MDA FRW: Mixing the principles of MDA and Software Factories

Abstract

Within the Software Engineering every day is gaining more strength the Model-Driven Engineering (MDE). In this paper we present a real case of application of Model-Driven Engineering (also called Model-Driven Development) in which we mix the principles of the two most important paradigms that there are nowadays. On one side is MDA, proposed by the OMG group, and on the other side are the Software Factories, proposed by Microsoft. Both paradigms have their pros and cons and we want to select the best from each of the two. Our real case focuses on controlling food traceability in five dairies with five different manufacturing processes but they have software developed specifically for each dairy without additional programming, only working with models.

Key words: MDA, MDD, DSL, Web Engineering, Traceability, Model

1 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 different paradigms like Software Factories (1) (proposal of Microsoft based on MDE) or Model-Driven Architecture (MDA) (2), (3), and (4) (proposal of the Object Management Group, also called 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 (5), 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 deeply the architecture of our system, the MDA FRW. Section 4 is related to cases of study in which MDA FRW 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 works.

1.1 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 same, that is, their model).

The use of models has several advantages, increases portability, interoperability and reusability of systems. Thanks to models, we have built a platform based on MDA (explained in the next section) that can transform models into aspecific platform output (by the moment we focus on .NET). The idea, is to start with some models (all models must be based on a formal meta-model, in such a way as to allow specify all elements of the system and make transformations among different 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 (their number depends on the number of platforms) to source code, ready to be used or refined before being used.

Finally, comment that OMG has more defined standards that serve as basis for the definition of MDA (MOF (6), UML (7), XMI (8), OCL (9), and QVT (10)). To know about real application of Model-Driven Arhitecture, we recommend (11)

1.2 Software Factories Principles

In the book (1) (although they had already written the article (12)), 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

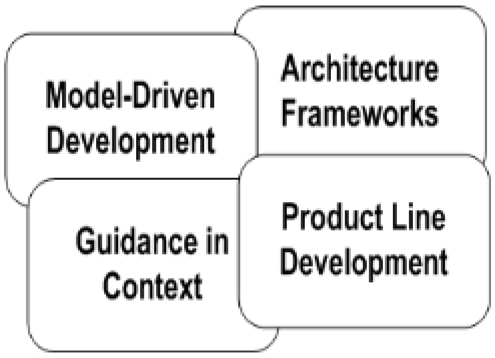


Fig. 1. The four pillars of the Software Factories

- Monolithic systems and increasing systems complexity
- Process immaturity
- Rapidly groling demand for software systems

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 (13) showed in 1.

- 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) (14).
- 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 (authors such as (15), (16), or (17) say that MDA is too pretentious and compare it with the ideas offered by CASE tools that failed in their attempt) and although they raise a lot the level of abstraction respect of traditional software, they do not do it like MDA.

2 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.

2.1 Traceability Problem

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) (18) 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 (one of the most famous cases could be the mad cow disease, although it could be in a much smaller scale) 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 can anyone 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.

2.2 Traceability of different types of cheese in Asturias (Spain)

In the early beginnings we only worked with a type of cheese, the Cabrales cheese, but in Asturias, a region of Spain, there are about seventy types of different cheese, each one with a different elaboration process.

To ensure the quality of food products in Asturias is essential to have a traceability of food manufactured. That is why institutions such as IDEPA (Institute for Economic Development of Asturias) or LILA (Laboratory Interprofessional Dairy Asturian) are putting every effort to ensure that products Asturian comply with the standards of traceability and thus ensure the quality of the products sold. Thus, by using the example of Cabrales cheese, the most famous of all cheeses in Asturias, there are other specific institutions such as the Board of the Designation of Origin "Cabrales" which ensures the quality, authenticity and enforcing standards, performing tasks such as inspect cows, sheep or goats in the area, inspect cheese factories and maturation caves in the area, controlling milk and cheese produced, advise and collaborate with the producers of cheese, and promote Cabrales cheese. The Cabrales cheese has Denomination of Origin since 1981 at the "Instituto de Denominaciones de Origen" (INDO) and complies with the European standard (EN-45011) on product certification agencies. So to preserve its Designation of Origin and ensure the quality of cheese produced, is necessary adapt to the current rules and implement a traceability system for all Cabrales cheese factories.

3 Architecture

The MDA FRW, our proposal based on MDD, uses primarily the approach promoted by the OMG, doing separations at different levels of abstraction, but actually, we do a mixture between the approaches of the OMG and Microsoft with the sole purpose of benefiting from the strengths of both. In the Figure 2 can be seen a general outline of MDA FRW.

3.1 Half Software Factory

First let's explain why we say we use the principles of Software Factories justifying each of its four main pillars.

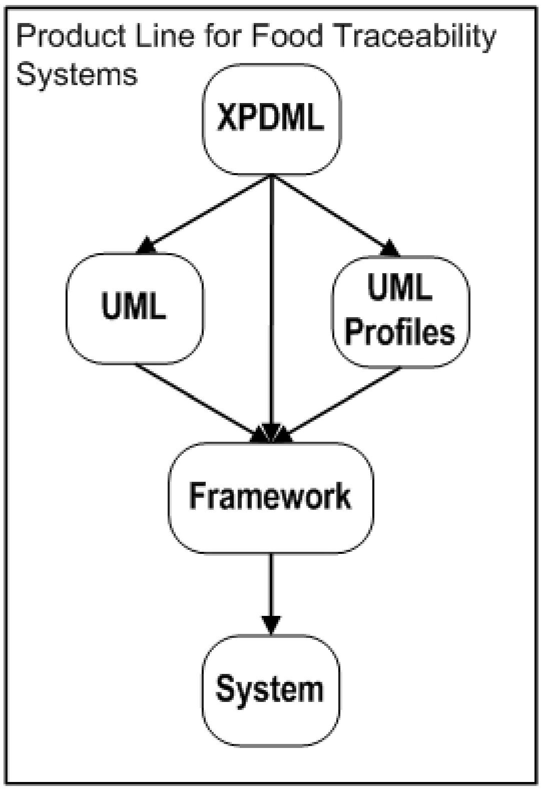


Fig. 2. Global architecture of MDA FRW

As previously mentioned one of its cornerstones is the use of a framework of development based on the common features that are implemented using best practices for development. Thus, we implement basic functions of food traceability systems, which otherwise would be impractical using only UML, especially thinking of complex systems in which more than one application are running at the same time and in interacting computers, industrial terminals, scales, RFID [1] readers, printers and labels with an enormous amount of source code. With our framework we provide a number of features common to all applications to develop, for instance:

- Management and configuration via Web.
- Activities via Web and industrial terminals.
- Access control of users with different user profiles (initially only admin, manager, supervisor, or operator).
- Ability to see a map of the entire process of the factory.
- Ability to see a map of all locations of the factory.
- Ability to see a traceability by batch number of products, or serial number of an individual product.
- Traceability forwards (to customers) and backward (to raw materials).
- Ability to display a full report through Crystal Report technology in order

[1] Radio Frequency Identification. You can learn more at http://en.wikipedia.org/wiki/RFID

to export it to a wide variety of formats.

- Ability to create different types of reports to facilitate management activities.
- Possibility to print labels for help in the development process with descriptive information, and to present the final product to the public, as well.
- Ability to completely manage the labels for each customer and product.
- Ability to view and edit all actions.
- Ability to view and edit all products that have been used.
- Possibility of making forecasts to calculate what production will be, and when you will get the production from a given raw material.
- Opportunity to create restrictions, to prevent certain conditions are met (not be permitted, for example, milk with 2 degrees Celsius)
- Ability to add different types of hardware
- Possibility to work in parallel with the hardware (for example, we could work with various terminals and printers at the same time without any problem)

Another pillar is Product line Development, and we have that very clear, as our specific domain of application is food traceability and to justify that, we only have to read the following quotation from (19)

> "A Software Product line is a set of software-intensive systems sharing a common, managed set of features that satisfy the needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way."

And, of course, we take into account the three most important concepts underlying the software lines. These are, scope (the solution scope, that is to say, developing traceability systems with the features previously appointed), variability (optional features used only in some solutions, for instance only some customers want to do calculating production forecasts), and expandability (we provide extension points in the solution that can be used to add, modify or remove features through the solution that can be opened with a development environment).

Regarding the Model-Driven Development, to make rapid and efficient developments we have created our own DSL, called XPDML (explained later), which allows you to specify variables aspects of the products of the production line and getting two very important things, on the one hand it increases the level of abstraction at which the solution is described and by the other hand it bridges the gap between design and coding, as code generation allows for direct transformation of the models into executable code thanks to MDA FRW.

Finally, the fourth pillar of software factories, the guidance in context. At the

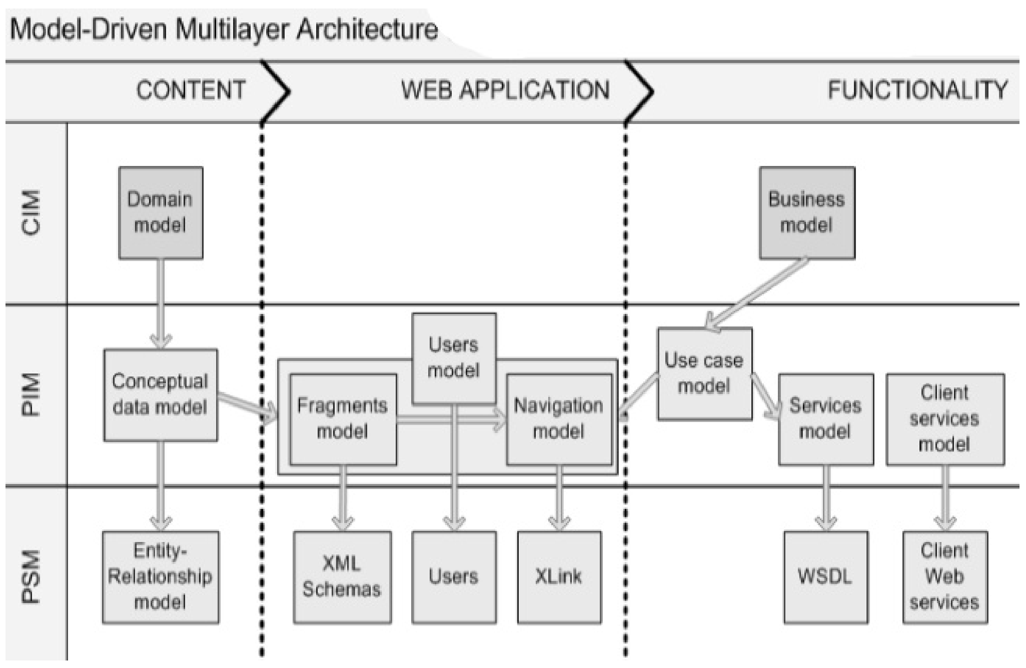


Fig. 3. MDA architecture of MDA FRW

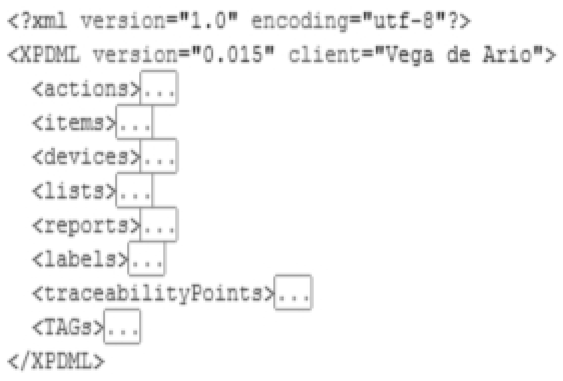
moment what we offer is an extensive user manual and examples with our 5 dairies already in operation.

We believe that these four pillars provide the foundation to make highly complex developments, quickly and with quality, which otherwise would take much longer and would be much more prone to errors. This is a huge step toward the industrialization of software development but like everything, also has its points to improve, for example dependence with a concrete platform.

3.2 Half MDA

But, in addition to Software Factories, we could not missed the opportunity to use the philosophy of MDA and this way get the benefits that provide their use, ie, increases portability, interoperability and reusability of systems. All these things are hard to get and the basic reason is that as many authors have said and we said before, MDA has a very high level of abstraction, but anyway, the promises of MDA are also very interesting and the use of a de facto standard as UML, promulgated by OMG, which makes the code that can be generated from UML metamodel or any other metamodel instance of the metametamodel MOF completely independent of the platform.

In short, MDA FRW uses the principles of Software Factories to provide a model driven development capable of automatically developing applications of a specific production line, but at the same time using the principles of



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

Fig. 4. XPDML example as XML document

MDA to avoid dependence on a specific platform and allowing the parts of the system that can change, can be generated from UML or if not, from UML profiles.

The MDA architecture of MDA FRW are displayed in Figure 3. The idea is to use standards that lead to somehow alleviate the weakness of the Software Factories regard to MDA that is its lowest portability. To achieve this we use a multilayer architecture divided into three layers:

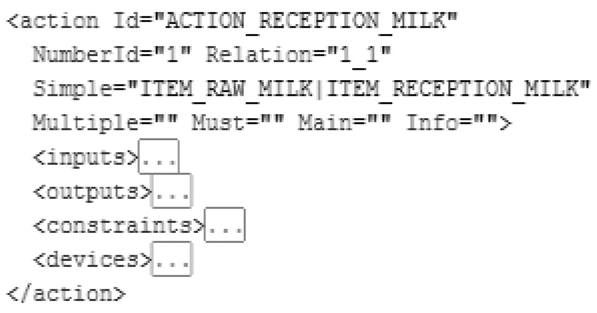
(1) Content. It has a correspondence with the data persistence layer from traditional applications.
(2) Web Application. It has a correspondence with the user interface layer from traditional applications.
(3) Content. It has a correspondence with the business logic layer from traditional applications

3.3 Inputs

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

3.3.1 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. This language has evolved constantly since the beginning of MDA FRW 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 4 is in its version 1.0 and has the following:



```
<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>
```

Fig. 5. Action specification in XPDML

(1) 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 5 can be seen a snippet of code which defines an action. Other sections are defined using snippets similar to this section.

(2) 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.

(3) 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).

(4) 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.

(5) Reports. The reports are very important, because collect and display necessary information about the system. There are several types of re-

ports, check list of the status of facilities, cleaning, product description, temperature control, etc.

(6) 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.

(7) TraceabilityPoints. Traceability is essential in the implementation generated by the MDA FRW. 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).

(8) 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 thanks to its identifier.

It is important to realize that the information contained in Figure 4, 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 7). 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.

3.3.2 Language files

The system that generates MDA FRW 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.

3.3.3 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 FRW to generate the system for a specific customer.

3.3.4 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.

3.3.5 Images

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

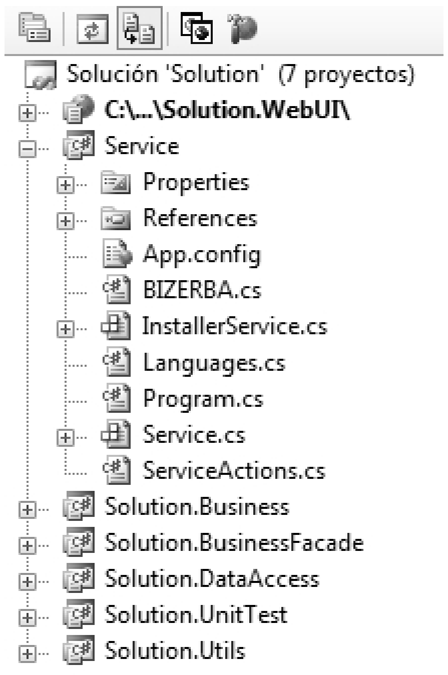
3.4 Outputs

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

3.4.1 Complete Solution

As a result of the MDA FRW process you will obtain a software solution consisting of seven projects ready to be compiled and deployed (In Figure 6 can be seen the aspect of the solution created automatically by MDA FRW working with Visual Studio):

(1) Web Application. In this project is all the source code required for the presentation of Web to the users. It includes, among other features, logs, Multilanguage support, authentication by profiles and usernames, and compliance with WCAG AA (20). With the Web application, it can be configured aspects of the system and see all the information about it. To make the different choices it will be necessary to authenticate on the system, and depending on the user profile, different choices can be performed (see the production map, see the map of the different locations, access to visualize traceability, access to reports, manage labels, review production forecasts, perform actions with articles, managing articles, manage lists of items, or access the administration menu).



Solución 'Solution' (7 proyectos)
C:\...\Solution.WebUI\
Service
Properties
References
App.config
BIZERBA.cs
InstallerService.cs
Languages.cs
Program.cs
Service.cs
ServiceActions.cs
Solution.Business
Solution.BusinessFacade
Solution.DataAccess
Solution.UnitTest
Solution.Utils

Fig. 6. Visual Studio solution generated by MDA FRW

(2) Windows Service. Given the characteristics of the food traceability it is clear that we must interact with hardware (terminals, scales, printers, bar-code readers, etc.). The Windows Service, which is running continuously on the server, will be the manager of hardware and the communications between all hardware. It is possible performed through the terminals managed by the Windows Service, as well as through the Web application, all actions that the system allows.
(3) Business Layer. To generate a solution with multiple layers, it has been created with a business layer to manage objects and allow then, their persistence in database.
(4) Business Facade Layer. Among the business layer and the data access layer is this layer which allows actions such as inserting, updating, deleting, or selecting.
(5) Data Access Layer. It is the responsible for independent access to the data. Thus, if the database suffers changes, the rest of the logic of the system is not affected.
(6) Unit Testing. The solution provides a project to make unit testing automatically without the need to create them by hand, thus saving time and effort.
(7) Utilities. It includes utility classes such as creating bar codes, batches, generating serials, or the management of label printing, among others.

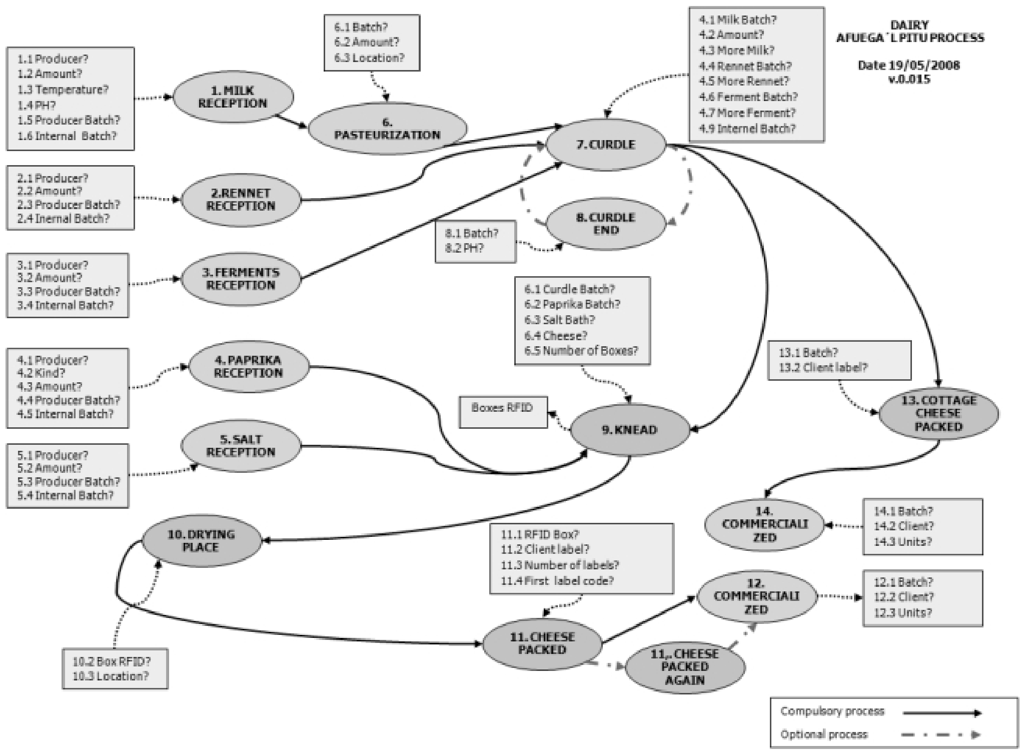
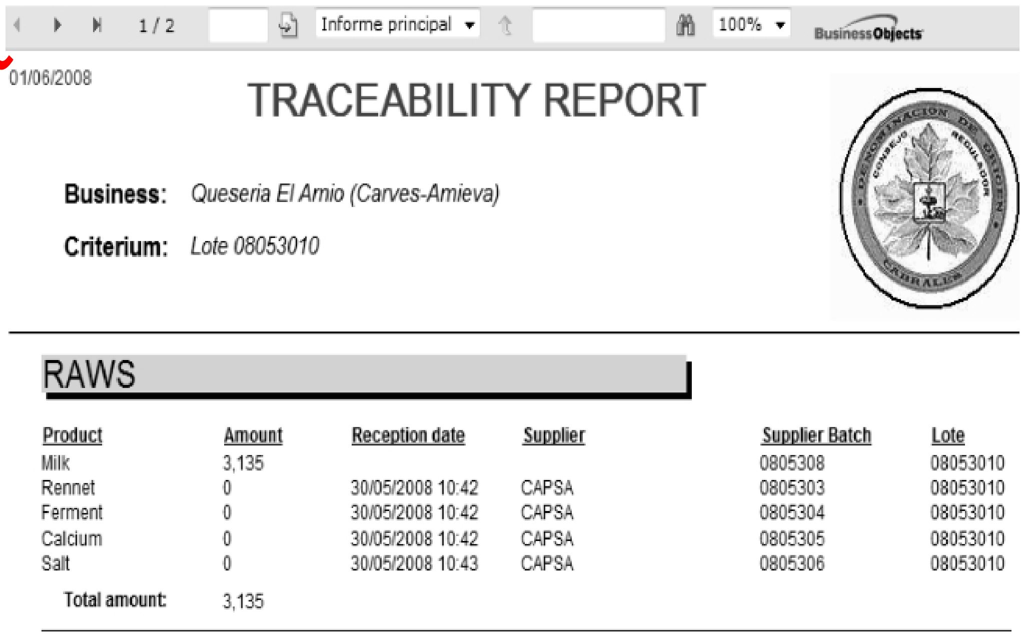


Fig. 7. Graphic representation of the model of Afuega'l pitu cheese



01/06/2008

TRACEABILITY REPORT

Business: *Queseria El Amio (Carves-Amieva)*

Criterium: *Lote 08053010*

RAWS

Product	Amount	Reception date	Supplier	Supplier Batch	Lote
Milk	3,135			0805308	08053010
Rennet	0	30/05/2008 10:42	CAPSA	0805303	08053010
Ferment	0	30/05/2008 10:42	CAPSA	0805304	08053010
Calcium	0	30/05/2008 10:42	CAPSA	0805305	08053010
Salt	0	30/05/2008 10:43	CAPSA	0805306	08053010
Total amount:	3,135				

Fig. 8. The Crystal Reports engine generates the reports

3.4.2 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.

3.4.3 Log files

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

3.5 Transformations

The heart of MDA FRW is the XPDML language. Thanks to 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 FRW also takes into account a transformation from PSM.XML to PSM-toPSM.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 base Framework. The new code is generated with text templates that has the ExpertCoder library (21) 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 FRW 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 FRW 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:

(1) 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.
(2) 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.
(3) 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.
(4) 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.
(5) 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 FRW, 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 7 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 FRW 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.

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). In the Figure 8, to show a different image of the generated system, you can see the fragment of a report generated with the system fully operational using the reporting technology from Crystal Reports.

5 Conclusions and Future Work

With both proposals MDA and Software Factories in action, we think that the most reasonable approach is a mix of the two proposals. We reduce the level of abstraction of MDA but at the same time, we are rising the level of abstraction of Software Factories, and thus, we can generate a fixed part of the system from scratch using the best practices to one specific platform, in our case .NET and others parts of the system are generated through the MDA layers (M3, M2, M1, and M0).

We will continue checking the pros and cons of MDA and Software Factories and we are looking forward to create a MDA system for all kinds of applications without a partial dependency of platform. Add also that we are satisfied with a system that is working in the food industry and growing each day.

It is clear that our MDA FRW 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.

Finally, we are thinking about how to enable someone to change at runtime all the variable parts of the system and obtain a dynamic graph of the system and how to add Model-Driven testing functionality (22)

References

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

[2] OMG, Mda guide. v1.0.1. http://www.omg.org/docs/omg/03-06-01.pdf. URL `http://www.omg.org/docs/omg/03-06-01.pdf`

[3] A. G.Kleppe, J. B.Warmer, W. Bast, MDA Explained: The Model Driven Architecture: Practice and Promise, Addison-Wesley, 2003.

[4] S. J.Mellor, K. Scott, A. Uhl, D. Weise, MDA Distilled: Principles of Model-Driven Architecture, Addison-Wesley, 2004.

[5] S. Leist, G. Zellner, Evaluation of current architecture frameworks, Proceedings of the 2006 ACM symposium on Applied computing (2006) 1546 – 1553.

[6] OMG, Meta object facility (mof) core specification. v2.0. http://www.omg.org/docs/formal/06-01-01.pdf. URL `http://www.omg.org/docs/formal/06-01-01.pdf`

[7] OMG, Omg unified modelling language infrastructure. v2.1.1. http://www.omg.org/docs/formal/07-11-04.pdf. URL `http://www.omg.org/docs/formal/07-11-04.pdf`

[8] OMG, Mof 2.0 xmi mapping. v2.1.1. http://www.omg.org/docs/formal/07-12-01.pdf. URL `http://www.omg.org/docs/formal/07-12-01.pdf`

[9] OMG, Object constraint language. v2.0. http://www.omg.org/docs/formal/06-05-01.pdf. URL `http://www.omg.org/docs/formal/06-05-01.pdf`

[10] OMG, Meta object facility (mof) 2.0 query view transformation specification. http://www.omg.org/docs/ptc/07-07-07.pdf. URL `http://www.omg.org/docs/ptc/07-07-07.pdf`

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

[12] J. Greenfield, K. Short, S. Cook, S. Kent, Software factories: assembling applications with patterns, models, frameworks and tools, Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications (2003) 16 – 27.

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

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

[15] D. Thomas, Mda: Revenge of the modeleres or uml utopia?, IEEE Software.

[16] S. Cook, Domain-specific modeling and model driven architecture, MDA Journal (2004) 2–10.

[17] M. Fowler, Model driven architecture. http://www.martinfowler.com.
URL `http://www.martinfowler.com`

[18] E. Parlament, The european parliament and the council of the european union. regulation (ec) no 178/2002 of the european parlament and of the council. http://eurlex.europa.eu.
URL `http://eurlex.europa.eu`

[19] P. Clements, L. Northrop, Software product lines, 2002.

[20] The world wide web consortium. http://www.w3.org (2008).

[21] Expertcoder. http://expertcoder.sourceforge.net (2008).
URL `http://expertcoder.sourceforge.net/`

[22] R. Heckel, M. Lohmann, Towards model-driven testing, Electronic Notes in Theoretical Computer Science 82.