Automatic Conversion of a Conceptual Model to a Standard Multi-view Web Services Definition

https://doi.org/10.3991/ijes.v6i1.8285

Anass Misbah(✉), Ahmed Ettalbi
Mohammed V University in Rabat, Morocco
anassmisbah@gmail.com

Abstract—Information systems are becoming more and more heterogeneous and here comes the need to have more generic transformation algorithms and more automatic generation Meta rules. In fact, the large number of terminals, devices, operating systems, platforms and environments require a high level of adaptation. Therefore, it is becoming more and more difficult to validate, generate and implement manually models, designs and codes.

Web services are one of the technologies that are used massively nowadays; hence, it is considered as one of technologies that require the most automatic rules of validation and automation. Many previous works have dealt with Web services by proposing new concepts such as Multi-view Web services, standard WSDL implementation of Multi-view Web services and even further Generic Meta rules for automatic generation of Multi-view Web services.

In this work we will propose a new way of generating Multi-view Web services, which is based on an engine algorithm that takes as input both an initial Conceptual Model and user's matrix and then unroll a generic algorithm to generate dynamically a validated set of points of view. This set of points of view will be transformed to a standard WSDL implementation of Multi-view Web services by means of the automatic transformation Meta rules.

Keywords—Multi-view, Web services, Standard, WSDL, Automatic transformation, Generation, Algorithm

1 Introduction

Web services are widely used in most information systems; this technology brings flexibility, interoperability and reusability. Thus, in this work we will continue improving the concept of Multi-view Web services by proposing a generic algorithm allowing the automatic transformation of a standard Conceptual Model and user's matrix to a set of Multi-view Web services. This algorithm takes under consideration all methods defined within the Conceptual Model (each method is considered as a view), check if the users have the rights to access these methods and then generate the points of view accordingly.

Subsequently, another algorithm is used to eliminate redundancy and to make each point of view unique and therefore to create a set of validated Points of view that can

be implemented directly as a standard Multi-view WSDL definition. This implementation can be reached through the generic Meta rules defined in the work of [1].

In the following section we will present brief description of Web services and Multi-view Web services, as well as a quick reminder of the Meta rules concept defined in our previous work [1].

In section 3 we will present our approach which consists of a schema of automatic transformation, an inference algorithm engine and also an algorithm to eliminate redundancy. Section 4 is dedicated to study cases illustrating the automatic transformation of a School Conceptual Model, and an Online Library Conceptual Model to a set of validated Points of view in addition to an automatic implementation of these points of view to standard Multi-view WSDL definitions.

Afterwards, in section 5 we will highlight the main advantages of our proposed approach, and then we will conclude in section 6 by giving some perspectives.

2 State of the art

2.1 Web Services

Web services can be seen as a specific case of SOA (Service Oriented Architecture) [6]. The main concept is to consider each task or multiples tasks that can be done separately as a standard function that can be invoked through a standard protocol and with a standard input and output messages format. The internet protocol used is HTTP (HyperText Transfer Protocol), and the messages can be exchanged using SOAP (Simple Object Access Protocol) or REST (REpresentational State Transfer) wrappers [7].

The Web services are hosted in a server called the Service provider, and the Web services definition is described through a standard XML format called WSDL (Web Services Description Language) [8]. Each Web service should be referenced in a Directory such as UDDI (Universal Description Discovery and Integration). Once the client find the Web service definition through the Directory, the request and response process can be done directly between the Client and the Server.

2.2 Multi-view Web Services

The Multi-view concept is related basically to the integration of the end user access rights in an early phase of the analysis and design. This notion has been incorporated in many UML diagrams and models including the work of [2] and [3]. Many advantages have been reached using this concept, such as enhancing security, avoiding redundancy and proposing a new user oriented approaches.

The main notions of this concept are Views and Points of view:

- **Views**: Are a representation of tasks that can be done separately (similar to Web services methods or functions)
- **Points of view**: Are a set of views that can be done by one or multiple users

The concept of Multi-view has been integrated with Web services in the work of [4] through the concept of Web services decomposition. Each user has a point of view which is composed of views that this user is allowed to use.

An implementation of this concept of composition has been proposed in the work of [5], this implementation is based on the extension of the existing WSDL standard format to a WSDL-Us specific format.

2.3 The Standard WSDL Definition of Multi-view Web Services

In [1] we have proposed an improvement of the work of [5] by introducing generic Meta rules for automatic generation of a standard Multi-view WSDL definition. This improvement concerns mainly three points:

- **Generic approach**: Automatic transformation of Web services to standard Multi-view Web services
- **Standard compliant**: The used WSDL syntax to generate the output file is completely standard
- **Quality of the script**: The generated standard WSDL is clearer and can be integrated natively with existing Web services platforms

Bellow a reminder of the proposed footsteps to follow for standard Multi-view WSDL automatic generation in the work of [1]:

1. Define the business objects as a "complex type" in types section of WSDL document
    - This definition represents a description of objects that will be exchanged through sub Web services invocations
2. Define a simple type called "Profile" which is basically a restriction that contains all users profiles "Points of view"
    - Each point of view will be transformed into a value of this enumeration, in addition to a generic value called "All" that will be used in case of an operation that can be invocated by all users. This tag means essentially that the concerned operation will have the same behavior whoever is the caller
3. Define Input and Output messages for each sub Web service (operation)
    - For each Input and Output massage we define the data structure that will be used as a parameter of the operation (business objects or simple types)
    - For each operation Input massage we define a parameter called "user" that stands for the user profile, this parameter will take inevitably a value among the enumeration "Profile" values. This key notion will allow us to implement Multi-view Web service in the WSDL definition
4. Define the main Web service as a port type and then define each sub Web service as an operation. The previous messages defined will be used as the Input and Output parameters for the operations

3 Proposed approach

In this work, we will go further with the same concept of automating the generation of Multi-view Web services by proposing a transformation schema "see Figure 1" that illustrates inputs and outputs of a Transformation Algorithm. This algorithm is the fundamental component of our proposed approach; its main role is to transform both a standard Conceptual Model and user matrix to a set of Points of view. The set of Points of view can be implemented with different mechanisms. In this work we will use the approach of [1] to implement the Points of view and generate a standard Multi-view WSDL definition.

3.1 Schema of the Automatic Transformation

The automatic conversion process is presented in "Figure 1". It considers mainly two inputs:

- **The Standard Conceptual Model**: This model is composed of the modeled system entities and each entity contains methods allowing interaction with this entity
- **The users table (or user's matrix)**: Is actually a set of the system users and for each user the access rights to each entity
- The transformation process is achieved using two algorithms:
- **Inference algorithm "see Figure 2"**: Its main objective is to generate a set of Points of view.
- **Redundancy elimination algorithm "see Figure 3"**: This is used to delete the redundant Points of view.

As a result of the transformation, a set of Points of view is generated and can be implemented to generate Multi-view Web services.

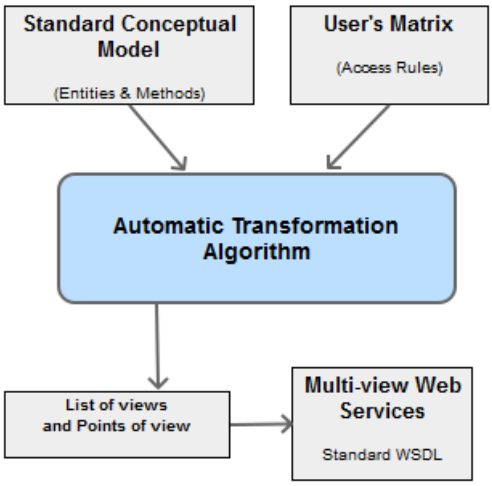

Fig. 1. Automatic transformation process

3.2 Transformation Algorithms

The first algorithm "see Figure 2" deals with the constitution of the Points of view sets. For each user U, we associate a set of methods called Point of View and named PointOfView(U) which is initially empty. Then the principle is to browse all methods of the Standard Conceptual Model and all users then check each time if the current user has the right to access the current method. If it is the case, the method is added to the user's Point of view set.

```
Foreach user U, we associate a Point Of View set called PointOfView(U),
   Initially, PointOfView(U) is empty for each user U
   Foreach Method m
      Foreach User U
        If U can use m
        //This condition can be checked via the user's matrix
           Add m to PointOfView(U)
        End If
      Loop
Loop
```

Fig. 2. Inference algorithm of Points of view automatic generation

The previous algorithm will generate a set of Points of view; each user has one point of view.

However, we may have in some cases multiple users with the same points of view, and here comes the need to eliminate redundancy. The following algorithm "see Figure 3" will go through all generated Points of view, and if two Points of view are found with the same content, then they are merged to one Point of view, so that to eliminate redundancy and make each point of view unique.

The "Figure 3" bellow represents the script of the redundancy elimination algorithm.

```
Foreach user U
     Foreach user V
       If PointOfView(U) == PointOfView(V)
          Delete PointOfView(V)
          SetName PointOfView(u) ← PointOfView(u,v)
       End If
     Loop
Loop
```

Fig. 3. Algorithm of redundancy elimination

4 Study cases

In this section, we will present 2 study cases. The first one concerns the School which contains three entities: Course, Exam and Absence. The second case study is the online library which consists of two entities: the books and the CDs. We will present in each case study the Conceptual Model, the user's Matrix and the set of Points of View generated with our proposed algorithm.

4.1 School Study Case

The associated Conceptual Model is shown in "Figure 4". This model is composed with three entities. Users that can interact with these entities are: Teacher, Student or Administrator. Hereafter a description of those entities:

- **Course**: Creating and managing all aspects related to online courses.
- **Exam**: Creating and managing all aspects related to the school exams.
- **Absence**: Creating and following up absences within the school.

Each method is a set of operations that can be done by the same user profile; therefore, each method stands for a view.

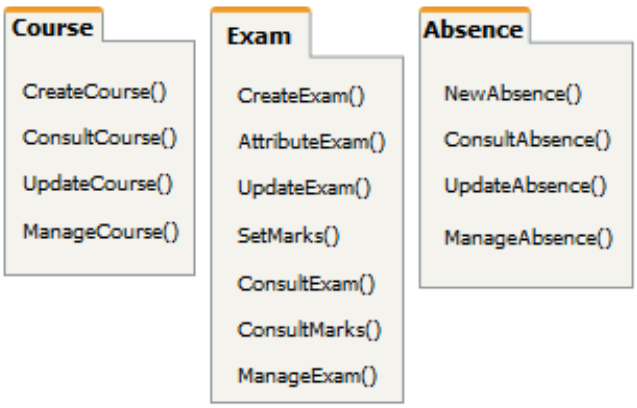

Fig. 4. Conceptual Model of the School study case

After defining the Conceptual Model, we need to specify the user's matrix which is in reality a table of system users and entities. This user's matrix is presented in "Table 1". For each entity we specify the operations (V**iews**) that can be done by each user.

Both the Conceptual Model "see Figure 4" and user's matrix "see Table 1" will serve as an input for the transformation Algorithm.

Table 1. User's matrix of the School study case

	Teacher	**Student**	**Administrator**
Course	Can update or consult a course	Can consult a course	Create or manage a course
Exam	Can update, setMarks, consultMarks or consult an exam	Can consult an exam and consultMarks	Create, attribute or manage an exam and consultMarks
Absence	Can create or consult an absence	Can update an absence	Consult, update or manage an absence

After applying the transformation algorithm, a set of validated Points of view is generated automatically. eIn this specific study case, the generated set of Points of view is already unique after the first algorithm, thus the second algorithm will generate the same set of Points of view.

The "Table 2" below represents the generated set of Points of view.

Table 2. Automatically generated Points of view of the School study case

PointOfView (Teacher)	**PointOfView (Student)**	**PointOfView (Administrator)**
ConsultCourse() UpdateCourse() UpdateExam() ConsultExam() SetMarks() ConsultMarks() NewAbsence() ConsultAbsence()	ConsultCourse() ConsultExam() ConsultMarks() UpdateAbsence()	CreateCourse() ManageCourse() CreateExam() AttributeExam() ManageExam() ConsultMarks() ConsultAbsence() UpdateAbsence() ManageAbsence()

All points of view are different, so no delete will be done.

This output "see Table 2" can be implemented in many ways. In our work, since we are dealing with Multi-view Web services, we chose to implement this result as a standard Multi-view WSDL using the approach proposed in [1]. This will allow an automatic transformation and validation of the generated set of Points of view to a standard Multi-view WSDL.

After applying Meta rules as specified in [1], the generated WSDL is shown in "see Figure 5".

```
<?xml version="1.0"?>
<definitions name="CourseService"
  targetNamespace="http://example.ma/Course.wsdl"
  xmlns:tns="http://example.ma/course.wsdl"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns="http://schemas.xmlsoap.org/wsdl/">

 <types>
```

```
<schematargetNamespace="http://example.ma/course.xsd"
   xmlns="http://www.w3.org/2000/10/XMLSchema">
    <element name="Course">
        <complexType name="CourseObject">
        //Business objects to be defined here
        </complexType>
        </element>
        <element name="Profiles">
        <simpleType name="Profile">
            <restriction base="xsd:String">
            <enumeration value ="All">
            <enumeration value ="Teacher">
            <enumeration value ="Student">
            <enumeration value ="Admin">
            </restriction>
        </simpleType>
        </element>
        </schema>
    </types>

  //Exchanged input and output parameters for Course views
    <message name="CreateCourseInput">
        <part name="user" element="Profiles" value="Admin">
    </message>
    <message name="CreateCourseOutput">
        <part name="result" element="String">
    </message>
    <message name="ConsultCourseInput">
     <part name="user" element="Profiles" value="Teacher ;
  Student">
    </message>
    <message name=" ConsultCourseOutput">
        <part name="result" element="String">
    </message>
    <message name="UpdateCourseInput">
     <part name="user" element="Profiles" value="Teacher">
    </message>
    <message name="UpdateCourseOutput">
        <part name="result" element="String">
    </message>
    <message name="ManageCourseInput">
      <part name="user" element="Profiles" val-
  ue="Administrator">
    </message>
    <message name="ManageCourseOutput">
```

```
            <part name="result" element="String">
        </message>

  //Exchanged input and output parameters for Exam views could
be defined the same way as Course Views …

  //Exchanged input and output parameters for Absence views
could be defined the same way as Course Views …

  //All views operations definition for Course Multi-view Web
service
        <portType name="Course">
        <operation name="CreateCourse">
        <input message="tns:CreateCourseInput"/>
        <output message="tns:CreateCourseOutput"/>
            </operation>
        <operation name="ConsultCourse">
        <input message="tns:ConsultCourseInput"/>
        <output message="tns:ConsultCourseOutput"/>
            </operation>
      <operation name="UpdateCourse">
        <input message="tns:UpdateCourseInput"/>
        <output message="tns:UpdateCourseOutput"/>
            </operation>
      <operation name="ManageCourse">
        <input message="tns:ManageCourseInput"/>
        <output message="tns:ManageCourseOutput"/>
        </operation>
        </portType>

  //All views operations definition for Exam Multi-view Web
service could be defined the same way as Course Multi-view Web
service …

  //All views operations definition for Absence Multi-view Web
service could be defined the same way as Course Multi-view Web
service …

  </definitions>
```

Fig. 5. The standard WSDL definition of Multi-view Web services automatically generated of the School study case

4.2 Online Library Study Case

As to emphasis the sustainability of our approach, we will use another study case and go through the same process as the previous one. We chose an Online Library Conceptual Model "see Figure 6" which is composed of two entities: Book and CD.

Users that can interact with those entities are: Author, Reader and Customer.

The entities are described as follow:

- **Book**: Represents all readable articles that might be posted up online for sale.
- **CD**: Corresponds to all listening materials that might be posted up online for sale.

Each method stands for a view as long as it can be done by one or multiple users.

The "Figure 6" bellow represents the Online Library Conceptual Model study case.

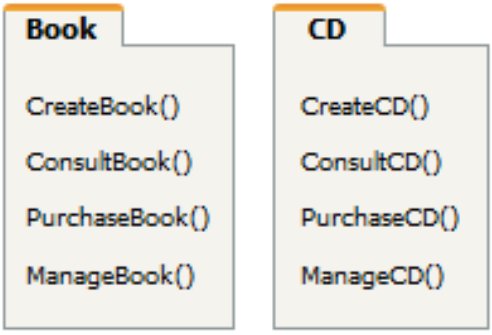

Fig. 6. Conceptual Model of the Online Library study case

Once the Conceptual Model is ready, we define the users table (user's matrix) as shown in "Table 3". This matrix allows specifying for each user the views that he actually has regarding each entity.

"Table 3" shows the users matrix for the Online Library study case.

Table 3. User's matrix of the Online Library study case

	Author	Reader	Customer
Book	Can Create or Manage a Book	Can consult or purchase a Book	Can consult or purchase a Book
CD	Can Create or Manage a CD	Can consult or purchase a CD	Can consult or purchase a CD

Even though both users Reader and Customer have the same views in the conceptual model, the Customer has previous purchases persisting data, while the Reader is browsing the Library materials for the first time.

After applying the inference algorithm engine (first step in the transformation process), a set of validated Points of view is generated automatically "see Table 4".

Table 4. Automatically generated Points of view of the Online Library study case

PointOfView (Author)	PointOfView (Reader)	PointOfView (Customer)
CreateBook() UpdateBook() CreateCD() UpdateCD()	ConsultBook() PurchaseBook() ConsultCD() PurchaseCD()	ConsultBook() PurchaseBook() ConsultCD() PurchaseCD()

In this study case, we can notice that the generated set of Points of view are not unique after the first algorithm, thus the result of the second algorithm will not generate the same set of Points of view "see Table 5". In fact, as the Reader and the Customer have both the same Points of view, these Points of view will be merged to one unique Point of view named (Reader, Customer).

The "Table 5" below represents the output of the second transformation algorithm (second step: redundancy elimination)

Table 5. A unique automatically generated Points of view of the Online Library study case

PointOfView (Author)	PointOfView (Reader,Customer)
CreateBook() UpdateBook() CreateCD() UpdateCD()	ConsultBook() PurchaseBook() ConsultCD() PurchaseCD()

As with the previous study case, we will implement the automatically generated set of Points of view using the approach proposed in [1] which is based on Meta rules to generate the standard WSDL Multi-view Web services definition.

As a result, the generated WSDL is shown in "see Figure 7".

```
<?xml version="1.0"?>
<definitions name="LibraryService"
  targetNamespace="http://example.ma/Library.wsdl"
  xmlns:tns="http://example.ma/library.wsdl"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns="http://schemas.xmlsoap.org/wsdl/">

 <types>
   <schematargetNamespace="http://example.ma/library.xsd"
     <element name="Library">
         <complexType name="LibraryObject">
         //Business objects to be defined here
         </complexType>
         </element>
         <element name="Profiles">
         <simpleType name="Profile">
             <restriction base="xsd:String">
```

```
                <enumeration value ="All">
                <enumeration value ="A:Author">
                <enumeration value ="R:Reader">
            <enumeration value ="C:Customer">
            </restriction>
        </simpleType>
            </element>
        </schema>
        </types>

  //Exchanged input and output parameters for Book views
    <message name="CreateBookInput">
       <part name="user" element="Profiles" value="A">
       </message>
    <message name="CreateBookOutput">
       <part name="result" element="String">
       </message>
    <message name="ConsultBookInput">
       <part name="user" element="Profiles" value="R,C">
       </message>
    <message name=" ConsultBookOutput">
       <part name="result" element="String">
       </message>
    <message name="PurchaseBookInput">
       <part name="user" element="Profiles" value="R,C">
       </message>
    <message name="PurchaseBookOutput">
       <part name="result" element="String">
       </message>
    <message name="ManageBookInput">
       <part name="user" element="Profiles" value="A">
       </message>
    <message name="ManageBookOutput">
            <part name="result" element="String">
       </message>

  //Exchanged input and output parameters for CD  views could
be defined as with Book Views …

  //All views operations definition for Book Multi-view Web
service
        <portType name="Book">
        <operation name="CreateBook">
            <input message="tns:CreateBookInput"/>
            <output message="tns:CreateBookOutput"/>
```

```
            </operation>
        <operation name="ConsultBook">
            <input message="tns:ConsultBookInput"/>
            <output message="tns:ConsultBookOutput"/>
            </operation>
      <operation name="PurchaseBook">
            <input message="tns:PurchaseBookInput"/>
            <output message="tns:PurchaseBookOutput"/>
            </operation>
      <operation name="ManageBook">
            <input message="tns:ManageBookInput"/>
            <output message="tns:ManageBookOutput"/>
            </operation>
        </portType>

  //All views operations definition for CD  Multi-view Web
service could be defined as with Book Multi-view Web service
…

  </definitions>
```

Fig. 7. The generated standard WSDL definition of Multi-view Web services of the Online Library study case

5 Advantages of the proposed approach

The proposed approach makes it possible to transform a generic Conceptual Model to a standard Multi-view WSDL definition that represents a set of validated views and Points of view. Therefore we can notice the following advantages:

- Automatic transformation of a generic Conceptual Model to a set of Points of view through the proposed transformation algorithms,
- Native validation of the generated set of Points of view since the transformation is reached automatically by means of the transformation algorithms,
- The redundancy is eliminated using the second algorithm. In fact, each Point of view will be unique and different from the rest of the Points of view,
- The generated Points of view set is standard compliant. Thus it can be implemented in different manners, such as the standard WSDL implementation proposed in [1],
- The proposed approach is easily extensible, given that the set of Points of view is automatically generated. We can make any updates on the Standard Conceptual Model or on the user's matrix (Add, Modify or Delete) and then generate automatically the set of Points of view model and then generate the standard WSDL implementation,
- The generated WSDL file is easy to use and to understand as well as easy to implement within the existing Web technologies.

6 Conclusion and perspectives

In this work, we proposed an automatic process to transform a generic Conceptual Model and a user matrix to a set of validated Points of view which represent the Multi-view Web services generated dynamically. Then we put forward an automatic implementation using standard WSDL definition through predefined Meta rules. The result is indeed a standard Multi-view WSDL definition created automatically, and thus the generated output is standard compliant, validated and verify the Multi-view concept's constraints and conditions.

As a future work, we will explore other possible implementations of Multi-view Web services. Furthermore, we will make a PoC (Proof of Concept) of the proposed approach from end to end with a concrete example using one of development frameworks such as Java.

7 References

[1] A. Misbah, A. Ettalbi.: Towards a standard WSDL implementation of Multiview web services. 5th International Conference on Multimedia Computing and Systems (ICMCS'16) – IEEE Conference, 29 September – 1 October 2016, Marrakech, Morocco.
[2] A. Kriouile.: VBOOM, object-oriented analysis and design method by points of view. PhD thesis, Mohammed V University, Rabat, Morocco, 1995.
[3] S. Marcaillou.: Integration of the points of view concept in object modeling - The VBOOL Language. PhD thesis, the Paul-Sabatier University, Toulouse, France, 1995.
[4] R. Boukour, A. Ettalbi and M. Nassar.: Multiview Web Service: The Integration of The Notion of View And Point of View in The Web Services. International Journal of Computer Science and Network Security (IJCSNS), vol. 14 no.2, pp. 31-36, February 2014.
[5] R. Boukour, A. Ettalbi and M. Nassar.: Multiview web service: The description Multiview WSDL of Web Services. International Symposium on Signal, Image, Video and Communications (ISIVC'2014), November 19-21, 2014, Marrakech, Morocco.
[6] M.P. Papazoglou, W. Heuvel.: Service oriented architectures: Approaches, technologies and research issues. The VLDB Journal, Springer Verlag, 2007, pp.389-415.
[7] G. Alonso, F. Casati, H. Kuno.: Web Services: Concepts, Architectures and Applications. Springer Verlag, Heidelberg, Germany, 2004.
[8] W3C: Web Services Description Language (SDL) 1.1. December, 2006.

8 Authors

Anass Misbah is PhD student at IMS Team, ADMIR Laboratory, ENSIAS, Rabat IT Center, Morocco.

Ahmed Ettalbi is Professor at Software Engineering Department of the Higher National School of Computer Science and Systems Analysis (ENSIAS) Rabat, Morocco. His main research interests Object Modeling with Viewpoints, Software Architecture and Business Process Modeling architecture.

Article submitted 22 January 2018. Resubmitted 04 February 2018. Final acceptance 23 February 2018. Final version published as submitted by the authors.