

Developing Medical Information System with MDA and Web Services

Simone A. B. Melo, Denivaldo Lopes and Zair Abdelouahab

Federal University of Maranhão

Campus Bacanga, São Luís, Brazil

simonebandeira@yahoo.com.br, denivaldo.lopes@gmail.com, zair@dee.ufma.br

Abstract

In this paper, we present a medical information system architecture allowing doctors to share information through the Internet. We also discuss a study on medical diagnosis process, describing the entrance and storage of the patients' information and the doctors' interaction. Thus, we propose a medical information system architecture. We develop this architecture through a MDA (Model Driven Architecture) approach in which the PIM (Platform Independent Model) is conforming to UML, and the PSM (Platform Specific Model) is conforming to Web Services Platform. In order to implement this medical information system, we provide metamodels for JWSDP (Java Web Services Developer Pack) and Oracle's framework for Web Services. Thus, we provide transformation definitions from UML (Unifying Modeling Language) to Web Services, JWSDP and Oracle's framework for Web Services.

Keywords: Telemedicine, Medical Information System, Web Services, Model Driven Architecture (MDA).

1. Introduction

In generally, telemedicine is adopted in hospitals and health institutions targeted by other benchmark institutions for data statistics and exchanging. Currently, it is also applied to obtain a second medical opinion when assisting chronic patients, senior citizens and high risk pregnant women. Telemedicine is also applied in small communities with less favorable geographical or socio-cultural conditions [1].

The medical diagnosis analysis presented in this paper was conducted in our geographical region. It is based on discussions held with doctors and on data obtained from medical literature. The interview with doctors is useful to understand how the medical diagnosis is done in practice, while the literature analysis has allowed us to be aware of existing applications on this field.

A problem that teams encounter in software development is the choice of technology to be used to create or evolve a software. Another problem is that software systems developed with a strong technological dependence might feature a reduction on its reuse potential, because its life cycle is associated to the life cycle of the platform chosen. In addition, the incompatibility between the platforms restricts the reutilization process only to the applications being developed in the same platform [3].

Aiming to decrease these effects and improve software development, OMG (Object Management Group) proposes a framework called MDA (Model Driven Architecture) [2], which provides an approach to: create a business model independent

from platform details, and transform this business model to another model dependent from platform. In this approach, the software developers are stimulated to focus more on the modeling of domain business logic, not caring about the platform where this logic will be implemented.

In the MDA approach, it is possible to separate domain business logic, called PIMs (Platform Independent Models), from the platform specific aspects, called PSMs (Platform Specific Models). The whole idea seems quite simple: we use the model transformation to generate a PSM from a PIM [4]. In this work, the PIM is a model for our Medical Information System Architecture with only domain business logic, and the PSM is another model for the same system with additional platform specific aspects of Web Services[13], JWSDP version 1.5 (Java Web Services Developer Pack) [5] and Oracle's framework for Web Services version 10.1.3.0.3 [7]. On the one hand, MDA supports the life cycle development of our system, and on the other hand Web Services support the interoperability on Internet.

This paper has been organized as follows. In section 2, we introduce an overview of medical diagnosis issues, MDA and Web services. In section 3, we present scenarios and an architecture of our medical information system for medical diagnosis. In section 4, we discuss about the model of our medical information system. In section 5, we present a transformation from UML to Web Services (WSDL). In section 6, we propose a metamodel for JWSDP and discuss about transformation from UML to JWSDP. In section 7, we provide a metamodel for Oracle's framework for Web Services

and we depict on transformation rules from UML to this framework. In the last section, we conclude this paper.

2. Background

2.1 Medical Diagnose

A doctor is only able to provide a diagnosis of a probable patient's diseases; afterwards the former has collected a myriad of data from the latter. However, a doctor has not a means of sharing these data and him diagnostic (information) with other physically distant doctors.

Today, medical knowledge is deployed mainly from the major centers which hold outstanding knowledge and better financial and infrastructure resources. Thus, patients need to travel to these centers searching for specialists and better services [1].

The communication between the centers and the locations requiring more specialized information is done through letters, fax, phone calls or e-mails. Communication is not effective concerning rapidity, conveyed data quality and interaction with the source. Therefore, many times it is necessary to carry the patients to the major centers or take the specialist to those places deprived of required knowledge to obtain a more accurate diagnosis [1].

Patient transportation may take some risk, according to seriousness of illness to be treated, extra costs, problems with exceeding number of patients in the major centers, lack of structure to accommodate the patient and his folks.

Concerning the financial, diagnosis quality and knowledge deployment issues, an adequate communication structure should be provided, where data can be accessed by specialists located in geographically distinct localities.

The medical diagnosis proposed on this paper provides the possibility for the doctor to have a second opinion on a faster and interactive basis. In this process, a doctor can use a distributed system of shared medical information to supply a more accurate diagnosis.

2.2 Web Services

Web services provide interoperability in the Internet, and do not depend on any programming language or technology and are universally accepted by the major software enterprises [6][8] [9].

A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems *interact with a Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards* [6].

2.3 MDA (Model Driven Architecture)

MDA aims at accomplishing software development through Platform Independent Models (PIMs) and transforming these models to Platform Specific Models (PSMs). The main idea behind this is to enable software developers to work in a higher abstraction level than the code level. As a consequence, models become the primary artifacts of software development. A model transformation takes one or more source models as input and produces one or more models as output, according to a set of transformation rules [4].

MDA introduces important concepts: Platform-Independent Model (PIM), Platform-Specific Model (PSM), transformation language, transformation rules and transformation engine [4]. In this approach, MOF (Meta Object Facility) [10] is the well established meta-metamodel used to build metamodels. The PIM reflects the functionalities, the structure and the behavior of a system. PSM is more implementation oriented and corresponds to a first binding phase of a given PIM to a given execution platform. PSM is not the final implementation, but has enough information to generate interface files, a programming language code, an interface definition language, configuration files and other details of implementation. Mapping from PIM to PSM determines the equivalent elements between two metamodels¹. Model transformation is accomplished by a transformation engine that executes transformation rules. Transformation rules specify how to generate a target model (i.e. PSM) from a source model (i.e. PIM). To transform a given model in another model, the transformation rules map the elements of the source metamodel to the elements of the target metamodel. The transformation engine takes the source model, executes the transformation rules, and gives the target model as output [4].

3. The Scenarios and Architecture of the Proposed Medical Information System

The possible scenarios that doctors interact with the proposed information system architecture are:

Scenario 1 – The doctor accesses the local medical system. This system has a database (patients, illnesses, medication, exams and symptoms);

Scenario 2 and 4 – The doctor accesses an integrated medical system² which also brings a

¹ Two or more elements of different metamodels are equivalents if they are compatible and they cannot contradict each other [4].

² Integrated medical system – A system storing data from many clinics (pediatrics, orthopedics, gynecology, general physician...) belonging to a medical center.

database (patients, illnesses, medications, exams, symptoms);

Scenario 3 – The doctor has neither a local medical system, nor an integrated one, so, he lacks a database. Under this circumstance, a system developed throughout the Internet can be accessed by the doctor, e.g., he accesses the data through his browser to deliver the diagnosis and ease his task.

Figure 1 illustrates the medical information system architecture proposed in this work.

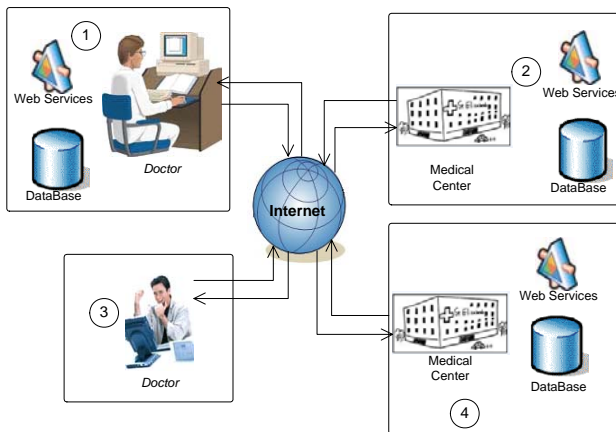


Figure 1 – Architecture of proposed medical information system

The inter-relationship among the scenarios 1, scenario 2, scenario 3 and scenario 4 can be illustrated as follows:

Inter-relationship of scenario 1 with scenario 2 or scenario 4 – The doctor makes use of the local medical system (1), to provide a diagnosis with increased accuracy, he can request more information from a medical center (2 or 4). In this type of inter-relationship, the local medical system (1) turns into a consumer of medical information services and the remote medical center (2 or 4) becomes a provider of these services.

Inter-relationship of scenario 2 with scenario 4 or scenario 1 – the doctor makes use of an integrated medical system from a given center (2) that requests information from another medical center (4). Here, besides the integrated medical system, (2) the doctor can request information for another medical center (4) to provide a clearer and more accurate diagnosis. In this type of inter-relationship the integrated medical system (2) turns into a consumer of medical information services and the remote medical center (4) in a service provider. Another issue is likely to happen within this same inter-relationship is the medical center (2 or 4) becomes a recipient of medical information services and the doctor, with a local medical system available (1) turns into a consumer of these services.

Inter-relationship of scenario 3 with scenario 1, 2 or 4 – the doctor accesses the medical information through Internet. In order to provide his patients a more accurate diagnosis, the doctor (3) becomes a

consumer of medical information services and (1, 2 or 4) he becomes a service provider.

4. Modeling the Proposed Medical Information System

In a medical diagnostic process, a doctor requires a patient to provide him with information about what he is feeling, in order to the former supplies a diagnosis to the latter.

Our proposed medical information system allows to a doctor establishes business relationships with medical centers in order to access shared medical data. Thus, this doctor can take advantages from shared medical data for providing an accurate diagnosis.

4.1 PIM in UML

Figure 2 shows the PIM in UML for the proposed medical information system. In this paper, part of the PIM is presented without the database modeling to simplify our work presentation. In this figure, some classes implement features of the proposed system: Doctor, symptoms, diseases, medication, surgeries, exams, patients, general list, screened list, local list, remote list, differential diagnosis, presumption and conclusive diagnosis. The descriptions of some classes defined in the model are detailed as follows:

4.1.1 - *General list class* defines some methods that represent the general list, how to obtain general list. This class is associated directly to the doctor class to obtain a list with the collected information, in the patient's consultation.

4.1.2 - *Screened list class* defines some methods that represent the screened list, how to show screened list. The screened list class inherits the characteristics of the general list class, but it is shown containing essential information for the patient's diagnosis

4.1.3 - *Local list class* defines some methods that represent the local list, how to create local list. In the local list a list of diseases is returned, with base in the screened list that is last for the local medical center.

4.1.4 - *Remote list class* defines some methods that represent the remote list, how to find bases of shared information, to create remote list. In the remote list class, a list of diseases is returned of the remote medical centers.

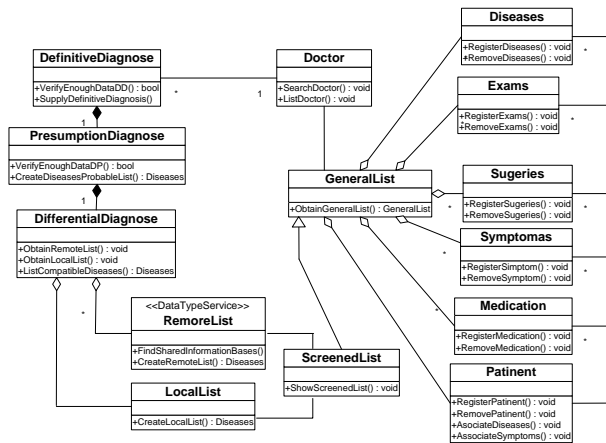


Figure 2 – PIM for our medical information system

4.1.5 - *Differential diagnosis class* defines some methods that represent the differential diagnosis, how to obtain remote list, to obtain local list, to obtain list of compatible diseases. The differential diagnosis is generated with base in the junction of the returned diseases list of the local medical center, and the list of the remote medical center. The differential diagnosis is obtained from a list of compatible diseases with the last information for the local medical center and remote medical centers.

4.1.6 - *Presumption diagnosis class* defines some methods that represent the presumption diagnosis, how to verify the data is enough for the presumption diagnosis and to create list of probable diseases. In the presumption diagnosis the number of hypothetical diseases decreases, in relation to the differential diagnosis

4.1.7 - *Definitive diagnosis class* defines some methods that represent the definitive diagnosis, how to verify the data is enough for the definitive diagnosis. In the definitive diagnosis, the doctor supplies the diagnosis definitively to the patient.

The medical centers provide the services to find the shared medical database, and show the list of diseases of this list so that the doctor can count on a second opinion and a support to guide him in the decision making process.

5. From PIM (UML) to PSM (Web Services)

The MDA is based on transformations among models. The models can be source and target, and they are in compliance with their respective metamodels. In our work, the metamodel source chosen is UML, which is in compliance with MOF. In order to implement the medical diagnosis system, we have chosen the Web Services, JWSDP and

Oracle's framework. Thus, a metamodel for Web Service is required to build platform specific models (PSM) of our medical diagnosis system.

Figure 3 illustrates a WSDL metamodel [4] that is used in our research.

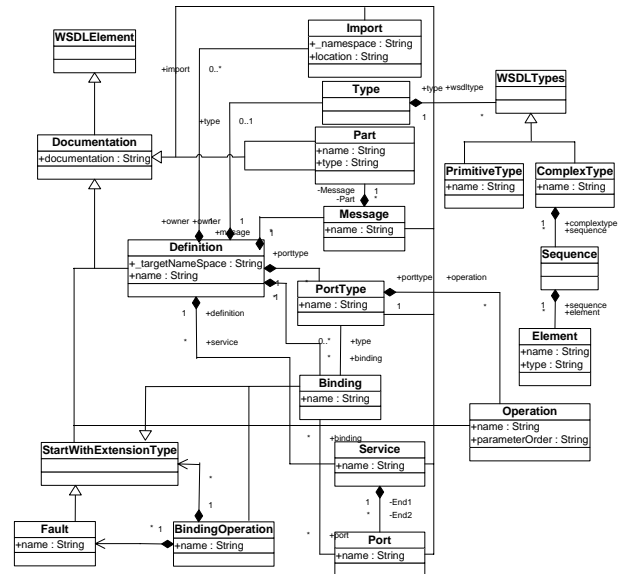


Figure 3 – WSDL Metamodel (fragment) [4]

We found some UML metamodel elements corresponding to the WSDL metamodel elements. This matching is performed by a set of rules which determine the UML metamodel elements equivalent to WSDL metamodel elements.

In our work, we utilize the ATL transformation language [11] to create our transformation rules. The ATL is in compliance with the MDA, and uses a repository to store and manipulate the source and target metamodels, and executes the transformation according to the defined transformation rules. The repository that we use in our work is MDR (Meta Data Repository) [12].

After we identified some equivalent elements between the UML and WSDL metamodels, we can describe the transformation rules from UML to WSDL.

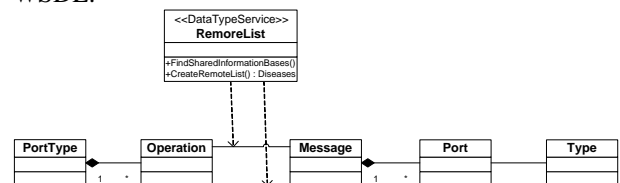


Figure 4 – A matching between a UML and WSDL model

In Figure 4, we present a fragment of equivalence between a UML and WSDL model. A class under the “Service” stereotype in UML corresponds to the PortType in WSDL. A UML method is equivalent to an operation in WSDL. An input or output parameter in UML matches a input or output message in WSDL. If the input or output parameters are in a UML method are based on types, but the primitive ones, e.g., being based in other classes of

UML, then these types will match the Type in WSDL [4].

According to the MDA literature, several rules had already been made [4]. We have improved some rules and proposed new ones as follows:

The UML DataType is mapped to WSDL PrimitiveType through the rule D2P:

```
rule D2P {
  from d : UML!DataType
  to t : WSDL!PrimitiveType(
    name <- d.name
  )
}
```

This rule creates an instance of WSDL!PrimitiveType. The name is set according to the value of d.name.

The UML Class is mapped to WSDL ComplexType through the rule C2C that calls up A2E:

```
rule C2C {
  from c: UML!Class (c.stereotype ->
    exists(e|e.name='DataTypeService'))
  to t: WSDL!ComplexType(
    name <- c.name,
    sequence <- seq
  ),
  seq: WSDL!"Sequence"(
    element <- c.feature->select
    (e|e.ocliIsTypeOf(UML!Attribute))
  )
}
rule A2E {
  from at : UML!Attribute
  to t : WSDL!Element(
    name <- at.name,
    type <- at.type.name
  )
}
```

This rule creates an instance of WSDL!ComplexType. The name is set according to the value of c.name. The sequence is set according to the value of seq (i.e. the created WSDL!Sequence). Seq creates an instance of WSDL!Sequence. The element is set according to the transformation from UML!Attribute to WSDL!Element realized by the rule A2E. In the rule A2E, name is set with the value at.name and the type is set with the value of at.type.name. The execution of these rules (D2P, C2C, A2E) result in a PSM that is presented in the XMI format as follows:

```
<?xml version = '1.0' encoding = 'windows-1252' ?>
<XMI xmi.version = '1.2' timestamp = 'Mon Dec 19 11:21:29 GMT-03:00 2005'>
***
  <XMI.content>
  ***
  <WSDL.ComplexType xmi.id = 'a4' name = 'RemoteList'>
    <WSDL.ComplexType.sequence>
      <WSDL.Sequence xmi.id = 'a5'>
        <WSDL.Sequence.element>
          <WSDL.Element xmi.id = 'a6' name = 'name' type = 'String' />

```

```
          <WSDL.Element xmi.id = 'a7' name = 'list' type = 'ScreenedList' />
        </WSDL.Sequence.element>
      </WSDL.Sequence>
    </WSDL.ComplexType.sequence>
  </WSDL.ComplexType>
  ***
</XMI.content>
</XMI>
```

The transformation from PIM (UML model) to PSM (WSDL model) is a transformation of type model-to-model. In order to obtain the final code (document WSDL), a transformation of type model-to-code is necessary. In other words, another transformation definition in ATL takes this PSM (WSDL) and generates the suitable WSDL document (i.e. XML document).

6. From PIM (UML) To PSM (JWSDP)

Figure 5 presents a metamodel for JWSDP version 1.5, this metamodel was created through the configuration and deployment files. This metamodel has four packages: webservices, configinterface, configwsdl and web. The package webservices has the classes WebServices, WSDDescription, PortComponet, WsdPort and ServiceImplBean. The package ConfigInterface has the classes _Configuration, Service and Interface, the package configwsdl has the classes Configuration and _WSDL and the package web has the classes WebApp, Servlet and ServletMapping. The descriptions of some package defined in the JWSDP metamodel are detailed as follows:

6.1 – The *Configwsdl* package is used to generate the Configwsdl.xml that specifies the place of the file WSDL. The *packageName* attribute specifies the Java package for the generated stubs. The *location* attribute of the WSDL file is specified as a URL.

6.2 – The *ConfigInterface* package is used to generate the ConfigInterface.xml that specifies information on the interface. The *name* attribute is the name of service. The *targetNamespace* and *typeName* attributes are defined as URLs. The *packagename* attribute specifies the package in that the classes of services will be.

6.3 – The *Web* package is used to generate the file web.xml that is used by the Web container to identify the service and to define some properties as: The *servletpattern* element that defines a mapping between a servlet and a URL pattern. The *servletname* attribute is the name of the servlet to which we are mapping a URL pattern. The *urlpattern* attribute describes a pattern used to resolve URLs. The *servlet* element contains the declarative data of a servlet. The *servletname* defines the canonical name of the servlet, used to reference the

servlet definition else where in the deployment descriptor.

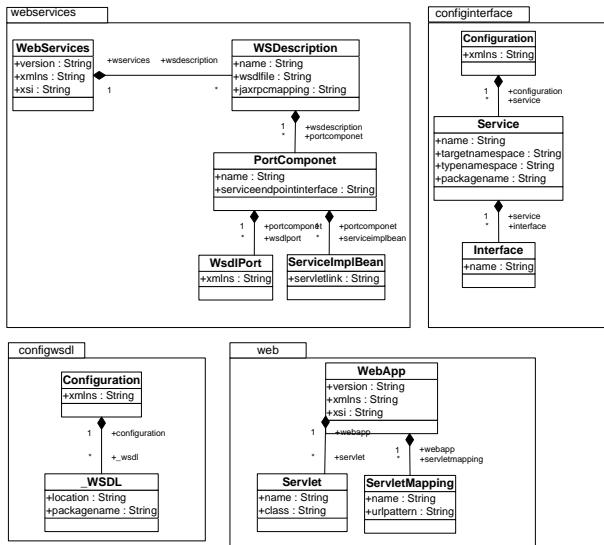


Figure 5 – JWS DP Metamodel (fragment)

The UML Class is mapped to JWS DP ConfigInterface configuration file through the rule C2ConfigInt:

```
rule C2ConfigInt{
  from c : UML!Class (c.stereotype ->
    exists(e|e.name='DataTypeService'))
  to conf : JWS DP!_Configuration(
    xmlns <-
      'http://java.sun.com/xml/ns/jax-
      rpc/ri/config',
    Service <- serv
  ),
  serv : JWS DP!Service(
    name <- c.name,
    targetnamespace <-
      'http://'+c.namespace.name+'.org/wsdl',
    typenamespace <-
      'http://'+c.namespace.name+'.org/wsdl',
    packageName <-
      c.namespace.name,
    Interface <- interf,
    configuration <- conf
  ),
  interf : JWS DP!Interface(
    name <- c.namespace.name
    + '.' + 'interf_' + c.namespace.name,
    Service <- serv
  )
}
```

This MDA approach has allowed the complete generation of WSDL documents, configuration and deployment files for JWS DP and Oracle's framework for Web Services from the PIM of our proposed medical information system. However, this approach has only allowed the generation of Java code skeletons for JWS DP and Oracle's framework for Web Services. Afterwards, we manually completed the java code skeletons.

Some researches [4] have pointed Semantic Action to support the complete generation of code based on programming language as Java.

7. From PIM (UML) to PSM (Oracle's Framework for Web Services)

Figure 6 illustrates a metamodel for Oracle's framework for Web Services. This metamodel is composed of the packages oraclewebservices, web, web services, orion-application, application, and data-sources.

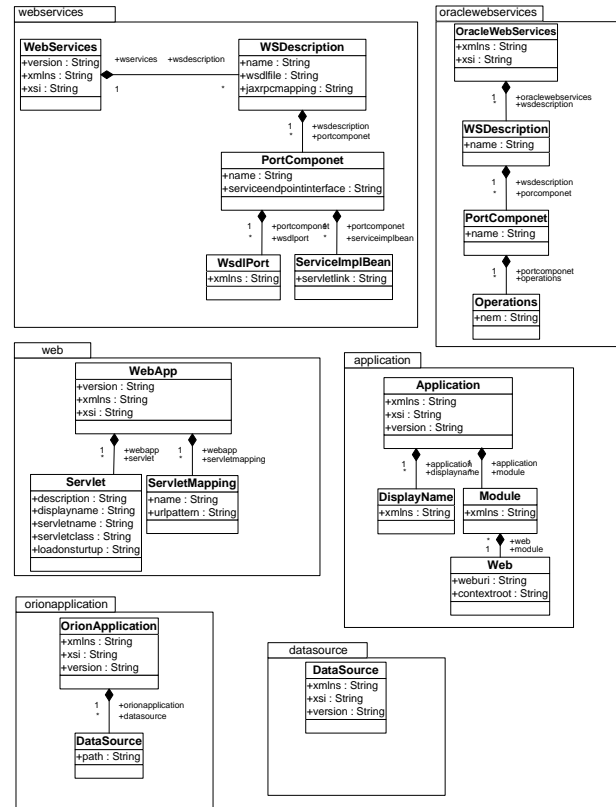


Figure 6 – Oracle Web Services Metamodel (fragment)

The descriptions of some package defined in this metamodel are detailed as follows:

7.1 – The *Web* package is used to generate the file web.xml, that configures the Web service as a servlet.

7.2 – The *Javawsdlmapping* package is used to generate the file javawsdlmapping.xml that mapping the Java interface/methods/parameters to WSDL

7.3 – The *Webservices* package is used to generate the file webservices.xml that defines the Web service endpoint and associated configuration files.

7.4 - The *oracle-webservices* package is used to generate the file oraclewebservices.xml that specific Oracle's Web service runtime

The UML Class is mapped to Web of the configuration file of Oracle's framework through the rule C2Web:

```
rule C2Web{
  from c : UML!Class (c.stereotype ->
    exists(e|e.name='DataTypeService'))
```

```

to webapp : WSORACLE!WebApp(
  xmlns <-
http://java.sun.com/xml/ns/j2ee ,
  xsi <-
'http://www.w3.org/2001/XMLSchema-
instance' ,
  version <- '2.4' ,
  Servlet <- serv
),
serv: WSORACLE!Servlet(
  description <- 'Web Service ' +
c.namespace.name +
'SoapHttpPort' ,
  displayName <- 'Web Service ' +
c.namespace.name +
'SoapHttpPort' ,
  servletname <- c.name +
'SoapHttpPort' ,
  servletclass <- c.namespace.name
+ '.' + c.name ,
  loadstartup <- '1' ,
  ServletMapping <- smpp
),
smpp : WSORACLE!ServletMapping(
  name <- c.namespace.name +
'SoapHttpPort' ,
  urlpattern <-c.namespace.name +
'SoapHttpPort'
)
}

```

These fragments of transformation definitions and XML documents illustrate some steps in the development of our medical information system using a MDA approach and Web Services platform.

8. Conclusion

In our paper, we propose an architecture for the medical information system. In this architecture, the medical data is shared using the Web services. The architecture is developed with MDA. The PIM is created using UML. Initially, here we just focused on the business logic of the medical domain, not concerned about the implementation details. Then, we transformed this PIM to a PSM based on Web Services. For this purpose, we have provided some insights in model transformation.

The MDA is still ascending, and many problems have not been resolved yet. One of the creation challenges lies in the automatic creation of transformation definition between a PIM and a PSM[4].

In future works, we aim extend our proposed medical information system to support the decision making process (like expert systems).

9. References

- [1] Alice Shimada Basic, A Colaborative Enviroment of Computer Aided Medical Diagnostic of High Performance, M.Sc. Dissertation , 2001.
- [2] OMG, Model Driven Architecture, Object Management Group, 2006 Available at: <http://www.omg.org/mda>.
- [3] Natanael E. N. Maia, Ana Paula B. Blois, Cláudia M. Werner, Odyssey-MDA: A Tool for UML Model

Transformations, 20° SBBD and 19°SBES 2005, Uberlândia, Brazil, 2005.

[4] Jean Bezivin, Slimane Hammoudi, Denivaldo Lopes, Jouault Jouault. Applying MDA Approach for Web Service Platform, Enterprise Distributed Object Computing Conference, Eighth IEEE International (EDOC'04), 2004.

[5] Sun. Java Web Services Developer Pack (JWSDP)1.5 Available at <http://java.sun.com/webservices/downloads/1.5/index.html>

[6] W3C. *Web Services Architecture (WSA)*, February 2004. Available at <http://www.w3.org/TR/ws-arch/#whatis>

[7] Oracle, Oracle JDeveloper 10g version 10.1.3.03, Available at <http://www.oracle.com/technology/software/products/jdev/-htdocs/soft1013studio.html>

[8] Ivanova, E., Web service architectures for distributed search in databases, 2nd International IEEE Conference Intelligent Systems, 2004.

[9] N. Al-Rawahi., Y. Baghdadi, Approaches to identify and develop Web services as instance of SOA architecture, Proceedings of ICSSSM '05, International Conference, 13-15 June 2005.

[10] OMG. Meta-Object Facility (MOF™), version 1.4. Available at <http://www.omg.org/technology/documents/formal/mof.htm>.

[11]ATL (Atlas Transformation Language), University of Nantes, 2005. Available at <http://www.sciences.univ-nantes.fr/lina/atl/>

[12] netBeans.org. Metadadta Repository (MDR). Available at <http://mdr.netbeans.org>.

[13] W3C. *Web Services Description Language (WSDL) 1.1*, March 2001. Available at http://www.w3.org/TR/2001/NOTE-wsdl-20010315#_introduction.