Using MDA in eGovernment

Abstract

Nowadays, most of the governments are in the implementation phase of the eGovernment, where there is a need to offer citizens much more than just a website with information. One of the services that citizens mostly demand, is the possibility to make through the internet all paperwork and proceedings without moving from their place. Developing web applications that permit these services, create certain difficulties like capturing requirements or the quick adaptation to legislation changes. This paper shows an approach based on the paradigm of the Model Driven Architecture (MDA), increasing the level of abstraction and involving users in development.

Key words: Web Engineering, MDA, eGovernment

1 Introduction

In the last few years, one of the major concerns of political solicitors in government civil services, is the strong presence of this services in internet and the need to attend citizens increasing demands of them through the network without having to physically go to the civil services.

In this context, eGovernment is defined in [1] like "the use of Information & Communication Technologies (ICTs) to make public administrations more efficient and effective, promoting growth by cutting red tape".

Web applications for eGovernment are not different from those used in other sectors, but some features may be especially critical as the need for a user-centered design [2], adherence to standards [3], compliance with strict requirements and efficient use of communications.

It's becoming a more common practice for governments to avoid finding themselves with a group of heterogeneous applications which are virtually impossible to maintain, and demanding software development companies to make applications using their own software framework or that applications must be based on web known and documented architectures.

These software frameworks allow you to define the reference of the architecture and the technological platform where applications will be developed, simplify

the development process and define the standards for quality and acceptance that will be required for this development.

These applications replace the traditional processing of an administrative file which otherwise citizens would need to fill in a form, hand it in the according administrative service and finally make the follow-up actions of the file (by phone, personally ...).

In this essay, we will refer to *file* as the whole of administrative proceedings carried out to solve a special issue, which is responsibility of Public Administrations. Each of these administrative proceedings is what is known with the name of *procedure* and is carried out by a public official.

These applications have the following handicaps:

(1) The difficulty to obtain requirements.
(2) The difficulty of quick adaptations when there are changes in Legislation.
(3) The resistance from suppliers to update frameworks.
(4) The high cost of training the staff involved in development.
    The knowledge gained by analysts from finding out solutions in development on a certain framework can not be transferred to another without previously having received training of the new framework.

To minimize or remove these handicaps, we propose using a solution based on the paradigm of MDE (Model Driven Engineering).

The Model Driven Engineering (MDE), proposes to focus software development on models, rather than on code. In this context a good definition of a model is provided by the Object Management Group (OMG): "A model of a system is a description or specification of that system and its environment for some certain purpose. A model is often presented as a combination of drawings and text. The text may be in a modelling language or in a natural language."

From these models, combining two basic aspects [4]: Domain-Specific Modelling Languages (DSML) to formalize the application structure, behaviour, and requirements within particular domain, and transformation engines and generators, to analyze certain aspects of models and then synthesize various types of artifacts, which can range from source code to alternative model representations.

Model Driven Architecture (MDA), is an OMG proposal [5] which seeks to standardize an implementation of MDE. The three primary goals of MDA are portability, interoperability and reusability through architectural separation of concerns.

Software Factories is an alternative to MDA leaded by Microsoft. Both are

opposite proposals, but an objective analysis of these [6] leads to the conclusion that both are not as antagonist as the confrontation of OMG vs. Microsoft could make us suppose.

A Software Factory, is defined as [7] "Is a software product line that provides a production facility for the product family by configuring extensible tools using a software template based on a software schema."

In this paper we present an architecture designed to automatically generate transactional web applications for eGovernment based on the paradigm of Model Driven Engineering (MDE).

The rest of this document is organized as follows: In section 2, we review other related researches. Section 3 presents the architecture. Section 4 is an example of how the system works, and in section 5 we present our conclusions and future work.

2 Related works

In recent years, multitude of related tools led by Model Driven Engineering (MDE) have appeared. Some of these tools automate specific tasks; others are general purpose and only a few, either singled or grouped with each other, are implementations of complete paradigm.

Most of these currently available implementations follow the MDA standard of OMG, but just a few of them allow the edition of the PIM model and its transformation to CIM [8].

None of the tools reviewed, met the requirements raised in this work specially in regard to edition of PIM and in being usable by people without technical knowledge.

The proposal of this work to balance responsibilities between domain experts and computer professionals is also seen in [9].

There are several frameworks for application development specialized in eGovernment and various standards to ensure interoperatibility between heterogeneous systems.

The proposal of Mittal et al. in [10], in addition to defining the architecture of the framework includes a set of tools to develop, deploy and manage complex, integrated, and standards-compliant eGovernance solutions. This proposal is more oriented towards the integration of heterogeneous applications.

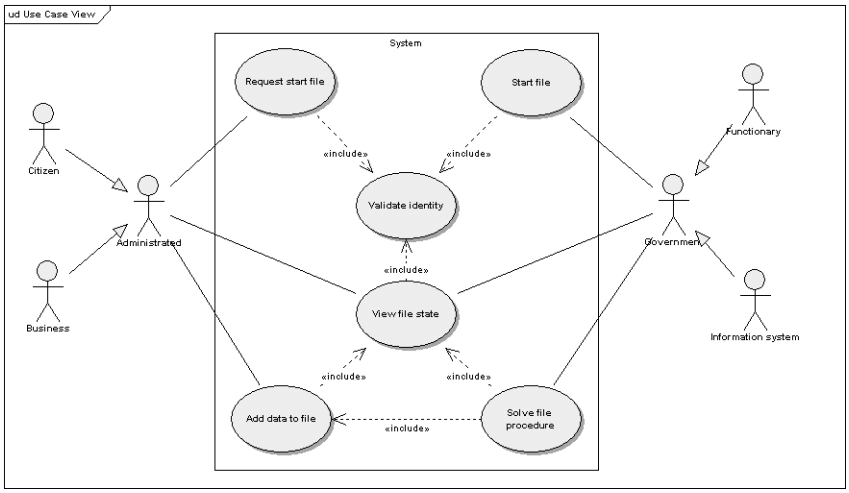

Fig. 1. Use Cases

3 Architecture

3.1 Characteristics

This architecture generates Web applications, which by their nature already have the desired characteristics of platform independence, ubiquity, the absence of installation and high availability [1] .

In addition to these features, this architecture favours specifically generation applications with a user-centered design, standards-based, capable of meeting strict requirements and efficient using of the communication channels. Either way, this architecture only "promotes" compliance with these requirements, which ultimately depends only on how developers use this tool.

The fact that this architecture is based on the paradigm of MDA, makes it enjoy the advantages of raising the level of abstraction in which developers are working. Working with models rather than code, overcomes the limitations that currently have eAdministracin applications that have already been seen in the introduction to this work.

this architecture has a modular design with enough flexibility in order to permit code generation on (for) different platforms. This modular design allows to make good use of the way in which the paradigm of MDA isolates the design (CIM and PIM) of the destination platform. The aim is that the pattern carried out in this architecture will create source code for several platforms by only changing the corresponding module.

[1] 24 hours a day, 7 days a week, 365 days a year

Finally, the user interface is designed to be used both by files supervisors (without needing to have technical knowledge) and by systems analysts.

3.2 Target applications

This architecture is designed to generate applications in any language or platform. It is desirable developing applications based in a institutional software framework or use some kind of framework of general purpose available.

In any way, the only thing really essential is that the target application should be written in an object oriented language capable to generate XHTML pages and use some system of persistence.

The application is organized in a manner that meets at least the use cases that can be seen in figure 1.

As mentioned, it does not impose any architecture to target application, but a reference architecture could be the one shown in figure 2 based on known model 2 for Web applications that use MVC pattern [11].

In it, there are three distinct modules:

- Controller: This module is responsible for monitoring all communication with the outside world and ensures that all external users are properly validated. Listens to customer requests and with the data obtained by the *file* handler generates WebPages with the requested response.
- *File* handler: This module is the business logic which is responsible for managing *files* and *procedures*. Conceptually *files* are contained in a directed graph, in with each node will be a *procedure*.
- Persistence system: This module makes the translation between data structures and management system databases. On one hand it will store the graph of the *file* in a database, and on another it will have access to other information systems to collect data that is necessary to complete some *procedures*.

3.3 Design

In figure 3 we can see the architecture and its relation with users.

There are two profiles of system users clearly differentiated:

- *File* supervisor: This user is responsible for the *file* processing, therefore who better knows its procedure. Interacts directly with the system through

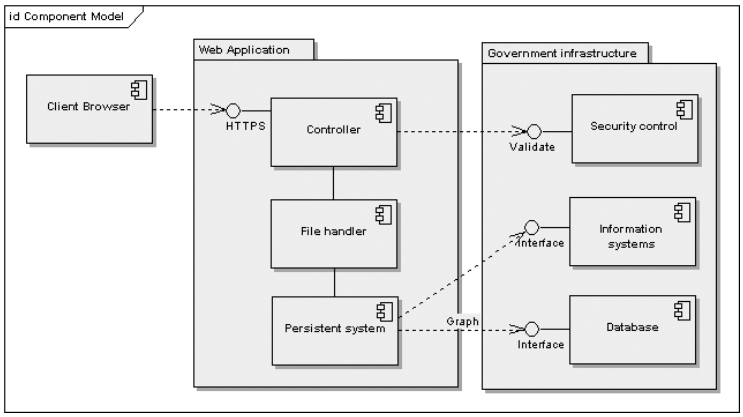

Fig. 2. Component Model

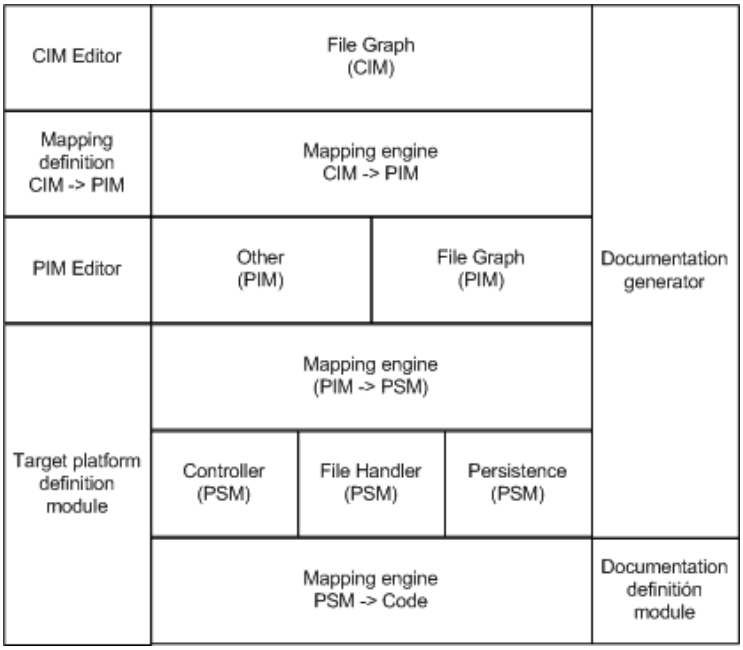

Fig. 3. Architecture

the CIM editor, permitting the modification of the *file*, adding procedures, forks, etc... This user doesn't need to have any technological skills beyond the level of personal computer user.

- System analyst: Is the computer engineer responsible for the application design. This is a person whose training allows to edit directly the PIM to modify aspects associated with technology and make design decisions depending on the requirements that have captured the person responsible for the dossier and the managers of the agency.

With the data provided by these users, the system is able to generate the entire source code of the application, the code needed to perform testing and the documentation required.

Following are described one by one all the modules of the application, indicating their roles within the system:

- CIM editor: Is a graphic editor that specializes in SBPMN notation that allows to handle the graph that defines the processing of a *file*.
- CIM to PIM mapping definitions: It lays down the mapping rules for conversion of CIM to PIM. These mapping rules are necessary to run the mapping engine.
- Mapping engine CIM to PIM: Takes as input the CIM and the mapping rules to generate the PIM of *file* graph.
- PIM editor: The system analyst uses it to edit the following PIM models (some of them already generated by the mapping engine CIM to PIM):
  · *File* handler class diagram.
  · Application engine class diagram.
  · Navigation diagram.
  · Users diagram.
  · Interface specification.
- Mapping engine PIM to PSM: In this module, the combination of PIM models and the transformation rules obtained from the target platform definition module generates the PSM models. The following three models are generated separately:
  · Application engine models.
  · *File* handler models.
  · Persistence system models.
- Mapping engine PSM to source code: In this module, the combination of PSM models and the transformation rules obtained from the Target Platform Definition Module, generates the application source code.
- Target platform definition module: This is the module that contains the definition of mapping rules PIM to PSM and PSM to source code. It contains all specific information the system needs to generate applications for a specific platform.

  If you want to create applications with another architecture or another platform, this module must be replaced.
- Documentation generator: When a public administration ordering an application requires the delivery of documentation that follows strictly any specific methodology (for example: mtrica in the Spanish case). This module collects the needed information from CIM, PIM and PSM to generate requested documentation.
- Documentation definition module: This is the module which is responsible to give format to the provided documentation generated by the previous module.

  We must replace this module if we want to change the format of documentation to hand.

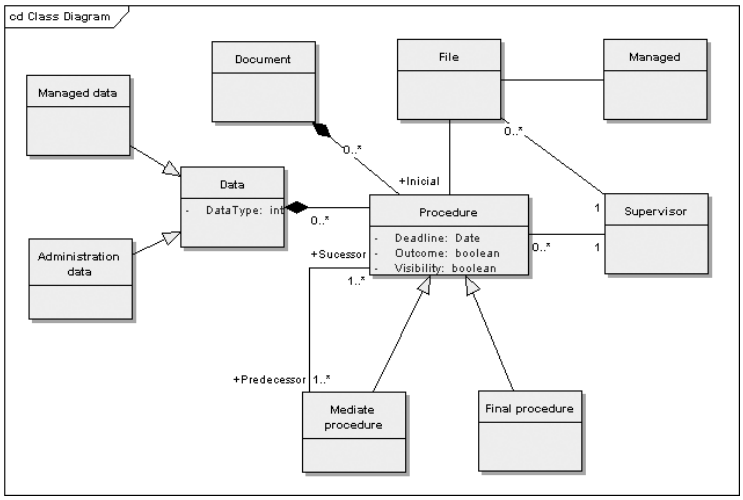

Fig. 4. Class diagram

3.3.1 CIM editing

The core of any MDA is the model of the highest level of abstraction that it is capable of handling. The system allows to describe the business rules in a computer independent model where the graph that describes the processing of a *file* is represented.

Each node in the graph represents the state of the *file* pending resolution of a particular procedure. For each node, the following information must be included:

- Preconditions (Each edge entering the node establishes a procedure that must be solved beforehand.)
- Data supplied by the citizen.
- Data supplied by public administrations.
- Outcome of the procedure.
- Associated documentation.
- Deadline for resolution.
- Visibility of the procedure (If the citizen can see the procedure or not).

3.3.2 Mapping CIM to PIM

CIM to PIM mapping uses the approach described in [12]. ATL [13] language is used to perform a model-model transformation from BPMN to XPDL [14] and other model-text transformation from XPDL to XMI. This XMI archive defines UML 2.0 instances diagram containing graph structure.

The structure of the graph is based on a UML 2.0 class diagram previously defined. In figure 4, a simplified version of it can be seen.

3.3.3 PIM editing

The next level of abstraction is occupied by the platform independent model. This model is edited by the analyst, and specifies the following information:

- Application interfaces with other information systems.
- CSS style sheets that defines the appearance the user interface will have.

In addition, the analyst should add the following information to each node in the description graph of the *file*:

- Interface for which citizens data is acquired.
- Interface for which public administrations data is acquired.
- User interface.

3.3.4 Mapping PIM to PSM and PSM to source code

Mapping definition witch is used with ATL language, forms the core of the definition platform module which allows the system to be used by different technologies.

Being these models stored in XMI, they can be easily edited by people in charge of module definition with easiness to import and to polish the design.

In the definition of this platform module, there is also the definition of three UML profiles necessary to represent PSM: controller profile, handler profile and persistence profile. These profiles are defined for UML 2.0 and XMI 2.1.

3.4 User interface

One of the goals of this architecture is the possibility of being used by people with no technical knowledge. This is possible if the graphical user interface (GUI) will be Eclipse. Currently, this environment is one of the most widespread, open source and platform independent [15].

To implement it, taking advantage of the modular architecture of both the system and Eclipse, the plug-in needed to provide each functionality has been created.

Finally, instead of directly providing plug-ins for users to add to Eclipse, a new completely adapted version using the facilities offered by version 3 as a Rich Client Platform (RCP) [16] has been prepared.

The use of RCP leads to a friendlier environment for *file* supervisors and

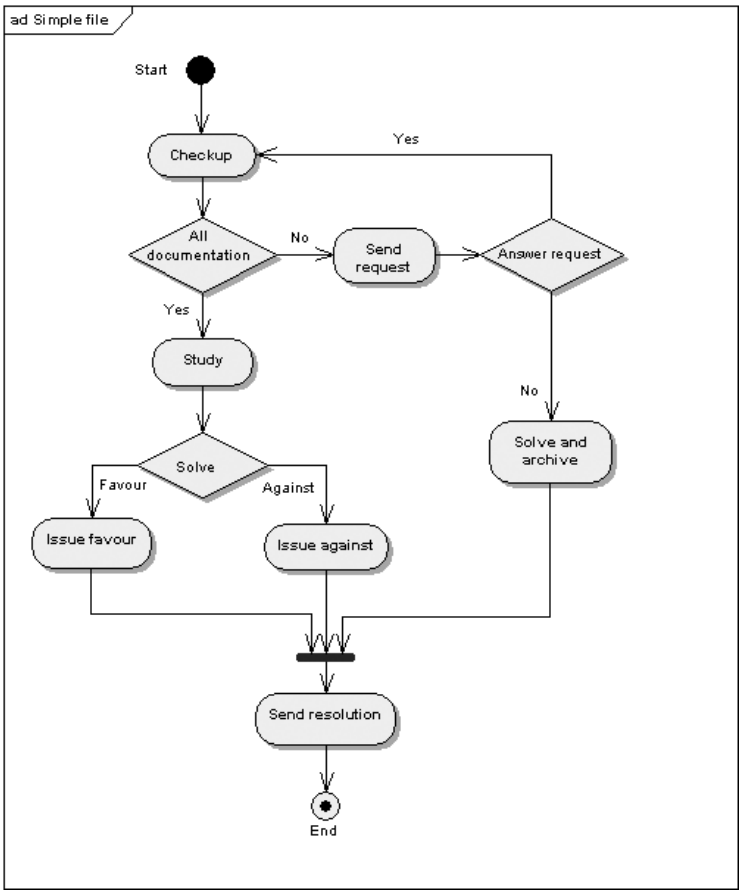

Fig. 5. Sample *file*

the removal of all those Eclipse elements without a clear functionality in the system and that could complicate the platform usability.

4 Case study

The diagram in figure 5 shows the processing of a very simple *file*. In it, the request is examined whether it is correct and if it has all the required documentation. If everything is correct, it is assembled by the officer responsible, and report is issued, either in favour or against. If the required documentation is not complete, this is claimed and once the petitioner answered, the *file* is re-examined. If the petitioner doesn't answer, the *file* process is finished and filed away.

This case study is designed to be used by for the Spanish government, but is extensible to any other.

Although this *file* is an extremely simple example, most are much more complex, with a complicated process and full of forks.

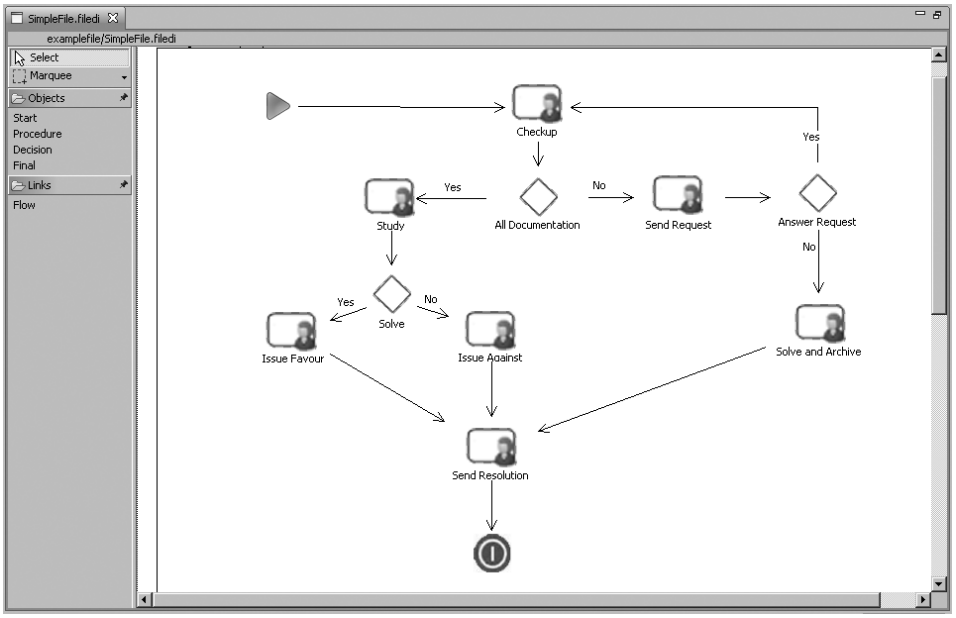

Fig. 6. *File* diagram view in graphic editor

```
                <UML:Class name="File" xmi.id="MDID_w8IxIM37" visibilit
"false" isAbstract="false" isActive="false">
                    <UML:ModelElement.taggedValue>
                </UML:Class>
                <UML:Class name="Managed data" xmi.id="MDID_0hcMYDOh" v
isLeaf="false" isAbstract="false" isActive="false">
                    <UML:ModelElement.taggedValue>
                    <UML:GeneralizableElement.generalization>
                        <UML:Generalization xmi.idref="MDID_Ri6XgzOi"/>
                    </UML:GeneralizableElement.generalization>
                </UML:Class>
                <UML:Class name="Final procedure" xmi.id="MID_ieKs4y4a"
"false" isLeaf="false" isAbstract="false" isActive="false">
                    <UML:ModelElement.taggedValue>
```

Fig. 7. Instance XMI

Based on this *file* example, the system is used to build the corresponding chart in SBPMN notation as we see in Figure 6.

When the system receives the order to transform CIM to PIM, it creates an XMI archive that contains a UML 2.0 instance diagram. In figure 7 you can see this archive and in figure 8 you can see the imported sample instance diagram based on the class diagram shown in figure 4.

This instance diagram is the basis of PIM, which is made up of the following archives:

(1) Interfaces.xml which contains the definition of the possible interfaces the application can present. In this case there is only the Web interface. This archive also specifies the interface that corresponds to each node.
(2) Style.css archive which contains the stylesheet the application will use.
(3) WebPages.xml wich contains a section for each node in the graph. Each section contains a XHTML template of the webpage which publishes the

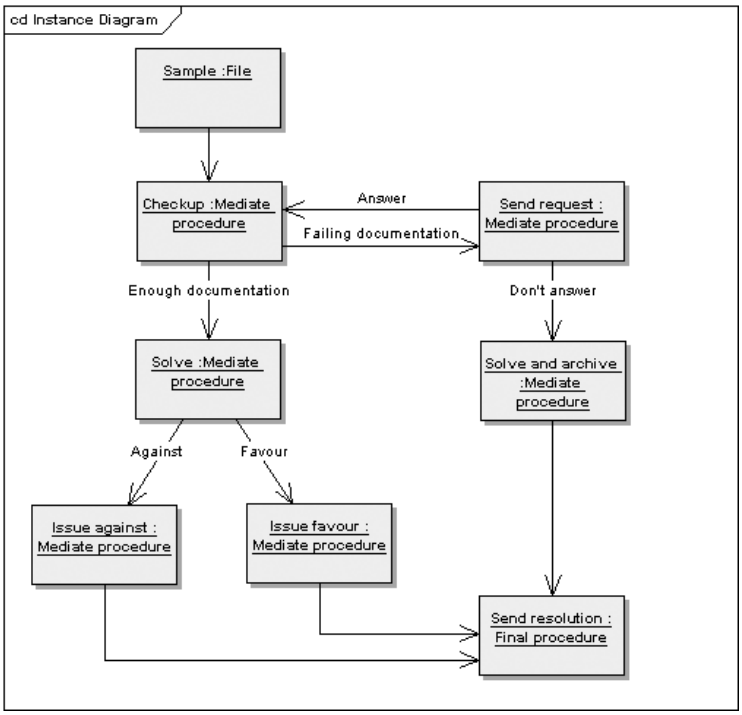

Fig. 8. Instance diagram

node.

The transformation of PIM to PSM, creates three sets of results:

(1) PSM of controller: This is an extended XMI file with the controller profile. In this case J2EE.
(2) PSM of *file* handler: This is an extended XMI file with the controller profile. In this case Java.
(3) PSM of persistence system: This is an extended XMI file with the persistence profile. In this case SQL.

Finally, these PSM's are processed in turn to obtain two sets of results:

- Source code: It contains all the source code of the application. In this case a Eclipse J2EE project for JBoss.
- Text code: It contains all the source code needed to perform unit testing. In this case for JUnit.

5 Conclusions and future work

This architecture represents a progress in various areas and fulfils the objectives that were initially proposed:

- In the **Model Driven Architecture (MDA)** Area, this architecture is a practical implementation of the paradigm which reaches all level of abstraction, including CIM. To allow CIM edition, this architecture uses the SBPMN notation which is actually in definition process.

  This architecture especially takes care of the usability of the tool to allow the developers a close approach to MDA paradigm and greater ease in capture of requirements.
- In the **eGovernment** Area, this architecture provides a specific tool for the peculiarities of the applications requested in this context.

  Having been designed to be directly used by the people responsible for expedient formalities, this tool allows them to take control over computer systems, increasing their confidence on these systems and the diffusion of their work across the network.

  In addition, any changes required by legislation can be directly carried out by the system *file* supervisor, and even thought the collaboration of specialized technicians might be needed to put it online, delays are minimal.

A future task will ensure that portability requirements are fulfilled, preparing a platform definition module for J2EE and another one for .NET. and testing them with identical CIM and PIM models.

The creation of a graphics editor for PIM models to be managed by the system analyst is also in prospect.

References

[1] CE, Culture & society: egovernment — europa - information society, http://ec.europa.eu/information_society/tl/soccul/egov/index_en.htm (May 2008).

[2] J. C. Plaza, "los sistemas de información de la administración electrónica: una oportunidad de mejora desde la experiencia de usuario" (in spanish), http://www.biguel.com/textos/ (May 2008).

[3] J. M. Alonso, egovernment and the web, http://www.w3.org/2008/Talks/0123-eGov-JA (2008).

[4] D. C. Schmidt, Guest editor's introduction: Model-driven engineering, Computer 39 (2) (2006) 25.

[5] J. Mukerji, J. Miller, Mda guide v1.0.1, Tech. Rep. omg/03-06-01, OMG (2003).

[6] J. Munoz, V. Pelechano, "mda vs factorías software" (in spanish), in: II Taller sobre Desarrollo de Software Dirigido por Modelos, MDA y Aplicaciones (DSDM), 2005.

[7] J. Greenfield, K. Short, S. Cook, S. Kent, Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools, Wiley, 2004.

[8] E. Palacios-González, H. Fernández-Fernández, V. García-Díaz, B. C. P. G-Bustelo, J. M. C. Lovelle, A review of intelligent software development tools, (Pending published) (2008).

[9] C. Vassilakis, G. Laskaridis, G. Lepouras, S. Rouvas, P. Georgiadis, A framework for managing the lifecycle of transactional e-government services, Telemat. Inf. 20 (4) (2003) 315–329.

[10] P. A. Mittal, M. Kumar, M. K. Mohania, M. Nair, N. Batra, P. Roy, A. Saronwala, L. Yagnik, A framework for egovernance solutions, IBM J. Res. Dev. 48 (5/6) (2004) 717–733.

[11] T. M. H. Reenskaug, Models-views-controllers, Tech. rep., Xerox PARC (1979).

[12] H. Fernández-Fernández, E. Palacios-González, V. García-Díaz, C. P. G-Bustelo, J. M. C. Lovelle, Design of intelligent business applications based in bpm and mde, (Pending published) (2008).

[13] F. Jouault, I. Kurtev, Transforming models with atl, in: Satellite Events at the MoDELS 2005 Conference, Montego Bay, Jamaica, Vol. 3844 of Lecture Notes in Computer Science, Springer Verlag, Berlin, 2006, pp. 128–138.

[14] WfMC, Process definition interface – xml process definition language, Tech. rep., Workflow Management Coalition (2005).

[15] E. S. Foundation, Eclipse.org home, http://www.eclipse.org/ (May 2008).

[16] B. Daum, Professional Eclipse 3 for Java Developers, Addison-Wesley Professional, Wrox, 2004.