveyance of messages from an Input Binding port to its Output one and from an Output Binding port to an input Port of a service.

The existing languages such as OWL, SAWSDL, etc, have merely introduced the semantics in WSDL at the syntactic representations level, but none operational semantics was defined yet. However, the main aim of this approach is to define not only the syntactic structure of WSDL statements, but also their behavior. Besides, this K based operational semantics of WSDL is effective, executable and its Maude translation is done in a transparent way.

4.2 Formal Definition of Invocation

This step comes to complete previous both steps intended mainly for the transformation of Web service description language WSDL to a formal language endowed of an executable Maude operational semantics.

Indeed, a service is described in order to be invoked by third parties. The standard SOA invocation is described via the SOAP protocol which describes the interactions between different services including the exchange of messages made between them to answer a user request. Therefore, a nice consequence of our contribution is to define an operational formal semantics, based on K rewrite rules, for the exchange protocol.

```
….
rule
 <k>... exchange B: BindingName. BP:BindingPortName
 to S:ServiceName  . R:Request   => . ...</k>
<Exchange> Rho:Map (. => B . BP |-> S . R)</Exchange>
```

Figure 8. K-Definition of Service Invocation

The formal semantics of a service invocation in K is translated by the above rule (see figure8), this rule describes the exchange message process between services.

A request R is transmitted from an output port P of the service requester to the service provider (the service holding the requested address) via the input port and the binding partners.

Informally, the invocation of the **UC2.dz** service is as follows:

A student sends a request and invokes the **UC2** service for enrolling to a university training, once added to the database, he receive his student card number as a registration confirmation .This code written in K reproduces the interaction (user/service) between a student and the **UC2.dz** service.

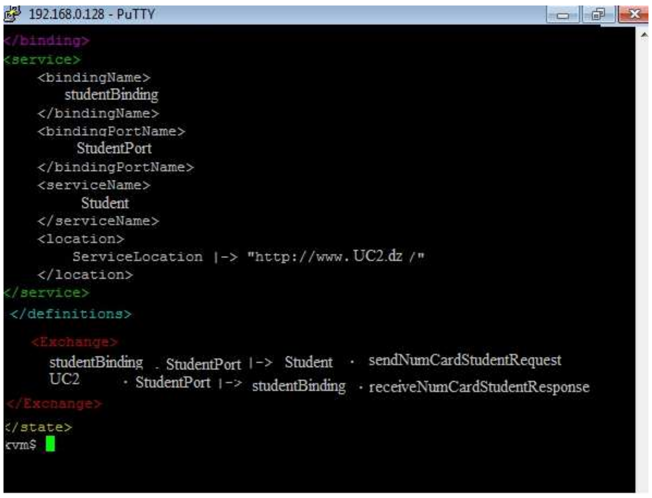
```
KWSDL Service Name Student:
{
messages : { addstudent  request : "NumCardStudentRequest"
                  RegistrationConfirmation                    response
:"NumCardStudent-
                  response" }
 ports : { StudentPort  = Input request : "NumCardStudentRequest"
            StudentPort=                                      Output
response:"NumCardStudentresponse"}
bindings : { StudBinding StudentPort SOAP rpc =
                  EncodingStyle = "http://schemas.xmlsoap.org/soap/
                                    encoding/"
                  Tns = "http://www.UC2.dz/wsdl/wiki.wsdl"
                 use = encoded}
 service : {     Student
                 StudBinding
                  StudentPort
                 ServiceLocation :"http://www.UC2.dz/"          }

 exchange  UC2 . Studentport to
 StudBinding . SendnumCardStudentRequest

exchange  Student .receiveNumCardStudentResponse to
 StudBinding .Studentport
}
```

Figure 9. Invocation Example

The scenario of interaction derived from the mutual invocations done between the two services is illustrated by the figure 10 below:



```
192.168.0.128 - PuTTY
</binding>
<service>
    <bindingName>
        studentBinding
    </bindingName>
    <bindingPortName>
        StudentPort
    </bindingPortName>
    <serviceName>
        Student
    </serviceName>
    <location>
        ServiceLocation |-> "http://www.UC2.dz /"
    </location>
</service>
 </definitions>

   <Exchange>
     studentBinding . StudentPort |-> Student  ·  sendNumCardStudentRequest
     UC2         · StudentPort |-> studentBinding  · receiveNumCardStudentResponse
</Exchange>
</state>
kvm$
```

Figure 10. Invocation Execution Results

The specificities of K language let us associating a simple, concise and executable specification to the Web services description and invocation. This prototype has been tested and evaluated by several rewrites under the K-Maude version [17].Our specification is the more simple and extensible one. It will integrate simply other rewrite rules and functionalities. It will even serve generally to the conception of other Web service languages such as BPEL (Business Protocol Execution Language).

5. TOWARDS A K-BASED FRAMEWORK FOR WEB SERVICES

In this section, we tempt to situate our actual work in a general framework having several phases, each one is compared to the already existing standard phase. Obviously, we note that for this paper, we intervene in the description, invocation and analysis phases.

Our work contribution consists in developing a framework, based on a particular well-founded language K, having a proof and prototyping environment (figure 11). It helps us to concentrate and reason on the true semantics of Web service. With this framework in hand, any Web service description (including WSDL interface description) can be transformed into an easy-understand rewriting theory with the underlined categorical model, in a transparent manner.

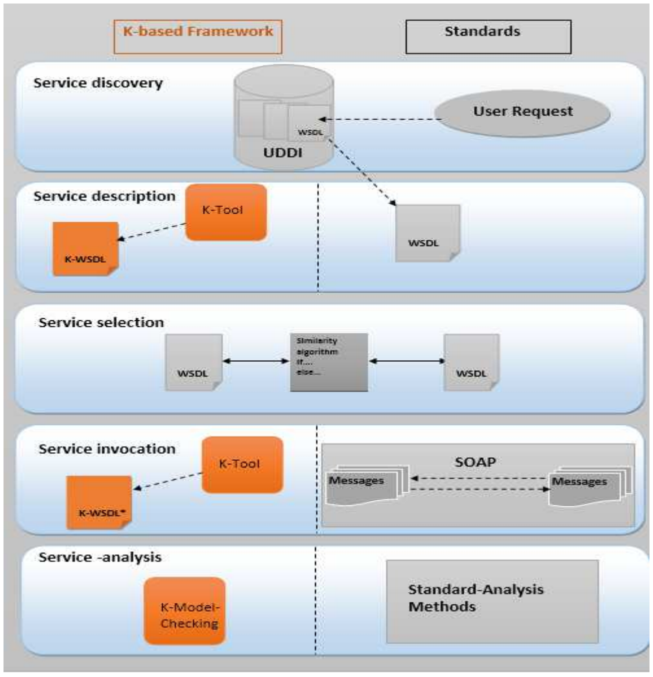


Figure 11. The K-Based Framework for Web Services

In the selection phase, we will test the various similarities between K-WSDL contracts and this both on the syntactic and semantic level, knowing that current test methods do it purely syntactically. The K tool is fully extending Maude tool, thus it benefits from its various tools, including model checker which can be applicable in each of the previous steps. The most important checking remains to the last step, where we can formally verify several inherent properties (accessibility, efficiency, etc.) of Web services.

6. CONCLUSION

We have put the accent on the use of formal methods in the Web services description. We tempted to demonstrate the utility to have a powerful and efficient logic to specify syntax aspects and operational semantics of languages devoted to manipulate Web services. In this goal, we associated to WSDL language a rewriting theory, in transparency, using the K-Maude framework. The rewriting rules describe the operations semantics on Web services, such as invocation, since interoperability constitutes an important aspect in Web service interaction. Indeed, we elaborated independent K-modules, able to be extended easily by several other aspects. Besides, these modules have been tested syntactically and analyzed formally.

In a forthcoming paper, we will complete the proposed K-based framework for Web services, by developing new tools for the analysis and the check of the various properties of the services (K-model checking).Also, we will deal with the dynamic composition and selection of the Web services under this framework.

7. REFERENCES

[1] Don Box, David Ehnebuske, Gopal Kakivaya, Andrew Layman, Noah Mendelsohn,Henrik Frystyk Nielsen, Satish Thatte, and Dave Winer. "Simple object access protocol (SOAP) "1.1, 2000.

[2] Web Services Description Language (WSDL), W3C, June 2007, http://www.w3.org/TR/wsdl/

[3] Universal Description, Discovery and Integration, Ariba, IBM and Microsoft, http://www.uddi.org.

[4] Traian Florin Şerbănuţă,Grigore Roşu, "K-Maude: A Rewriting Based Tool for Semantics of Programming Languages", Rewriting Logic and Its Applications Lecture Notes in Computer Science Volume 6381, 2010, pp 104-122.

[5] J. Kopecky, T. Vitvar, C. Bournez, et J. Farrell. " SAWSDL : Semantic annotations for wsdl and xml schema".IEEE Internet Computing, 11(6) :60–67, 2007.

[6] C.Feier, D. Roman, "Towards intelligent web services : The Web service modelingcontology (WSMO) ". International Conference on Intelligent Computing (ICIC) 2005.

[7] David Martin, Massimo Paolucci,"Bringing semantics to webservices : The owl-s approach"..Lecture Notes in Computer Science, 3387 :26–42, 2005.

[8] Jayadev Misra. Computation orchestration: A basis for wide-area computing. In Manfred Broy, editor,Proc. of the NATO Advanced Study Institute,Engineering Theories of Software Intensive Systems, NATO ASI Series,Marktoberdorf, Germany, 2004.

[9] Jayadev Misra and William R. Cook. Computation orchestration: A basis for wide-area computing .Journal of Software and Systems Modeling, May 2006.

[10] Sidney Rosario, David Kitchin, Albert Benveniste, William Cook, Stefan Haar, and Claude Jard. Event structure semantics of ORC. In 4th International Workshop on Web

Services and Formal Methods (WS-FM2007),Brisbane, Australia, October 2007.

[11] Roberto Bruni, Hern´an Melgratti, and Emilio Tuosto. Translating Orc features into petri nets and the join calculus.Web Services and Formal Methods, pages 123–137, 2006.

[12] an Wehrman, David Kitchin, William R. Cook, and Jayadev Misra. A timed semantics of Orc.Theoretical Computer Science, July 2007.

[13] Fateh Latreche, Faiza Belala , A Novel Semantic Framework for Analyzing Dynamic Web Services, 20th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises, WETICE 2011, Paris, France, 27-29 June 2011.

[14] Extensible Markup Language (XML) 1.0, Second Edition, Tim Bray et al., eds., W3C, 6 October 2000, http://www.w3.org/TR/REC-xml .

[15] Traian Florin Serbanuta, Andrei Arusoaie, David Lazar, Chucky Ellison, Dorel Lucanu and Grigore Rosu ,The K Primer (version 2.5) Technical Report, January 2012

[16] Garshol L. M., "BNF and EBNF: What are they and how do they work?" 2008 http://www.garshol.p-riv.no/download/text/bnf.
html#id2.

[17] Traian Florin Serbanuta, Andrei Arusoae, Chucky Ellison, David Lazar, Dorel Lucanu, Radu Mereuta, Grigore Rosu Collection of K utilities (interpreters, visualizers, state-space searchers, model checkers) 2010.