Towards an improvement of software development process based on Software Architecture, Model Driven Architecture and Ontologies

Fernando Bartolo Espiritu
Autonomous University of Puebla
Computer Science Department
Puebla, Mexico
esp1@live.com.mx

Abraham Sanchez Lopez
Autonomous University of Puebla
Computer Science Department
Puebla, Mexico
asanchez@cs.buap.mx

Luis Josue Calva Rosales
Autonomous University of Puebla
Computer Science Department
Puebla, Mexico
luis.calva@solarium.cs.buap.mx

Abstract— **Nowadays software development can be done through various methods, the use of a methodology depends on the preference of each development group. One element that emerged to improve the development process is the software architecture, however, the use of this element has been decreasing. This paper considers that the development of software through software architecture is a good practice, so this paper presents a methodology to develop software, which integrates tools such as ontologies and model driven architecture in the development process of software based on software architecture. The aim of this paper is to present a methodology to modernize and improve the development process based on software architecture through the use of tools that exist today.**

Keywords—Software Architecture, Ontology, Model Driven Architecture, UML, Methodology.

I. INTRODUCTION

Software development is a process which involves various opinions and approaches; because of this the software development process has required the specification of methodologies for creating software under uniform standards.

Today, there are various software development methodologies such as RUP, BPM, design patterns and agile methodologies. However, each person uses one of these methodologies in which he can find better results.

One of the current trends in the development of software is that the software is developed automatically; this is accomplished today using CASE (Computer Aided Software Engineering) tools such as Enterprise Architect, Rational IBM, etc.

This paper proposes a software development methodology based on software architecture, which combines the use of various techniques in order to make software development process more efficient and agile. The section two will present a theoretical framework of the technologies used in this methodology. The third section will show how to integrate different technologies to obtain a unified methodology for developing software. The fourth section will show how an application has developed by using the proposed methodology.

II. MODEL DRIVEN ARCHITECTURE, SOFTWARE ARCHITECTURE AND ONTOLOGIES

A. Model Driven Architecture

Model Driven Architecture (MDA) is an approach of software design, proposed and sponsored by the Object Management Group (OMG). MDA is intended to support engineering models directed to software systems. MDA is an architecture that provides a set of guidelines for structuring specifications expressed as models [1].

MDA provides an approach to:

- Specifying a system independently of the support platform.
- Choosing a particular platform for the system.
- Transforming the system specification for a particular platform.

Fig. 1 shows the process of development software with MDA.

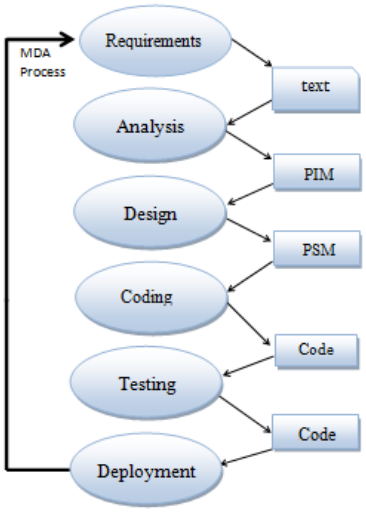

Figure 1. Development software with MDA.

MDA defines three major elements, these are CIM, PIM and PSM, each one of these elements is a stage of development in MDA. The next paragraphs will describe each stage.

- Computation Independent Model (CIM). CIM is a view of the system from the point of view of requirements analysis. CIM does not teach the structure of the systems. At this stage the main objective is the analysis of requirements.
- Platform Independent Model (PIM). It is a system view centered on its operation that hides the necessary details for a particular platform.
- Platform Specific Model (PSM). PSM is the view from the point of view of a specific platform. A PSM combines the specifications in the PIM with the details that specify how the system uses a particular type of platform.

The primary goals of MDA are portability, interoperability and reusability through architectural separation of concepts.

B. *Ontology*

Ontology's definition is not unique, it depends on the context to be used and the use of the author, one of the most accepted definitions in the context of knowledge reuse is that of T. R. Gruber [2] which is defined as an explicit specification of a conceptualization.

The definition of ontology used in this paper is that ontology is a formal method that provides an explicit specification for a shared conceptualization. Conceptualization refers to an abstract model of some phenomenon in the world having identified the relevant concepts of that phenomenon. Explicit means that the type of concept used, and restrictions on their use are explicitly defined. Formal refers to the fact that the ontology should be machine understandable. Shared reflects to the notion that an ontology captures consensual knowledge, which is not typical of any individual if it is accepted by a whole group [3].

According to Jasper [4], Uschold and Gruninger [5] the following are functions of an ontology in software development.

- Communication. Ontologies allow the reduction of conceptual and terminological ambiguity.
- Interoperability. In this sense, ontologies can act as a "translator", which can be used to support the translation between different languages and representations, because it is more efficient to have a translator for each party involved (with an ontology exchange) that the design of a translator for each pair of parties involved (languages or representations).
- Specification. The role of ontologies in the specification depends on the level of formality and fragmentation within the system design methodology. From a casual perspective, ontologies help in the process of identifying the requirements and understanding the relationships between components.
- Reuse. To increase its usefulness, ontology must be able to support the import and export of modules (parts of the ontology).
- Maintenance. One of the main efforts during the maintenance phase of the software system is the study of the system. For this reason, ontologies are used which allow an improvement of the documentation and reduced maintenance costs.

C. *Software Architecture*

The concept of Software Architecture (SA) was conceived in the early 90's of last century, this concept arises from the problems faced by companies and others who developed software, these problems were:

- Projects not finished on time.
- Projects did not fit the initial budget.
- Low Quality Software generated.
- Software that does not meet specifications.
- Code not maintainable that made difficult to manage and develop the project.

The IEEE (Institute of Electrical and Electronics Engineers) defines the SA in the standard "IEEE Std 1471-2000", as follows: "The Software Architecture is the fundamental organization of a system represented in its components, their relationships, the environment and the principles guiding its design and evolution [6]."

This definition is also taken by organizations such as Microsoft, where software development is through SA.

Some of the advantages of developing through a SA according to [7] and [8] are:

- Communication. SA represents a high level of abstraction, which participants can use as a basis for

creating mutual understanding, forming consensus and communicating.

- Constraint constructive. An architectural description provides rules and limits scope for the development of the system, indicating the components and dependencies between them.
- An architecture can be the basis for vocational training. Architecture includes a description of how elements interact to carry out the required conduct, it can serve as an introduction to the system for new project members.
- Evolution. SA can expose the dimensions through which a system can be expected to evolve.
- Management. Experience shows that successful projects considered feasible architecture as a key achievement of the industrial development process.

III. PROPOSAL

The methodology proposed here is to develop software through SA, in addition to using ontologies and MDA for the specification and implementation of the SA.

1) Development software based on SA

Before showing how to integrate MDA and ontologies in software development with SA, firstly is necessary to show the software development process based on an SA, according to [8], the software development process with SA is as shown in the Fig. 2.

Once the needs of a system have been identified and SA has been created for a system, this architecture meets the needs of the system, however, when the SA has been defined, it only shows the components of a system abstractly. To develop the components must be developed views of the architecture, which will serve to be defining the internal structure of the components of the architecture, to get a view that enables the implementation of the system.

A view can be regarded as a phase in which the previous stage is improved, eliminating the abstraction of the previous stage.

Generally, a software architecture allows software development through a process of identifying components and possible relationships among them. Then, each component will be defined in a less abstract component through creating views. When the implementation of the architecture is possible through a view, the process of debugging of the architecture ends.

2) MDA and ontologies in the developmet software based on SA

As shown previously ontology is a tool that allows specifying a domain, ontology has been used for the specification of systems, its ability to limit domains and its contribution to the definition of a system is why the ontology will include in the development process with SA. This combination specifies and defines the abstract architectural elements which can describe elements that can be mapped to UML class diagrams.

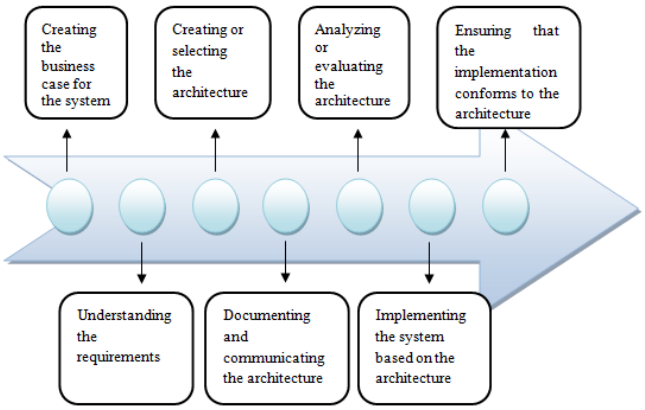

Figure 2. Deveplotment software based on SA

MDA is a software tool to develop software in a generic and fast way, because of this consideration, MDA function is automatically set so views of architecture allowing better define the design and implementation stages of the views. With this process the documentation of architecture improvement and development of the architecture is a robust process.

To involve MDA and Ontology in software development, remember that MDA comprises three stages, CIM, PIM and PSM. The last two stages are defined by MDA, but CIM stage is not set, this stage is free to be specified according to the needs of each development group.

In this proposal, CIM is at the stage where the ontology is used, the decision to use the ontology rests on two important points, the first is that the engineering ontology enables the specification of software systems; second is that both technologies MDA and Ontology are based on the use of meta-data, having with this the possibility to use an Ontology in CIM and then map the model obtained by the ontology to PIM naturally [9].

Until this time has been defined as MDA and Ontology integrate the software development, these tools will serve to improve the development process that is based in the use of an SA. To attach the software development with SA together with MDA and ontologies, the process in the Fig 2 have been distributed as follows.

CIM Stage

- Creating the business case for the system.
- Understanding the requirements.
- Creating or selecting the architecture.
- Documenting, specifying and communicating the architecture (through ontologies).
- Analyzing and evaluating the architecture (through ontologies).

Stages PIM and PSM

- Implementing the system based on the architecture (Mapping ontologies into PIM models, debugging PIM models and getting PSM models)

The final step:

- Ensure that the implementation conforms to the architecture.

CIM is a stage which the system requirements are obtained and analyzed, so according to the process shown in Fig. 2, the CIM stage comprises the first three steps, a developer can choose any existing techniques to develop them.

CIM includes the steps of specifying, documenting, analyzing, evaluating and communicating the SA. These steps will be developed through the ontology. The use of the ontology at this stage will get the first non-abstract internal elements of the architecture, these elements will be the basis to develop and implement the architecture.

Once obtained the first architecture model, this will be debugged with MDA. PIM and PSM stages represent the implementation of the SA; PIM allows any change made in the SA is reflected in the stage PSM and code. PIM and PSM refer to software design, in these stages is when the models obtained in step CIM are implemented.

PIM models define the behavior of each module and to establish the form of interaction of each element of the architecture. PIM stage handles UML class diagrams, these elements will define the structure of the architecture.

PSM is a model specializes in a programming language, this model will be the model of SA to be implemented in a programming language.

In general, this approach is established to improve the software development process based on an SA, the methodology proposed in this paper is shown in the Fig. 3.

Up to this point the methodology has been presented as integrating ontologies and MDA in software development process based on the SA, however, it is necessary to present some intermediate steps that have been defined for the methodology.

3) Procces to create an Ontology

In this paper the process for creating ontologies has been used to specify the architecture, this is because this process provides guidelines for specifying a system. According to [10] the process to create an ontology consists of seven steps, these steps are:

a) Step 1. Determine the domain and scope of the ontology.

This step considers starting the development of an ontology by defining its domain and scope. That is, answer several basic questions:

- What is the domain that the ontology will cover?
- For what we are going to use the ontology?
- For what types of questions the information in the ontology should provide answers?
- Who will use and maintain the ontology?

The answers to these questions may change during the ontology-design process, but at any given time they help limit the scope of the model.

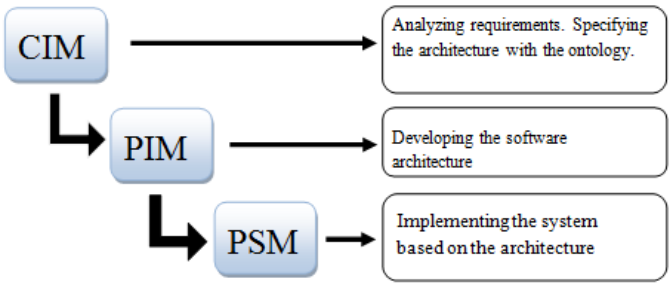

Figure 3 Methodology for developing software with AS using MDA and Ontologies

This step in development software allows defining the scope of a system. This process allows to identify system features and to define certain elements of architecture, this process also allows to remove abstraction to architecture.

b) Step 2. Consider reusing existing ontologies

Reusing existing ontologies may be a requirement if our system needs to interact with other applications that have already committed to particular ontologies or controlled vocabularies.

c) Step 3. Enumerate important terms in the ontology

This step allows defining items that should be in a system considering the requirements analysis. Abstraction in this step is very high, but this process defines general elements of a system.

d) Step 4.Define the classes and the class hierarchy

In this step, the concepts defined above are mapped to classes in the ontology, besides establishing a hierarchy of these classes.

This step in the software development process defines those elements that will form the basis of the architecture, the definition of the hierarchy, in this case it serves to establish the relationship between each of these elements in the functioning of the architecture.

e) Step 5. Define the properties of classes (slots)

The classes alone will not provide enough information to answer the competency questions from Step 1. Once we have defined some of the classes, we must describe the internal structure of concepts.

f) Step 6. Define the facets of the slots

Slots can have different facets describing the type value, allowed values, the number of the values (cardinality), and other features of the values the slot can take.

- Slot cardinality. Defines how many values a slot can have.
- Slot-value type. A value-type facet describes what types of values can fill in the slot. Here is a list of the more common types value in an Ontology:

- String is the simplest type value which is used for slots such as name: the value is a simple string.
- Number (sometimes more specific types value of Float and Integer are used) describes slots with numeric values. For example, a price of a wine can have a float value.
- Boolean slots are simple yes–no flags.
- Enumerated slots specify a list of specific allowed values for the slot.
- Instance-type slots allow definition of relationships between individuals. Slots with type value Instance must also define a list of allowed classes from which the instances can come.

- Domain and range of a slot. Allowed classes for slots of type Instance are often called a range of a slot. The classes to which a slot is attached or a class which property a slot describes, are called the domain of the slot.

These properties (slots) that can be defined in the ontology are sufficient to describe the internal structural architecture, later will see that these properties are equivalent to UML class properties. This property is essential to connect ontologies with elements that are used in MDA, such as UML classes.

g) Step 7. Create instances

The last step is to create individual instances of classes in the hierarchy. Defining an individual instance of a class requires (1) choosing a class, (2) creating an individual instance of that class, and (3) filling in the slot values.

This process gives an idea of how this first structure interacts architecture, making this process also allows a debugging architecture. The creation of an ontology to define the first structure of architecture, this definition is the first view of the architecture.

Once the ontology has been defined, the next step is to use the ontology in the MDA process, to achieve this goal an ontology should be mapped to a UML class diagram, which is used in MDA. The next section will show how to do the mapping of an ontology to UML class diagrams.

4) Mapping Ontology to UML class diagram

Although ontologies and UML use some similar concepts, such as classes and attributes, these concepts do not have the same sense. According to [11] and [12] and the differences between UML ontology are as follows.

a) Classes and individuals

Both UML and ontology have the class concept, but UML refers to an abstract model that describes characteristics (attributes) and behavior (operations) of an object. A UML instance of an object refers to the specification of an abstract object.

On the other hand, the class concept in the ontology refers to a set of individuals representing a specific type. The concept of individual instance refers to a member of a class.

Another difference between object (UML) or instance of ontology, is that the instance concept of ontology can exist without being associated with a class, and may not have the same attributes as the class [13].

b) Properties

Property is another concept that exists in UML as ontologies, and both have different functions from one context to another. In UML a property is a class property, which can never exist alone, must be associated with a class.

An attribute in the ontology can exist by itself, namely an attribute is independent of a class. In the ontology there are two types of attributes "DatatypeProperty" and "ObjectProperty". The difference between both lies in the fact that "DatatypeProperty" are individuals associated with primitive values such as integers, strings, boolean and so on. On the other hand the attribute "ObjectProperty" is associated with individuals who are individuals.

c) Mapping

This section presents the mapping used to transform an ontology model to a model of UML class diagram. Before presenting the mapping used is worth noting that this paper has worked with ontologies thinking that this model would be used in the domain of UML.

The mapping used between Ontology and UML is based on [11], [12], [14] which defined UML profiles for modeling ontologies. With that profiles will be done the mapping of ontologies obtained in CIM.

An ontology class will be taken to a UML class with the stereotype <<OntClass>>, in the ontologies are different class types such as enumeration, union and disjunction. However, these classes will not be used. Avoiding the use of this type of class, the class concept of ontology is similar to UML class concept.

Ontology has two differents properties: simple and complex properties, this paper only use simple properties which are "DatatypeProperty" and "ObjectProperty". "DatatypeProperty" can establish a relationship with the attributes defined in XML Schema, so due to this relationship, these attributes can be mapped as UML attributes.

"ObjectProperty" is an attribute that is handled in the ontology as a relationship between existing classes, and since in this case have not been used complex classes, according to [14] this type of attribute can be mapped to UML as an association relationship between classes.

Thus, the ontologies obtained in CIM are mapped to UML models (class diagrams), to start the development of architecture through MDA.

5) Improving PIM models

Class diagrams obtained from ontology classes define an architecture view, but this view is static, it only defines the elements (classes) and attributes of architecture. However,

necessary to define the behavior of each component of the architecture, for this purpose, UML elements have been used to define the behavior of the architecture.

The elements used to define the behavior of the architecture are sequence diagrams, which help define the methods of each of the UML classes. Through this strategy we have obtained the behavior of classes.

After making a less abstract description of PIM models, a new view of the architecture has also been defined.

6) PSM models

PIM models are transformed into specialized models in a programming language in PSM stage, so class diagrams defined in PIM are transformed into specialized diagrams in a programming language.

To map the PIM models to PSM models CASE tool has been used, these models are elements that can be implemented in code.

However, PSM elements can create a new view of architecture, for this has resorted to using another UML element, package diagrams, they can organize UML classes according to the functions that each class has within the system.

Use package diagram to organize the elements of PSM stage allows a modular design of the architecture, which allows any system be more robust and scalable.

The definition of the intermediate processes is used in the method proposed here. It has been shown how the integration of ontologies, MDA and UML elements in the software development process based on SA, allows agile, robust and well-documented development software.

7) Advantages of the methodology

With the integration of MDA and ontologies in software development based on software architecture, there are advantages such as.

- Better architecture specification. This is accomplished with the ontology, because ontology is a tool to define the architecture when there is only an abstract concept of architecture.
- Better definition of the stages of design and implementation of the architecture. With MDA achieves this goal, because that defining the PIM and PSM models, architecture has two stages of development, first define the architecture without considering a platform , and then given the definition of an independent model can create a specific model, defining well the design and implementation stages .
- Better architecture documentation. By integrating ontologies and MDA, the development of architecture is done through stages, where each stage clearly identifies as development architecture. Each stage provides documentation.
- Automation of the process of development of architecture. With the use of MDA this process becomes more feasible to build PSM models from PIM models.

IV. APPLICATION DEVELOPED WITH THE PROPOSED METHODOLOGY

In this section an application developed through the proposed methodology will be shown, the architecture used is obtained from [15] and shown in Fig. 4.

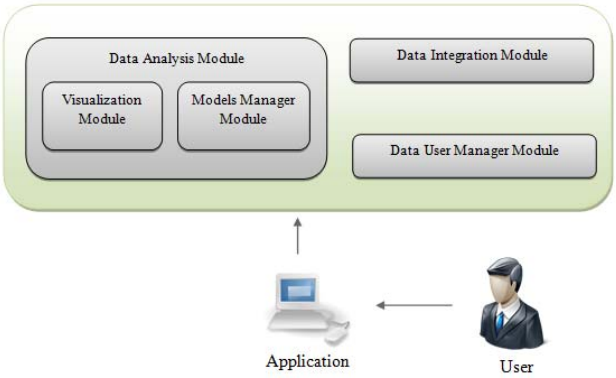

Figure 4 An architecture for a decision support system implemented through web services

The architecture defines a decision support system that is implemented through web services. To show how to use the proposed methodology to develop software, it is displayed as defined an element of architecture with the methodology, this element will be data user manager module.

Since the architecture is implemented through web services, the logic layer web service will be defined with the methodology.

First, the general requirements of this module are defined. The main features that will be covered in this module are:

- Allowing the user access to the system..
- Managing system users.
- Creation and allocation of resources (files and directories) for a user.
- User authentication.

With the description of the requirements of this module, Fig. 5 shows the first view of the architecture of this module.

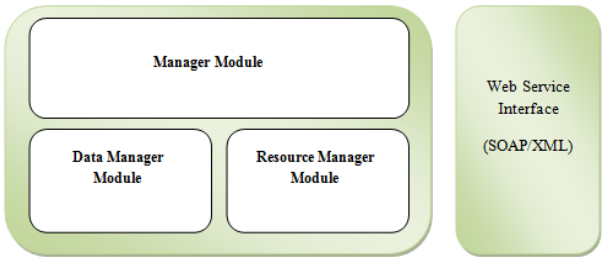

Figure 5 Internal architecture of the data user manager module

After defining the general elements of the module, the internal structure of each component of the module must be defined. To define the internal structure must be used ontology and ontological engineering.

Fig. 6 shows the elements of the user handler module defined through the ontological engineering. Ontology has been implemented with the Protégé tool.

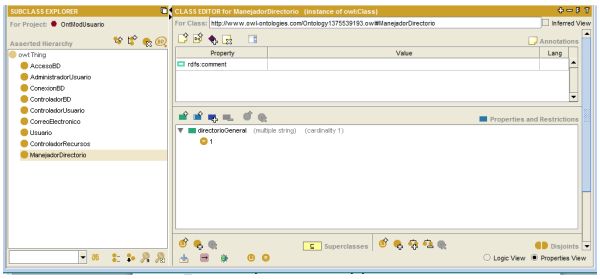

Figure 6 Ontology for data user manager module

In this way, the first structure of the architecture was obtained by the definition of ontological classes and the relationships between classes. In addition, this model is a less abstract view of the architecture, which defines a static behavior of the system.

Once defined ontology, ontological classes are mapped to UML class diagrams. This mapping is shown in Fig. 7, the figure shows that UML classes are mapped with the stereotype "<OntClass>", specifying that these classes come from an ontology.

The relationships between the classes specified in the ontology have been mapped as association relationships between UML classes. Also, the figure shows the mapping of the ontology class attributes to UML class attributes.

It is worth mentioning that process to map ontology classes to UML classes, was made as a manual procedure, since this mapping requires necessary conditions are detected in this process in order to get a class diagram correct.

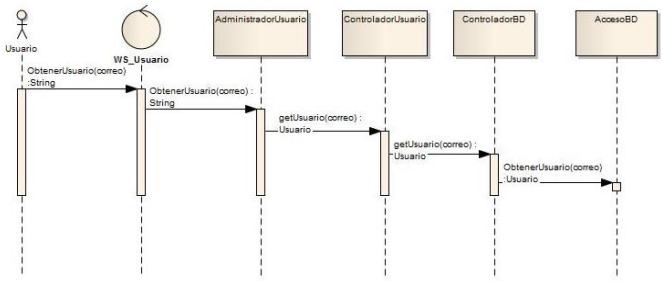

Figure 8 Sequence diagram of the method to obtain an user

The diagram also shows how it has mapped the cardinality of each attribute, adding the navigability of the relationships between classes, this can also be obtained from the ontology. However, in this case a label has been annexed in the diagram which shows the relationship between classes, to make clearly the relationship between classes.

This step is a view of the architecture, besides this view has created a generic model of the architecture which will be used in the PIM stage of MDA.

The next step that follows is to refine the previous model, for this it is necessary to define the behavior of the architecture, defining the methods of the classes to achieve the full performance of this module.

In this paper to define class methods has been used UML sequence diagrams, these diagrams help the definition of the methods showing the flows of operations, in addition, UML sequence diagrams show how the classes interact with each other to perform a task. It should be noted that during the process of defining methods this process also helps to make the inclusion, deletion or debugging of classes or attributes.

Fig. 8 shows a sequence diagram which describes an internal operation of the module, this operation shows how an user is obtained.

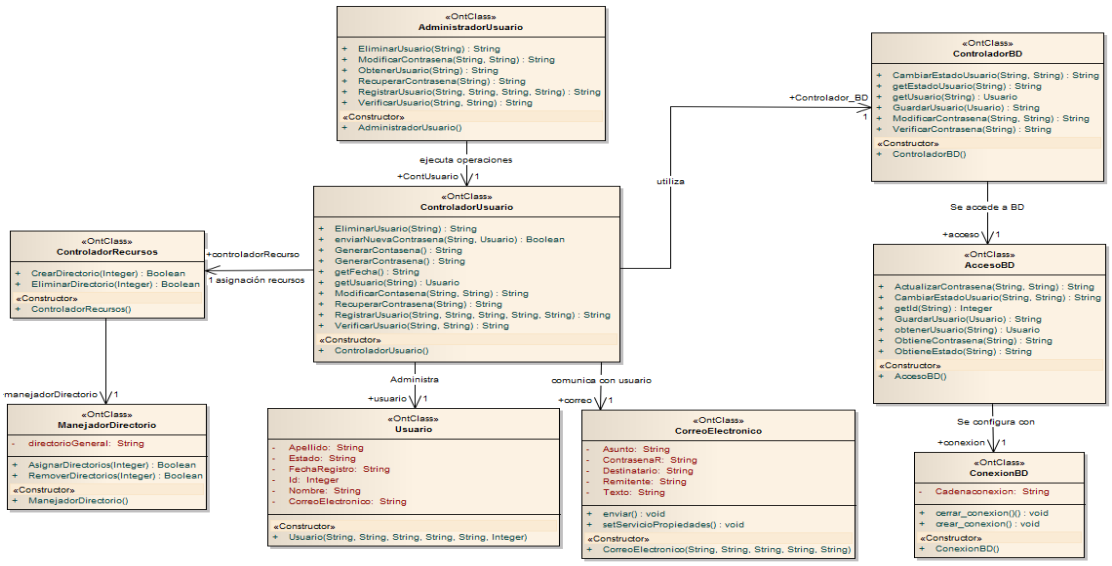

Figure 7 Model PIM of the data user manager module

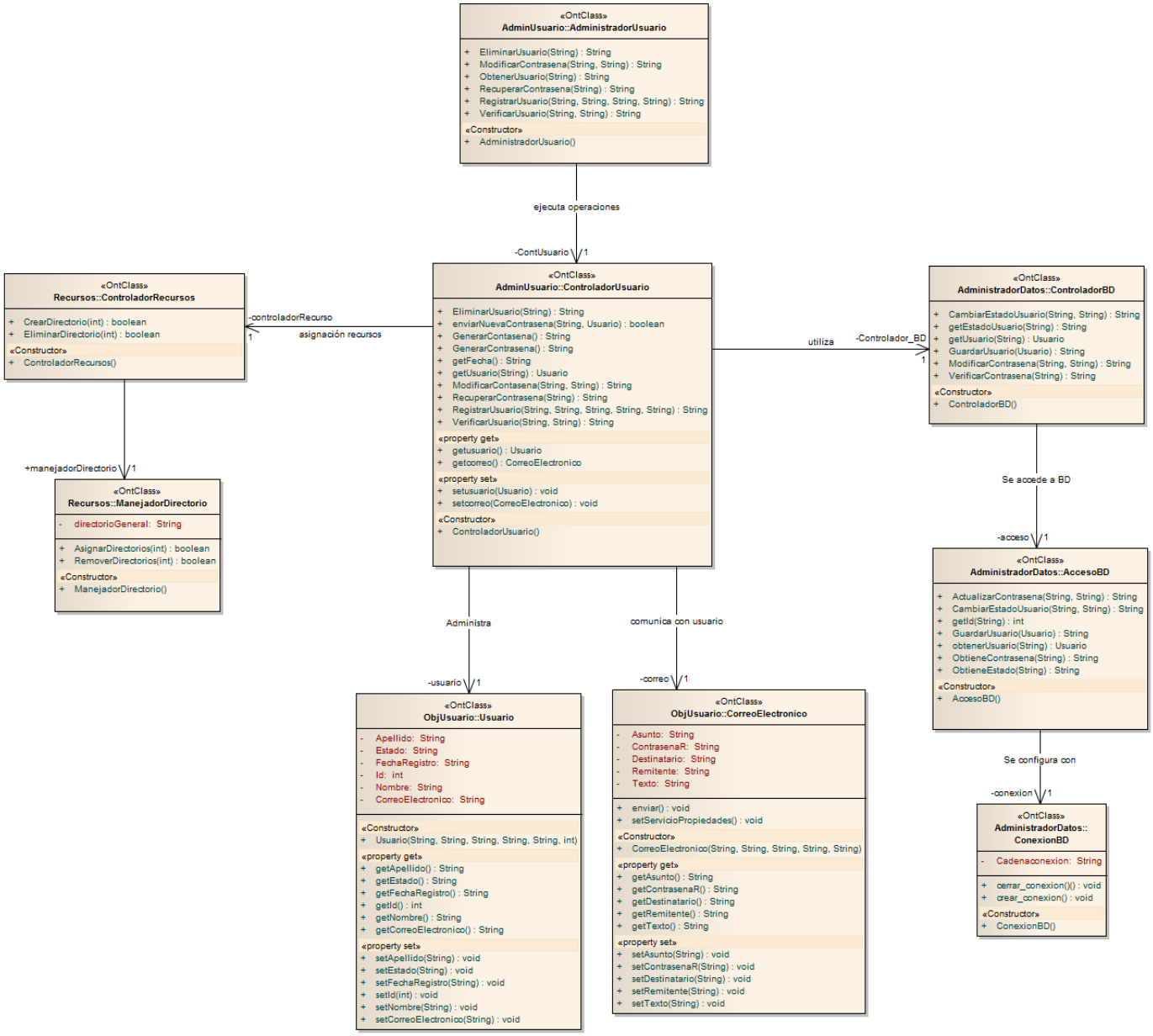

Figure 9 PSM model of the data user manager module

Sequence diagrams are made with each of the operations which are defined to get the requirements of a system.

To complete the PIM stage, the methods defined in the sequence diagrams should be added in class diagrams, Fig. 9 shows the class diagram of user data handler module of the PIM stage.

With the definition of the behavior of the architecture, PSM models should be created, this paper uses a CASE tool to do this mapping, and obtains models related to any programming language.

For the example this paper is managing the programming language used is Java, therefore the class diagram is mapped to a Java class diagram. This class diagram is ready to be implemented in code. Fig. 9 shows the Java class diagram for data manager user module web service.

However, PSM defines other stage architecture view, this view is defined through UML package diagram. UML package diagrams allow to have a modular architecture, with this fact any system implemented in based on the architecture becomes reusable and scalable.

Fig. 10 shows the package diagram which shows the organization of classes of the data user manager module.

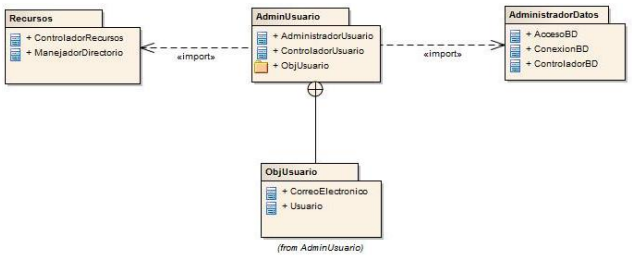

Figure 10 Package diagram for data user manager module

The last step that remains is to map PSM models to code and with this data user manager module is ready to be exposed as a Web service. In general this step is where the development process of architecture has culminated.

Fig. 11 shows how the main class of data user manager module is declared in the class that provides Java to create web services, the main class of data user manager module is responsible for managing the features of the module, with this declaration working example is ready to be exposed as a web service.

This example has shown how it has developed system architecture, showing how they would use the architecture here proposed.

This example has shown how the system architecture has been developed with the methodology here proposed.

```
@WebService(serviceName = "WS_Usuario")
public class WS_Usuario {
    private AdministradorUsuario admin;
```

Figure 11 Web service in Java for data user manager module

V. CONCLUSIONS AND FUTURE WORK

In this work is presented how to integrate ontologies, MDA and UML elements in software development based on software architecture.

The conclusions that have been obtained in this work are:

- This paper shows how ontologies can be used to make the definition and specification of the architecture in the early stages of development.
- Using MDA with software architecture enables automation of the views.
- Using UML elements allows connection between each passage defined by the methodology also allows these steps can be refined, so that at each step improves a view.
- Using MDA, ontologies and UML in the process of developing software based on architecture not only more dynamic development process, but also allows for a well-documented process, which allows scalability, update systems.

As future work to be performed is:

- Developing an automatic intermediate step to map ontologies to UML class diagrams.
- Creating templates that allow mapping PIM models to PSM models which can use elements of the programming language, in order to make the definition of PSM models more specialized in each programming language.
- Integrating the proposed methodology under agile development scheme, to optimize the software development process.

REFERENCES

[1] OMG, "MDA Guide Version 1.0.1". Document Number: omg/2003-06-01.

[2] T. R. Gruber, "A Translation Approach to Portable Ontology Specifications", Knowledge Acquisition, pp. 199-220, 1993.

[3] C. Calero, F. Ruiz, M. Piattini, "Ontologies for Software Engineering and Software Technology" Springer Berlin Heidelberg, 2007.

[4] M. Uschold and R. Jasper, "A Framework for Understanding and Classifying Ontology Applications", Proceedings of IJCAI99 Workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends. CEUR Publications, vol. 18, 1999.

[5] M. Uschold and M. Grüninger, "Ontologies: Principles, Methods and Applications", Knowledge Engineering Review, vol. 11, pp. 93-155, 1996.

[6] IEEE, "Recommended Practice for Architectural Description of Software-Intensive Systems IEEE Standard 1471-2000", 2000.

[7] P. Clements and L. Northrop, *Software architecture: An executive overview*. Technical Report, CMU/SEI-96-TR-003, ESC-TR-96-003, 1996.

[8] L. Bass, P. Clements and R. Kazman, "Software Architecture in Practice", Addison-Wesley, 1998.

[9] D. M., Sánchez; J. M. Cavero and E. Marcos, "Ontologías y MDA: una revisión de la literatura. Ministerio de Ciencia y Tecnología de España" (TIC 2002-12378-E), 2002.

[10] Natalya F. Noy and Deborah L. McGuinness, "Ontology Development 101: A Guide to Creating Your First Ontology" Obtained from http://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html.

[11] G., Dragan, D. Dragan and D. Vladan, "*Model Driven Engineering and Ontology Development*", Springer-Verlag, ISBN 978-3-642-00282-3, 2009.

[12] A. Wafaa and S. Akram, "Ontology Modeling Profile, an Extension for the Ontology UML Profile". International Journal of Computer Applications, vol. 6, 2010.

[13] K. S. Michael, W. Chris and L. M. Deborah, "Owl Web ontology language guide". Obtained from http://www.w3.org/TR/owl-guide/, 2013 .

[14] Kenneth, K. K., Mieczyslaw, B, Paul, A. K, Lewis, H; Jefferey, S; William, S. H; Letkowski, Jerzy; L., Aronson Michael, "Extending the unified modeling language for ontology development", Software and System Modeling (SoSyM),vol 1, pp. 142-156, 2002.

[15] F. Bartolo, "Una arquitectura de software para sistemas de toma de decisiones en la web" (Masters Thesis), Autonomous University of Puebla, Mexico, 2013.