Automated code generation support for BI with MDA TALISMAN

Vicente García-Díaz, Héctor Fernández-Fernández, Elías Palacios-González,
B. Cristina Pelayo G-Bustelo, Óscar Sanjuan-Martínez, Juan Manuel Cueva Lovelle

***Abstract* – Model Driven Engineering (MDE) is gaining ever more strength due to the fact that with MDE the software development can be much more productive and this is the way to go closer to real software industrialization. With MDA TALISMAN, we have succeeded in creating complex software solutions for food traceability adapted to different customers, ready to be deployed. We rely on the approach to MDE most extended at present, MDA (Model-Driven Development) but as we shall see, we also use the main pillars that support the Software Factories, The proposal from Microsoft to MDE. Besides, in this paper we present five cases of success with MDA TALISMAN.**

Keywords: Intelligent, Automatic, MDA, MDD, MDE, Software Factory, Traceability.

I. Introduction

It is well known that Model-Driven Engineering (MDE) is an approach of the Software Engineering that is gaining more strength each day and that tries to achieve the generation of the code of applications either automatically or semi-automatically. MDE is a generic term that refers to paradigms like Software Factories [1], the proposal of Microsoft based on MDE or the Model-Driven Architecture (MDA) [2], the proposal of the Object Management Group (OMG), that is who is giving further impetus to MDE, at least in terms of the number of publications. There are some others approaches and some others architectures [3], but currently they do not have great relevance. We will explain briefly these two concepts now.

The remaining paper is structured as follows: In the following lines we will talk about the principles of MDA and Software Factories. In Section 2 we explain what the food traceability and what the origin of our work are. In section 3 we will explain the architecture of our system, the MDA TALISMAN. Section 4 is related to cases of study in which MDA TALISMAN can generate others systems for food traceability depending on the type of cheese. In section 5 we comment the conclusions and future work, and finally we show the references to other papers.

This work was supported in part by the Oviedo University under Grant IOVD-01. A.

A. *MDA Principles*

The proposal is based mainly on reducing the weight of the implementation and increase the weight that has the modeling of the system, this follows a similar approach to other engineering that are not equal to the computer science. For example the entire world agrees that to building a bridge requires a detailed plane of the bridge, that is, its model.

The use of models has several advantages, increases portability, interoperability and reusability of systems. The idea, is to start with some models of a high level of abstraction (CIM, Computational Independent Model) that pick up the requirements of the system without using a computer language. Later, becomes a transformation lowering the level of abstraction to a computer model but independent of the computer platform used (PIM, Platform Independent Model) which represent solutions at design level for the requirements of the CIM. The PIM can be transformed into one or more models dependent on the computing platform used (PSM, Platform Specific Model) that provides specific models of one or more desired technological solutions. The last transformation is to convert the various PSM to source code, ready to be used or refined before being used.

Finally, we want to comment that the OMG has more defined standards that serve as basis for the definition of MDA. MOF [4], UML [5], XMI [6], OCL [7], and QVT [8]. To know about real application of Model-Driven Architecture, we recommend [9].

In [1] after comparing software engineering with others, such as civil engineering, its authors have identified the following reasons why the traditional software development has problems:

- One-off development.
- Monolithic systems and increasing system complexity.
- Process immaturity.
- Growing demand for software systems.

In order to try to solve these problems Microsoft has created the concept of Software Factory. To explain this concept we have to talk about the four pillars of Software Factories [10] showed in Fig. 1

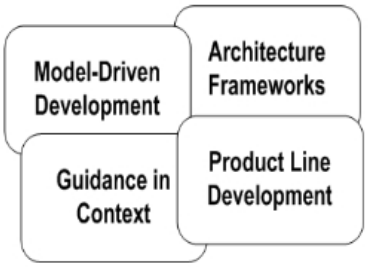

Figure 1. The four pillars of the Software Factories

- Architecture Frameworks. It refers to that we should implement the common features of a system on a basic Framework, which has to provide extension points where components can be integrated and extended.
- Product line Development. It refers to that a product line should only attempt to cover a specific domain or market segment without attempt to cover all the possible domains.
- Model-Driven Development. It is the closest point regarding MDA, Also it is closely related to domain-specific languages (DSL) [11].
- Guidance in Context. It refers to that we should include facilities such as code samples, how-to help pages, articles, and so on.

It should be keep in mind that there are always two perspectives from which we can see the Software Factories, on the one hand is the viewpoint of the developer of the Software Factory (author's view), and on the other hand is the viewpoint of the developer who uses the Software Factory to create software (consumer's view).

The idea is not to create a system whereby we can create all kinds of applications automatically as aspires MDA. The Software Factories are more realistic and although they raise a lot the level of abstraction respect of traditional software, they do not do it like MDA. Authors such as [12] or [13] say that MDA is too pretentious and compare it with the ideas offered by CASE tools that failed in their attempt.

II. FOOD TRACEABILITY

The food traceability is becoming more important in our days and that is the reason why we consider important introduce that term and by the way, what the origin of our work is. To explain what food traceability is we will give a simple example. If you have ever been wondering yourself what was the origin of any food, for instance, the cow that gave the milk that is in your cheese, then you have been wondering about the traceability of this cheese. To tell the truth, on January 1, 2005 (in accordance with article 18 of the European regulation 1782002) [14] all food businesses should have a traceability system but, unfortunately, today the reality is quite different. It can be concluded that food traceability is a need with which to tackle the problems that can give products for the food consumption. It consists of collecting data during all phases of the production process of an article and whether health authorities require it, to have such information. Information is beginning to store with the raw materials used in manufacturing (origin, quantity, supplier, and any other information that may be of interest). Later will be stored intermediate processes that occur in the manufacturing of articles (e.g. dates or temperatures) and finally, who sold the article before arriving at the hands of the end-user (such as an intermediary or a supermarket). The idea is to have fully determined the history of an article. The goal is to have stored information and that can be used by health authorities, but it may also be of interest to the producer to store statistics, ensure product quality or determine responsibilities, and of course the ultimate consumer may be interested in know the origin of what they eat. For that reason might be a good choice to provide consumers with mechanisms to be able to get some kind of information on the articles they consume, as this will increase confidence through transparency. One of the main causes of the introduction of food traceability is that when there is an alert food we must locate and withdraw from the chain supply any product that might be affected in some way, from effectively and efficiently, because so far no one can say that things are not well without food traceability systems but neither anyone can prove that the companies are doing well and of course a failure in a food business for a particular item (e.g. chicken meat) can affect the image and thus in the economic aspect of all other companies who market the same or similar articles.

III. MDA TALISMAN

The MDA TALISMAN [15], our proposal based on MDE, use the approach promoted by the OMG, doing separations at different levels of abstraction. The highest level of abstraction, CIM, is transformed to a PIM manually, and since then the process is automatic, the PIM is refined and becomes another PIM, the PIM becomes a PSM, the PSM is refined and becomes another PSM, and finally the PSM is transformed directly into source code. Although we have adapted it to the real case presented in this paper, its basic architecture is shown in the following figure.

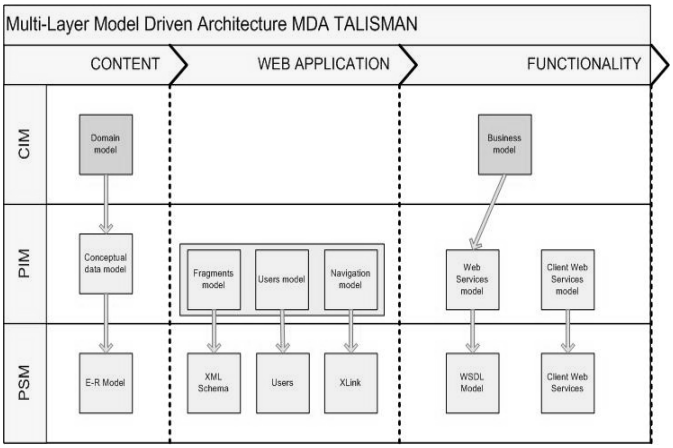

Figure 2. Basic architecture of the MDA TALISMAN

There are three distinct layers, Content, Web Application (we are orientated towards Web applications), and Functionality. Each one of the three layers can be divided into

the three views promoted by MDA (CIM, PIM, and PSM):

- Content. This layer defines the data to create the business layer and the main structure of the database.
- Web Application. This layer is responsible for creating the structure of the pages viewed by the end-user and the relationships between them. It is also responsible for user profiles, and users who can access the cited pages or portions thereof.
- Functionality. Thanks to this layer it is able to offer Web Services to the outside or may use other Web Services supplied from other URLs.

A. *Architectural features*

Initially, MDA TALISMAN was conceived as an architecture for Web applications. After that, it was found its flexibility to adapt to complex software solutions involving Web applications, Windows services and hardware of various kinds, generating software solutions ready to be deployed (Fig. 5). UML, by itself, does not have the power required to generate solutions so sophisticated. There are UML profiles, which can be adapted to the needs of those who used them and continue to rely on MOF as well as UML but the complexity of development increases too much to achieve the level of detail needed to make food traceability software, and that is our goal with MDA TALISMAN. So the solution was to create a basic framework. MDA Talisman focuses on the family of applications for food traceability and the basic framework will have generated all the common features to all applications for food traceability. Then what will be done, is to inject the parties variables code in the basic framework. Our idea, perfectly house with three of the basic pillars of Software Factories that we commented at the beginning of this work:

- Product line Development. MDA TALISMAN focuses on the family for food traceability applications.
- Architecture Frameworks. MDA TALISMAN uses a framework to provide all the common functionality that not vary across products of the production line (across each cheese factory).
- Model-driven development. MDA TALISMAN uses a model for generating all the functionality that changes from one product to another of the production line.

B. *Inputs*

To do their tasks MDA TALISMAN needs a series of artifacts that will discuss below:

1) *eXtensible Process Definition Markup Language (XPDML)*

To achieve our goals we have set up a DSL called XPDML (eXtensible Process Definition Markup Language) which is a subset of XML (eXtensible Markup Language) and that is the heart of our system. The reason for creating a new language is to have a language with the concepts of the domain in order to facilitate ways in which producers adapt the application to their needs. This language has evolved constantly since the beginning of MDA TALISMAN adapting to new needs required by customers (owners of the dairies) that have been added and adjusted progressively over the past year. Currently, to define the variable aspects of a dairy we use a document with eight sections. XPDML (displayed in Figure 3) is in its version 1.0 and has the following:

```
<?xml version="1.0" encoding="utf-8"?>
<XPDML version="0.015" client="Vega de Ario">
  <actions>...
  <items>...
  <devices>...
  <lists>...
  <reports>...
  <labels>...
  <traceabilityPoints>...
  <TAGs>...
</XPDML>
```

Figure 3. XPDML example as XML document

```
<action Id="ACTION_RECEPTION_MILK"
  NumberId="1" Relation="1_1"
  Simple="ITEM_RAW_MILK|ITEM_RECEPTION_MILK"
  Multiple="" Must="" Main="" Info="">
  <inputs>...
  <outputs>...
  <constraints>...
  <devices>...
</action>
```

Figure 4. Action specification in XPDML

- Actions. This section describes in unambiguous manner all actions that will be on development process. One action, as well as its attributes, has to indicate what products will receive as input and what products will appear in the output after running the action. It should be seen as a graph in which a node has entrances and has exits (e.g. milk mixture consists of mixing milk, rennet and salt and as a result we get curd). In addition, to control the various operations (e.g. the range of temperature of the milk that is received at the factory) restrictions can be defined for each. Another interesting thing is the list of hardware devices to intervene in an action (for example when a cheese is packed, it will generate a label with a labeler). In Figure 4 can be seen a snippet of code which defines an action. Other sections are defined using snippets similar to this section
- Items. As important as the actions are the articles, because the actions (nodes) will have articles at the entrance, and articles at the exit (arcs). For instance we will talk about the cheese commercialized. The products have other sub-definable features such as properties (for example, who customer has bought the cheese or on what date), definition of forecasts for production calculating (e.g. how many kilos of cheese will be produced with x litres of milk), definition of locations (e.g. different drying caves where cheeses

can lead), and lastly the hardware devices associated with that article (a scale weighing would be an example). For the above example, it has not been necessary to define forecasts, locations or hardware devices, but it will be necessary for other articles.

- Devices. For the system to work properly it is necessary that the server and other hardware interact. A device could be defined through an Ethernet connection to an IP address and a determined port. In one case the hardware is a terminal (could be another, such as a labeler).
- Lists. We used different lists of items to give functionality to the system. For instance the list of possible designs for a label to be printed, initially, there will be two possible designs, design one and design two, respectively stored in two diffent files. In addition to this list, there will be very different lists with other information such as customers, suppliers, business data, types of milk, and so on.
- Reports. The reports are very important, because collect and display necessary information about the system. There are several types of reports, check list of the status of facilities, cleaning, product description, temperature control, etc.
- Labels. It is necessary that all factories have a labeling system to label their products before selling them to an intermediary or a final customer. To define a label for a client and a for a definite article, we should indicate diverse information, such as label design, the style of the label, the type of bar code, the initial digits of the bar code, or a series of fields which give descriptions.
- TraceabilityPoints. Traceability is essential in the implementation generated by the MDA TALISMAN. In this section we indicate interesting point that we want to register. We will indicate the product for which we want to record information and the property that we want to save (for example, we might want to save the date and time when creating a new batch of cheese).
- Tags. Here are listed all possible TAGs strings of 16 digits that have the chips that are used for identification) associated with an identification number in the database. It will achieve two things, on the one hand be able to work from identifiers, which are much shorter than the TAGs, and on the other hand if it breaks down a chip during a process, may be substituted by another with its identifier.

It is important to realize that the information contained in Figure 3 is the same information that is contained in the graphic representation of the model (an example of graphical representation of the model is shown in Figure 6). Indeed, in addition to the graphic representation of information that can be seen in the figure, there are other important information to be added to the XPDML document that enrich the definition of a particular dairy.

2) Language files

The system that generates MDA TALISMAN is a multilanguage system. For that reason, we need to provide files with the translation into the language or languages desired. Basically there will be a file called Basic.XML containing translations valid for the entire production line, and another called Language.XML that will be specific to the variable part and therefore it will change from one system to another. In addition to the two files described there will be others who depend on the architecture, for instance ASP.NET has files associated with each of their pages in order to change the culture of a system without too much difficulty.

3) Labels

During product development, there will be several points on which may require the printing of a label. Obviously, these labels will be configured with preferences or requirements of each producer and therefore they will be used with the MDA TALISMAN to generate the system for a specific customer.

4) Style Sheet

You can also use a stylesheet line with the preferences of the client to generate Web pages to your liking and not having to make changes afterwards. Its use provides a fast and flexible way to change the whole aspect.

5) Images

There will be pictures such as the logo which will also be used with MDA TALISMAN, returning well, to avoid having to make any changes afterwards. We use around ten images for each client.

C. Outputs

MDA TALISMAN produces a series of artefacts which are listed below:

1) Complete Solution

As a result of the MDA TALISMAN process you will obtain a software solution consisting of seven projects ready to be compiled and deployed (In Figure 5 can be seen the aspect of the solution created automatically by MDA TALISMAN working with Visual Studio).

2) Data Base Script

The output will also contain a file called DataBase.sql, which is a file used to create all the static and dynamic information from the database, that is, both tables and the information necessary to operate the system.

3) Log Files

MDA TALISMAN uses Log4net to create log files that contains information from all processes it has accomplished during his transformations. In addition to MDA TALISMAN, log4net is also used for systems generated with MDA TALISMAN.

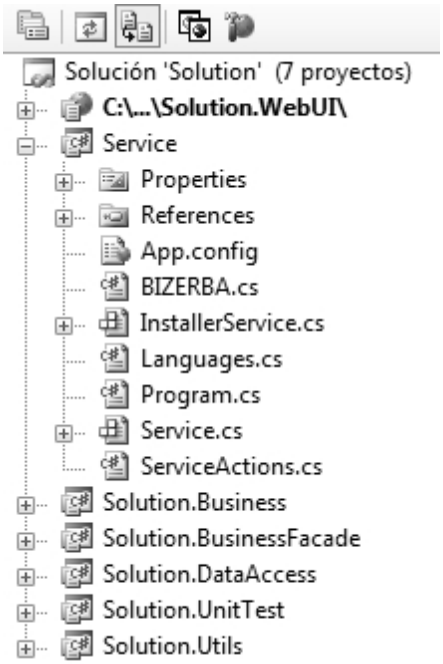

Figure 5. Visual Studio Solution generated by MDA TALISMAN

4) Transformations

The heart of MDA TALISMAN is the XPDML language. With it, there are a number of transformations that lead to the solution previously commented. At this point we try to discuss these changes so that, to make it clear. First of all it should be noted that the XPDML file corresponds to the PIM stage of the MDA philosophy, and that is why the document XPDML is called PIM.XML. The idea is to introduce in this file all the platform independent information and therefore begin the process of transformation. The second step will be to convert the PIM.XML file into PIMtoPIM.XML file since the new model created from the previous is also platform-independent but now the information it contains is no longer XPDML, it is a document in XML format in which UML and UML profiles information are serialized, and thus a part of the system is already under the umbrella of the MDA guidelines. It should be noted that the reason for using XML standard instead of XMI as proposed by the OMG group is none other than the fact that there are several XMI formats, mutually incompatible, and that the tools with which UML diagrams can be created also tend to use their own XMI format incompatible with the other existing tools on the market. In the PIMtoPIM.XML file we can find the following:

- Conceptual data model. It is part of the Content layer and with it are specified aspects needed to achieve the persistence of data.
- Users model. It is part of the Web Application layer and specifies the users and the profiles of the system.
- Fragments model. It is part of the Web Application layer and indicates the different web pages.
- Navigation model. It is part of the Web Application layer and specifies the navigation possibilities on the site.
- Services model. It is part of the Business layer and specifies the Web services offered to other sites.
- Client services model. It is part of the Business layer and specifies Web services for which the system is a customer.

Then, the PIMtoPIM.XML file is transformed into the PSM.XML file that already is a platform dependent model. An example of transformation that can be given in this step is to convert a platform independent data type in a platform specific data type. In our case, .NET would be the target platform and CSharp would be the used language. Thus, for example in PIMtoPIM.XML we have a TEXT data type and in PSM.XML that data type becomes a String. The major transformations taking place in this step are:

- The Conceptual data model becomes the Entity-relationship model.
- The Fragments model becomes XMLSchemas documents, which contain the information to be displayed on Web pages.
- The Navigation model becomes XLink document, which contain information of links between elements.

MDA TALISMAN also takes into account a transformation from PSM.XML to PSMtoPSM.XML, however for the time being done without any processing. It is taken into account because it is one of the MDA steps and we may give it some kind of use in the future. Finally, we need to generate code from the PSM.XML file and put it together with the code already generated with the basic framework. The new code is generated with text templates that has the ExpertCoder library [16] that give a lot of flexibility and avoid having to generate code by hand. For example, the Users model contains information of different users and user profiles that can use the web Application and for this reason generate code is necessary to this model fulfill its purpose. Thus, part of the code will be placed in the Web.config file that is the file where it was introduced security policies for each of the different web pages but it will also be necessary, for example, take into account the different profiles and users to create a script that initializes the database to be used. To appoint another of the transformations taking place we will discuss the Entity-relationship model. From that model, MDA TALISMAN generates the tables in the database that match the model, but also it generates the Business and Business Facade projects with all the necessary files to work with them, using NHibernate for independent access to data.

4 Five cases of real usage

When we gave this work the first thing we did was a survey of dairies that there are different in Asturias (Spain), their common points (from which we get our basic Framework) and points where they differ (from which we get our XPDML).

After the analysis we have done, we realized that despite the relatively small size of factories, how to make cheese is quite different from one company to another. Even companies that manufacture the same type of cheese have many points of disagreement in the manufacture. In addition, MDA

TALISMAN could generate, with no problem, traceability systems for other food industries different from those dairies, since all follow the same basic principles. We want to briefly explain some of the differences between the different cheeses for better understanding the needs of adaptation of software, and so far, the five types of cheese for which we have developed their system are as follows:

- Cabrales. It is the cheese for which we made the first prototype of all. It is a cheese made from milk of sheep, goat or cow (also can be any of the three combinations) that after a maturation period of approximately 60 days in the caves of the Picos de Europe is sold at a price that depends on its size. The treatment is done by units of cheese, so a cheese from the same batch may for instance be inserted on the site where they are going to dry before another. That is why chips are used (they have the size of a currency euro) which identifies each cheese and that are read by RFID reader devices. There are 21 different possible subtypes of cheese.
- Casín. It is a cheese that is produced quite faster than Cabrales cheese. For its production, some different raw material is used to those of Cabrales cheese such as calcium and ferments. It comprises three different types of kneading of products that are getting increasingly close to the cheese. The treatment is done by groups of cheese and to achieve it, an RFID label is printed with a label printer and then placed on different shelves for each batch. All cheeses are sold at the same price and have the same size.
- Afuega'l pitu. It is similar to Casín cheese but have differences such as that in addition to cheese, also is prepared cottage cheese, and that there are several different types of cheese depending on the type of paprika being used. In addition there are boxes (to which we add RFID labels) that are marked in order to move the cheese and not mixing batches.
- Gamoneu. The main difference between this cheese and Cabrales cheese is that the production of the dairies of Gamoneu is much greater in number than those of Cabrales, hence, it will be necessary to bring an organization using batch of cheeses instead of individual cheeses but at the time of packaging is necessary weight each of the cheeses. There are 6 possible subtypes of different cheeses. The type of chips is similar to Cabrales.
- Beyos. The Beyos cheese can be seen as a mixture of Afuega'l pitu cheese and Casín cheese with some changes in the various processes. Some RFID tags are printed in one of the steps that are placed on shelves where the cheese is placed during different stages of their manufacture.

Basically all of them need specific software generated by MDA TALISMAN, SQL server data base, a computer, a scale, a printer, a label printer, RFID readers, industrial terminals, and lots of chips. But depending on the mode of manufacturing the number of items may change, or for example the client could require various other devices such as RFID printers or RFID labels. What never changes is that our software is the one who controls and manages all mentioned hardware.

To better understand the process, Figure 6 shows the graph that we created after some visits to a dairy of Afuega'l pitu. This graph shows the most important aspects (in the absence of many details) of the process of development and is taken as the main base for the generation of XPDML file which in turn serves for the generation of the specific traceability system. Then, one of the things that MDA TALISMAN automatically generated from the entry into XPDML format, is a graph with all the different actions (processes) and items (elements) that there is throughout the whole process of preparing the cheese. That is a SVG format file that contains the graph generated from the input of Afuega'l pitu.

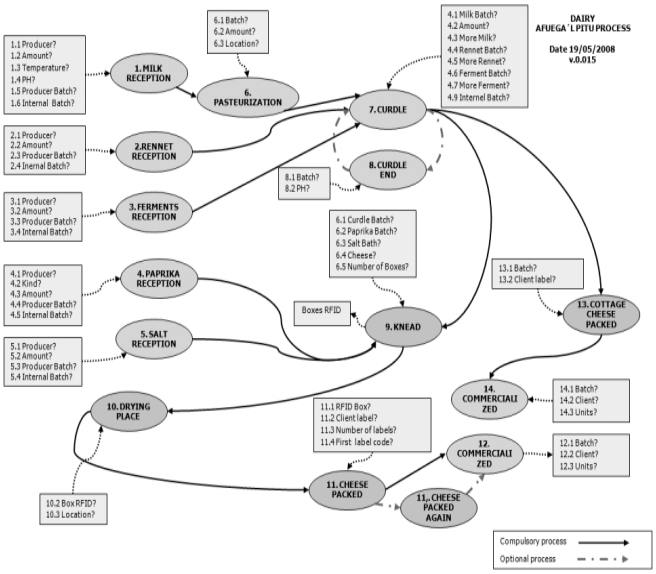

Figure 6. Graphic representation of the model of Afuega'l pitu cheese

This graph is very useful because people can quickly understand all the steps that have the manufacturing process and most importantly of all is that any changes made in the XPDML file is reflected in the graph, changing automatically the parts of the graph as may be necessary. In addition, the image is modified at runtime so that depending on the user profile that is being used in the application at any given time, will activate links to undertake some or other actions (to be able to carry out the action straight from the graph).

IV. Conclusions and Future Work

Our MDA TALISMAN serves to generate applications automatically following the steps of the MDA specification. What happens is that to generate complete software solutions such complex and heterogeneous, ready to deploy in an environment, is today impossible, then we used a framework to inject the code generated through MDA transformations.

This injection of code in a framework leads our MDA TALISMAN towards the ideas promulgated by defenders of the Software Factories. This fact makes that the target of our next job is being to analyze the pros and cons of MDA and the Software Factories to see how we can improve MDA TALISMAN to be capable of generating all kinds of software solutions using the best MDE practices. Also, it is clear that our MDA TALISMAN begins its process with a PIM (A representation of the manufacturing process of each individual case). For this reason we are developing a method thanks to which we will automatically go from CIM to PIM and this would greatly facilitate the task, because people outside the world of computers could generate their own applications for food traceability from their specific requirements independent of computing (CIM) easily. More at [17]

Finally, this work has been done with the contract FUO-EM-120-07 "Software Development for the realization of traceability", working together the University of Oviedo and the company LINK

REFERENCES

[1] J. Greenfield., K. Short, S. Cook, S. Kent, J. Crupi "Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools". Wiley Computer Publishing, Sep 2004

[15] Object Management Group (OMG). "MDA Guide. V1.0.1"; http://www.omg.org/docs/omg/03-06-01.pdf, Jun 2003

[16] Leist. S, Zellner. G "Evaluation of current architecture framework". Proceeding of the 2006 ACM symposium on Applied computing, 2006.

[17] Object Management Group (OMG). "Meta Object Facility (MOF) Core Specification. v2.0"; http://www.omg.org/docs/formal/06-01-01.pdf, Jan 2006

[18] Object Management Group (OMG). "OMG Unified Modelling Language Infrastructure. v2.1.1"; http://www.omg.org/docs/formal/07-11-04.pdf, Nov 2007

[19] Object Management Group (OMG). "MOF 2.0 XMI Mapping. v2.1.1.pdf"; http://www.omg.org/docs/formal/07-12-01.pdf, Dec 2007

[20] Object Management Group (OMG). "Object Constraint Language. v2.0"; http://www.omg.org/docs/formal/06-05-01.pdf, May 2006

[21] Object Management Group (OMG). "Meta Object Facility (MOF) 2.0 Query View Transformation Specification"; http://www.omg.org/docs/ptc/07-07-07.pdf, Jul 2007

[22] M. Guttman, J. Parodi "Real-Life MDA: Solving Business Problems with Model Driven Architecture". Morgan Kaufmann, 2007.

[23] G. Lenz, R. Wienands "Practical Software Factories in .NET". Apress, 2006

[24] S. Cook, G. Jones, S. Kent, A. Wills "Domain-Specific Development with Visual Studio DSL Tools". Addison-Wesley, 2007

[25] D. Thomas "MDA: Revenge of the modelers or UML utopia?". IEEE Software, 21, 2004

[26] S. Cook "Domain-specific modelling and model driven architecture". MDA Journal, 2004

[27] The European Parliament and the Council of the European Union. "REGULATION (EC) No 178/2002 OF THE EUROPEAN PARLAMENT AND OF THE COUNCIL";

[28] http://eurlex.europa.eu/pri/en/oj/dat/2002/l_031/l_03120020201en00010024.pdf

[29] C. G-Bustelo. "TALISMAN: Desarrollo Ágil de Software con Arquitctura Dirigida por modelos". PhdTesis, University of Oviedo, 2007

[30] Expertcoder. http://expertcoder.sourceforge.net

[31] H. Hernández-Fernández, V. García-Díaz, E. Palacios-González, C. G-Bustelo, J. Cueva Lovelle "Design of intelligent business applications based on BPM and MDE". The 2008 World Congress in Computer Science, 2008

Vicente García Díaz is a Computer Engineer, Ph.D student in the Department of Computer Science at the University of Oviedo, Asturias (Spain). He has a Project Management Cerfication by Project Management Institute. He is Superior Technician in Occupational Hazard Prevention. His research interests include model-driven engineering, domain-specific languages, project risk management, software development processes and practices. He can be reached at garciavicente@uniovi.es

Dr. B. Cristina Pelayo García-Bustelo: She is Computer Engineer and Ph. D. by University of Oviedo, Lecturer in the Computer Science Department of the University of Oviedo. She has developed her doctoral thesis in model driven development (MDD), with various publications and business projects based on this thesis. Her research interest include Modeling Software with MDA, BPM, DSL, Object-Oriented technology, Web Engineering, e-Government, Design Patterns, Semantic Web and Web 2.0. She's author of books, articles and conference papers.

Dr. Oscar San Juan Martinez: He is Ph.D. from the Pontifical University of Salamanca in Computer Engineering. His research interest includes Object-Oriented technology, Web Engineering, Software Agents, Modeling Software with BPM, DSL and MDA. Electronics HNC from the University of Wales, graduate in International Studies from the SEI. He has translated and contributed to the review of multiple reference books in the field of Software Engineering. He has published over 80 articles in journals and national conferences and international prestige. In his teaching, he has taught over 30 seminars and conferences in Europe and Latin America on Intelligent Agents, Evolutionary systems, Biologically-inspired computing, Interactive Software, Multimedia, Games and Virtual Reality. In his business, he was Director of the Office of R & D of the Pontifical University of Salamanca, Coordinator of the Master in developing video game and entertainment software and technology consultant for "Vector Information Technologies" Spanish company dedicated to developing highly skilled computer projects in the area of Internet and Mobile Systems, Ubiquity and GIS. Currently works at the University of Oviedo where he developed his research in the field of Model Driven Development, intelligent systems, bio-inspired and interactive, accessibility, emerging systems and the future of the Internet (Smart-Objects and Objects-Net).

Professor Juan Manuel Cueva Lovelle: He is Ph.D. from the Polytechnic University of Madrid in Mining Engineering, Professor in the Computer Science Department of the University of Oviedo (Spain). Has knowledge of metallurgical processes. He is voting member of ACM and IEEE. His research areas include Model Driven Development, Object-Oriented Technology, Language Processors, Human-Computer Interaction and Web Engineering. He conducted several research projects and PhD in Computer Engineering. He is author of books, articles and conference papers, coordinates the research laboratory of Technologies of Object-Oriented Computing department at the University of Oviedo (OOTLab).