The MDArte Experience
Organizational Aspects Acquired from a Successful Partnership between Government and Academia using Model-Driven Development

Rodrigo Salvador Monteiro[1], Roque Elias Assumpção Pinel[2],
Geraldo Zimbrão[2] and Jano Moreira de Souza[2]
[1]*Computer Science Department, Fluminense Federal University (UFF), Niterói, Brazil*
[2]*COPPE, Graduate School of Engineering, Federal University of Rio de Janeiro (UFRJ), Rio de Janeiro, Brazil*

Keywords: MDA, Organizational Aspects, Practical Experience, Success Cases.

Abstract: Developing and evolving critical information systems in order to cope with changes in regulations and laws is a constant worry for governments in the e-Government era. Due to the frequent challenges and some previous frustrating experiences, the Brazilian Government has searched for an alternative development method that could better fit its needs. At the same time, Brazilian Academia, represented in this case by the Computer Science Department of COPPE/UFRJ, has been researching how to bridge the gaps in order to harness the promise of Model-Driven Architecture in real life projects. A successful partnership between both started in 2005, which gave birth to a complete MDA development environment that would later be known as MDArte. The MDArte framework has been freely available since 2009 through the SPB portal (Brazilian Public Software Portal – www.softwarepublico.gov.br) and throughout its construction and evolution more than a dozen real projects have been built and maintained. During the development of such real life projects a lot of critical organizational aspects raised. The main contribution of this paper is to reveal and motivate the discussion on such organizational aspects that must be faced when deciding to adopt an MDA approach. In order to contextualize our perceptions, we present the MDArte experience, including its origin, evolution and current state. The major challenging and complex systems that have been developed with MDArte are presented as success cases along with an analysis of the benefits of using a Model Driven Development (MDD) approach.

1 INTRODUCTION

The Model-Driven Architecture (MDA, 2003) approach as released by the Object Management Group (OMG, 2013) aims at addressing a whole bunch of recurrent problems faced in information systems development. The promises of the MDA approach are attractive and the benefits presented in slide sheets make any software development manager yell, 'I do want to use that!' However, using the MDA approach in practice is far from trivial as can be confirmed by other initiatives (Rios et al., 2006); (Staron, 2006). There are huge cultural and technical gaps that may lead your pilot project to fail. Understanding the current limitations on one hand and the powerful potential in the medium and long run on the other hand, are key issues in avoiding unrealistic expectations. In this paper we describe the origin, evolution and current state of the MDArte framework (MDArte, 2013) in order to pinpoint the challenges involved in the adoption of an MDA approach. The MDArte framework has delivered a dozen information systems. Two of them that are considered to be the most challenging are presented as success cases: SICONV and SGDC-P. Their business contexts are addressed, the major challenges are pointed out, and issues where the MDA proved its value are highlighted.

This paper is organized as follows: section 2 presents the origin, history and current state of the MDArte framework; two success cases – the SGDC-P and SICONV are reported, respectively, in sections 3 and 4; section 5 summarizes the lessons learned regarding organizational aspects through almost ten years of experience; and finally, we conclude the paper in section 6 and outline the next steps as future work.

Salvador Monteiro R., Elias Assumpção Pinel R., Zimbrão G. and Moreira de Souza J..
The MDArte Experience - Organizational Aspects Acquired from a Successful Partnership between Government and Academia using Model-Driven Development.
DOI: 10.5220/0004714305750584
In *Proceedings of the 2nd International Conference on Model-Driven Engineering and Software Development* (MODELSWARD-2014), pages 575-584
ISBN: 978-989-758-007-9

2 THE MDArte FRAMEWORK

Model Driven Architecture (MDA) is a methodology for information system development based on values from Model Driven Development (MDD, 2003). The OMG released it in 2001, and its main goal is to formalize a methodology on code generation that uses the Unified Modeling Language (UML, 2005) and some other industry standards to build abstraction layers from model to code.

These layers represent the main idea behind the MDA: promoting the separation between what is considered specific to or independent of technology. Thus, the same model could be used to generate the information system for different platforms without too much additional work. From the models, more than 80% of application code can be automatically generated; leaving around 20% to be written to fit the specific platform needs (Guttman and Parodi, 2006). Based on the fact that the generation covers almost all the code, the development requires less specific knowledge from developers and reduces both development time and cost.

After the methodology was released by the OMG, different organizations started migrating and building code generation frameworks to work with its concepts. One of the most widely known and used MDA frameworks is the AndroMDA (AndroMDA, 2013), an open source project that deals with different target technologies. In Brazil, the project was extended due to the Brazilian Government's need to standardize the development of its information systems, resulting in the creation of the MDArte framework. Today, it is an open collaborative community involving public and private sectors, from industry to academia. Besides this, although both frameworks have been exploring the generation of specific information systems, they are still contributing to each other in order to streamline code generation.

2.1 Historical Context

The OMG – a non-profit international consortium of industrial and computing organizations created in 1989 with the purpose of establishing corporate modeling and integration standards – announced an initial (but incomplete) formal version of MDA specification in September 2001. In mid-2003 a more detailed definition of the architecture was published. In February 2003, an open source initiative released an MDA framework. This open source MDA framework was baptized as AndroMDA and by July 2003 it had been published in its final 2.0.2 version. From 2003 to 2005 COPPE/UFRJ developed a series of academic projects and small pilot projects using AndroMDA. Such projects aimed at identifying weak points that should be researched in order to provide viable solutions for the development of large scale real life information systems. In February 2005, a project involving the Ministry of Planning, Budget and Management, the Ministry of Defence and COPPE/UFRJ was established for the creation of the Brazilian National Codification System. One of the sub-projects included the development of a new version of the Catalogue Data Management System (SGDC-P) using the MDA approach via the AndroMDA framework. The AndroMDA framework can be freely customized by changing the existing plug-ins or creating new ones. In the AndroMDA context, such plug-ins are responsible for specifying how the UML models provided as input should be processed and what should be produced as output. Also, these plug-ins are commonly called cartridges by the AndroMDA community. Since the beginning of the development of the SGDC-P a series of new requirements for the AndroMDA cartridges have been identified. These requirements were analyzed, generic solutions to cope with them were designed and, finally, implemented and made available in new versions of the cartridges. These solutions are readily available for use in new projects. During the development of the Agreements Portal (SICONV), in a partnership between the Ministry of Planning and COPPE/UFRJ, which began in September 2007, the productivity gains obtained from the reuse of solutions embedded into the AndroMDA were already evident. The new set of AndroMDA cartridges constructed through this process was called MDArte and is publically available at www.softwarepublico.gov.br since 2009. Since then more than ten other information systems have been developed and put into production using this development platform. Nowadays, the MDArte community counts more than 2000 users and is maintained by its open source community.

2.2 Current State

MDArte had its origin in the academic field by way of a federal government partnership in the area of database research with the Systems and Computing Engineering Program (PESC), which is part of the Alberto Luiz Coimbra Institute (COPPE) of the Federal University of Rio de Janeiro (UFRJ) (PESC, 2013). COPPE maintains a sector dedicated to the

research of MDA methodology in which MDArte has shown to be an excellent learning tool. As a result, several projects related to the framework have been developed (e.g. Final Course Projects, Masters Thesis, etc) and published in conferences as well as national and international journals. Some current research topics are:

- Evolution impact analyzer for generated systems.
- Metrics analyzer, counting function points and other statistics (Pinel, 2012).
- Android application generation.
- Test flow generation with JUnit (Pinel et al., 2011).
- Python application generation.
- Dependency extractor and architecture violation identifier (Antelio, 2011).
- Development support tools.
- Integration with the jBPM framework.
- Support for Data Warehouse (DW) environments (Fernandes et al., 2010).

Many of these projects such as the JUnit generation, statistics generation and the development support tools are already part of the latest version of MDArte. Some other projects, like the dependency extractor and architecture violation identifier, are in the final phase of testing and should become public soon. Moreover, the Fluminense Federal University is now part of the MDArte community and there are plans to create a research group focused on MDA and MDD in the near future.

3 SUCCESS CASE 1: SGDC-P

The Catalogue Data Management System was conceived in partnership with the Brazilian Armed Forces Catalogue Center of the Defence Ministry. Its initial version, the SGDC, allowed Brazil to achieve a Tier 2 level on the NATO Codification System (NCS). Being a Tier 2 nation means to be able to actively participate in the catalogue of manufacturers and supply items offered through the NCS network. As a fundamental requirement for reaching such level, the country must have an approved information system that allows the exchange of codification messages following a rigorous standard defined by NATO. The development of the SGDC and its production availability resulted in the requested requirements being fully satisfied. The development of the SGDC-P began in 2005; this corresponds to an evolution of the original SGDC and includes parameterization flexibility as well as a series of technological innovations. Among the innovations it is worth highlighting the possibility of exchanging codification messages online using Web Services technology.

3.1 Major Challenges

If we analyze the NATO Codification System (NCS), we can split it into two major "layers": the first one is composed of the whole set of business rules, which manage the catalogue of manufacturers and supply items. The second layer is formed by the communication protocol, which is nothing more than an EDI (Bergeron and Raymond, 1992) protocol defined by NATO. In this protocol, the NCS transactions are an embedding of business transactions necessary to accomplish all the codification processes with the controls necessary for their transmission. Thus, in a simplified way we can portray the NCS as a great repository of distributed data, composed of manufacturers and supply items, which use the transactions as a way of querying, creating, changing and deleting this information, keeping all the client databases synchronized.

Ever since the NATO Codification System was conceived in the middle of the 20th century there has been no clear separation between the business layer and the communication layer and several interferences between them can be perceived. Also, because it is an old protocol, many restrictions were applied due to the high costs of communication and data processing existing at that time. And due to these restrictions, any new implementation in this system demands very complex programming. Currently, with the high level of maturity of tools and processes in Information Systems development, it is obvious that the separation between communication and business layers is fundamental for the development of a more robust system in terms of Codification rules (its core) that is less dependent on technological aspects for communication purposes. By adopting this practice, it is possible to gain benefits from standards already established by the software development community that have solved many communication problems in an efficient and maintainable way – as established by data communication via web services (Web Services Architecture, 2013) – and thus focus all the development efforts exclusively on business aspects in order to build a system fully compliant with all the codification rules and capable of meeting all users' requirements. Following this approach, the SGDC-P was built with more focus on Codification

rules, relegating communication aspects to a lower level of concern.

The above-mentioned approach of separation between communication and business layers lead us to defining the architecture depicted in Figure 1 for the SGDC-P. In the following we briefly describe the role of the main components of the SGDC-P Architecture:

MESSAGE: By means of the message, the actions demanded by the system operator are stored for processing. A message can act on manufacturers or supply items. This way, when the operator wants to query, create, change or delete data, the system creates a message which will run through the system until it is processed.

MESSAGE PROCESSOR: The Message Processor (MP) is a service capable of interpreting a message, executing the actions required and validating them. In the first step, the MP does not update the database, but simulates the actions indicated in the message and retrieves a temporary version of the item/manufacturer with the changes required by the message and the possible errors originating from these. Depending on the step in the codification workflow the changes originating from the message processing may or may not be persisted in the database.

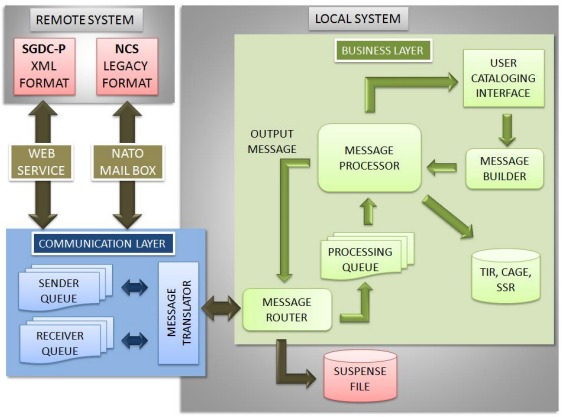

Figure 1: SGDC-P Architecture.

INTERFACE FOR USER CATALOGUE: These are the screens that allow interactions between the operator and the system. It is through some of these screens that the messages are built. Once the message is completed, the MP is demanded and a feedback is provided for the operator. This is particularly interesting, since it allows the operator to know in advance the expected results before the message is actually processed and the actions have persisted in the database. Another advantage is that there are no duplications in terms of business rules; all the validation is done uniquely through the MP.

MESSAGE BUILDER (MB): The MB works with the version concept (for items and for manufacturers) in order to let the system know which actions the operator has performed to modify the item or the manufacturer. Based on the differences between the previous version and the modified version, the actions demanded by the operator are deduced by the system. This solution allows the storage of all the actions performed for a given item, as well as permitting validation of those actions, by using the MP to simulate the processing. As such, during the codification process the codifier is capable of being aware of the possible errors encountered during the process, before finishing its actions.

MESSAGE ROUTER (MR): The MR defines if the message is to be processed locally (into the system) or remotely (the message is to be sent via web service/XML protocol or via NCS Legacy format).

MESSAGE TRANSLATOR (MT): By means of the MT, NCS Legacy format transactions and XML messages are converted into SGDC-P internal messages and vice versa. This is a very important step, as it lets the MP work exclusively with a unique kind of standardized message (the SGDC-P internal message), thus allowing it to focus exclusively on business validations. Any error related to the format of the transmission is detected by this service, preventing the continuation of the message to the next step which would be the business processing.

RECEIVING, SENDING AND PROCESSING QUEUES: By means of queues, it is possible to ascertain that the message will reach its destination. Once in the queue, the message will be processed obeying chronological order.

The requirements associated to the above mentioned components were only partially fulfilled by the features available in MDArte at the beginning of the project. In the next section we discuss some examples of new features developed in the MDArte cartridges in order to cope with the development needs.

3.2 Sample of Model Transformation Evolution in the SGDC-P Context

The SGDC-P development presented a lot of new challenges for the MDArte team. The existing model transformations at the beginning of the project had

no adequate support for a lot of features. In fact, the development started in two forefronts: the first one (thread 1) started implementing the use cases that could be expressed through the existing model transformations; the second one (thread 2) started building new model transformations for coping with the new requirements. Specially, looking at the description presented in the previous section we can notice the need of having ways of modeling message queues and the exposure of services through web services. We chose to present these two evolutions in the following in order to illustrate how the work was conducted.

When it comes to message queuing, the missing features intended to model components representing the queue, the queue listener and the queue client. Introducing a semantic notation for expressing such components at a platform independent level should be enough for building new transformation for generating the required artifacts at the target language. In such a way, we created two new stereotypes and one new tagged value. The first new stereotype was *«MessageListener»* associated with the new tagged value *@mdarte.service.QueueName*. These new elements were conceived to expresses that a class receiving the stereotype *«MessageListener»* would behave as a component waiting for receiving messages sent to a queue identified by the name provided in *@mdarte.service.QueueName*. The second stereotype was *«MessageListenerClient»* and the class receiving this stereotype would behave as a component able to send messages to a specific queue. The notation chosen to express to which queue the queue client would send the messages was to model a dependency relationship between the class with the *«MessageListenerClient»* stereotype and the class with the *«MessageListener»* stereotype. Figure 2 presents a class diagram using the notation described to represent the three queues depicted in the SGDC-P Architecture in Figure 1.

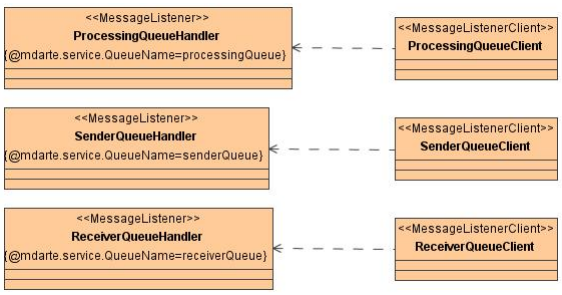

Figure 2: Modeling Message Queuing Behavior.

The team responsible for evolving the model transformation performed changes in the EJB cartridge in order to register new components aware of the above modeling features. New templates were developed to generate all the artifacts required for deploying a MDB (Message Driven Bean) component of the EJB (Enterprise Java Beans) specification. The most interesting feature is that the development team focused on thread 1 only needed to be told about the modeling features depicted in Figure 2. No details on how building and deploying an MDB component were required for them.

Regarding the exposure of services as web services once again there were a lot of features missing. How would we model that a service must be exposed or that some component will consume an external service? In this case, we have created three new stereotypes and one new tagged value. A stereotype *«WebService»* was created in order to be used together with the existing one *«Service»* in order to express that it should be published as a web service. A new stereotype *«WebServiceClient»* associated with the tagged value *@mdarte.web.service.client.wsdl.location* were created to model a component that should behave as a consumer of a web service described by the WSDL (Web Service Description Language) file provided in the tagged value. Finally, a stereotype called *«WebServiceData»* was created to indicate that a modeled data structure will be sent or received via a web service. Figure 3 presents a diagram using the new embedded semantic information.

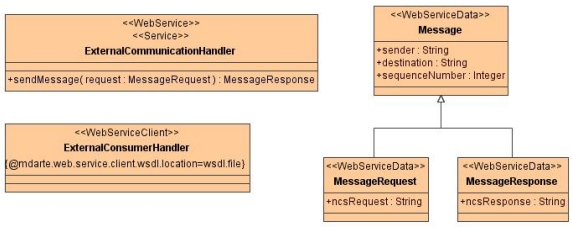

Figure 3: Modeling Web Service Components.

The model transformation team once again evolved the cartridges adding new components associated to the new notation elements. New templates were developed to generate the artifacts required as input to the JWSDP (Java Web Services Development Pack) for the elements annotated with the *«WebService»* and *«WebServiceClient»* stereotypes. Other templates were implemented to generate Java classes conforming to the W3C standard for supported data types when the *«WebServiceData»* stereotype is provided.

3.3 Analysis of MDA Benefits

The major challenges involved in the SGDC-P development address technical issues. Therefore, designing solutions to cope with the translation between different protocols and to perform the role of the architecture's components are central problems perceived by the client. This was an extremely favorable environment for the MDA adoption. It was easier for the client to understand what the infrastructure team was working on.

The development of the SGDC-P required solutions for deploying and managing queues, executing simulations, web service deployment and W3C Data Structure compliance, W3C Data Structure vs. Internal Structure translation. The use of MDA allowed delivering solutions to all these topics. Most importantly, in almost all of them the development team was simply told: "focus on the business issues, everything else is already solved". In other words, the experience of applying MDA on the SGDC-P development confirmed that it really proves its value when it comes to hide framework technical details and protocol standards. For instance, if you want to publish a service on the web via a web service, just add a stereotype to your service saying so. If you want to exchange data through web services using W3C compliant data structures, just model your data and add a stereotype saying that it will be used to transport data through web services. By doing so, the development team was released from a whole bunch of technical details and could focus on the business rules implementation.

4 SUCCESS CASE 2: SICONV

With the identification of the misuse of public funds personified by the overpriced ambulances scandal, the Audit Court (TCU) determined by the 2066/2006 accord "... to the Planning, Budget and Management Ministry, in order to allow the transparency that should be afforded public actions, as a way of facilitating social control and the principle of publicity …, a technical study will be presented to this Tribunal for the implementation of a web information system platform that allows online tracking of all agreements and other instruments of law used to transfer federal funds to other agencies/entities, federal entities and private sector entities; that can be accessed by any citizen via the worldwide web and contains information related to the formalized instruments ...". Compliance with this determination was made possible through a partnership between COPPE and the Ministry of Planning, Budget and Management for the development of a new Agreements Portal (SICONV) to replace the old system which only satisfied a small part of the process implemented by the Union and was not accessible to ordinary citizens.

Due to the short amount of time for the construction of the new information system, the COPPE staff responsible for the project made use of their experience in the development of systems focused on models and MDA (Model Driven Architecture) to accelerate development and meet deadlines. After meeting the requirements agreed upon for the partnership, the development of the new SICONV was transferred to SERPRO (Federal Data Processing Service) through training focused both on business and the MDA technology used. The system was successfully transferred and is now fully maintained by SERPRO, which is totally qualified to deal with the system's evolution using the new technology.

SICONV (the Agreements Portal) is responsible for the voluntary transfer of resources from the Union to the states, municipalities and private non-profit entities, which exceeds tens of billions of reais annually (R$23.78 billion in 2007, which roughly corresponds to US$12 billion). The progress of the transfers and allocation of resources can be tracked online on the website http://www.convenios.gov.br.

4.1 Major Challenges

In Brazil, the federal constitution states that the Union is responsible for the majority of tributes, levied in the form of taxes, improvement contributions, compulsory loans, import tariffs, export tariffs, income tax (IRPF and IRPJ), tariffs on industrial products (IPI), duty on credit transactions (IOF) and rural property tax (ITR) amongst others. As a consequence, mechanisms were created to decentralize these resources to other spheres of government (states and municipalities) or to private non-profit entities in order to meet the needs of society. A portion of these resources comes from mandatory transfers and follows rules predefined by the Federal Constitution, or from specific laws as in the case of the Municipalities Participation Fund. The other portion comes from voluntary and discretionary transfer for use in projects of mutual interest, accomplished by means of formal instruments generically known as "Covenants" that, when providing for the intervention of an official bank, are known as "Transfer Agreements".

According to Hely Lopes Meireles, “Covenants are agreements signed by various kinds of public entities or between them and private organizations, for achieving objectives of common interest to the participants”. Over the past decades, the Federal Public Administration has been consistently investing in the implementation of automated information systems to improve the collection, allocation and execution of public resources; however, until recently such initiatives were very weak in terms of decentralized resource execution, since in most cases the other administrative spheres did not have instruments equivalent to the federal sphere for control and transparency of spending. Such a situation left these resources subject to arbitrary uses and in opposition to the original objectives of the transfers - inflated pricing and misappropriation of funds frequently appeared in the headlines of the major newspapers. It is the transparency in the operation of these funds that the Agreement and Transfer Contract Management System (SICONV) is focused.

Among the main challenges, the ones that stand out are those relative to major clients and to the progressive development of the modules – due to the prerequisites not being completely defined yet. The lack of standardized enforcement mechanisms has led many authorities to develop procedures and even their own automated solutions. Associated with this, a certain aversion to controls intensified the resistance to change and significantly complicated the implementation of the new system. The human resistance to change is always a constant, but in the case of this initiative there are particular difficulties associated with almost all the actors involved in the process. Strictly speaking, everyone agrees with the objectives proposed by the new system and wanted the change, given the successive scandals that constantly appeared in the major headlines; however, in practice, the absence of standardized procedures and the superficial control led to everyone developing a method for treating the theme and investing in homegrown solutions, some of good quality but not satisfying the full scope and purposes and with knowledge restrictions for the whole process.

4.2 Sample of Model Transformation Evolution in the SICONV Context

A very sensitive and present requirement in the SICONV development was the ability to track the user responsible for inserting, updating or deleting any information. The need of storing change logs was in evidence as well. Therefore, creating automatic mechanisms for coping with such requirements would release the business development team from such worries. The model transformation team delivered two options for expressing the need of an auditing mechanism. The first one was to simply add a general property on the project stating that every class on the Data Layer must be auditable. The second one was more selective and allowed the business development team to add a new stereotype <<*Auditable*>> on the Data Layer components that must be handled by the auditing mechanism. Figure 4 presents an example on how this annotation is modeled.

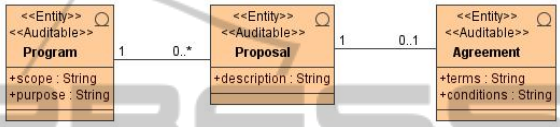

Figure 4: Modeling Auditing Behavior.

The model transformation team performed changes in the Hibernate cartridge in order to automatically generate all database objects required for storing the audit trail. Every auditable Data Layer component received additional attributes for logging the user, the timestamp and the operation performed, as well as a complementary log table to store the historical data. Finally, an interceptor component was attached to the Hibernate framework in order to monitor every data manipulation performed by the application. Performance tests were conducted to evaluate the overhead created by the auditing mechanism and the solution was tuned to reach an acceptable level for the application context.

Besides, it is worth mentioning that the strong security issues involved in the application context demanded solutions to prevent some kind of common attacks. The model transformation team applied well-known solutions to generate a safe code preventing, for instance, script-injection and SQL-injection.

4.3 Analysis of MDA Benefits

The major problems faced by the SICONV development were far away from the technical perspective. Most of them were cultural and social problems. Moreover, due to the absence of standardized procedures there were no clearly defined business processes and requirements.

The adoption of MDA in this project was clearly made to take advantage of the team expertise and high productivity. The main benefits of using MDA

had little to contribute to the major challenges faced. Indeed sometimes it was even an additional argument to strengthen the cultural resistance. As the other extreme faced in the SGDC-P development, the environment was extremely contrary to the MDA adoption.

It is true that the use of the MDArte framework allied with the team expertise was decisive in meeting the strict and short deadlines imposed. Although, throughout the development its use was many times blamed as the responsible for different problems that would arise in any other development option. Problems from performance and security issues to bugs in the Java framework chosen for the web layer (Struts framework) were assigned to the adoption of MDA.

Fortunately, the development team was always capable of quickly detecting the cause of reported problems and give fast answers and solutions. Nevertheless, it was very difficult for the client to understand the technical details and the doubt about if the MDA was the right choice started to gain strength.

5 LESSONS LEARNED

In this section we summarize a set of lessons learned over the past almost ten years of experience in managing, planning and developing information systems using the MDA approach. Our intention is to share some hints that can help other MDA practical initiatives to avoid common pitfalls, resistance and unrealistic expectations. Although, before presenting such hints some important remarks must be made. In our discussion we are constrained to traditional information systems. This constraint removes from the scope safety critical applications, such as avionics and automotive control systems for example. In such scenarios qualified code generators are being used which cannot be adapted or tempered by the development team due to certification considerations. In other words, do not try to change the behavior of the code generator if you do not have full control and knowledge on how it must work! Once that the constraints are imposed, please find our hints in the following:

- Do not start any MDA initiative without the support of a solid MDA infrastructure team

It seams obvious but we have noticed many initiatives trying to use MDA in a project facing it as a off the shelf package or component of the development environment. The real benefits and gains of applying MDA come when your team is capable of assuming total control over the model transformations in order to adapt and evolve them aligned with the project requirements. Otherwise, your development team will simply use the chosen MDA tool as a regular automatic code generator module and it is almost sure that the team will struggle and complain about the code being generated or the flexibility provided. Actually, this is why many automatic code generation initiatives have failed. The code being generated must be adapted to meet the architecture and patterns established as standards for the running project. In practice, we have witnessed development teams trying to adapt them selves to the generated code. If you are trying to discipline your team to obey the established standards then its fine, but normally this happens because the code being generated is not aligned with the project requirements and the development team has to find ways to overcome the barriers imposed by it. If the latter happens, soon the development team will resist and refuse to support the use of MDA. Indeed, some projects we conducted at the beginning faced such problems. Some projects were conducted only by developers without the support of a background infrastructure team. Some of them after while quit using MDA.

- Establish clear priorities for the MDA infrastructure team

A very common error in a project applying MDA is to consider the capacity of the MDA infrastructure team when planning the deliverables schedule and making commitments. This is not to say that the MDA infrastructure team should not work and help on the business requirements development. Indeed the MDA team must help in such tasks as much as it can, but there must be some time reserved on the planning to cope with unexpected needs that could not be envisaged. The problems arise because as soon as the model transformations are well established and stable it is tempting to engage all the MDA infrastructure team direct in the implementation of business requirements. The project will face new requirements and challenges that will require changes and evolutions in the model transformations. Therefore, there must be some time reserved for performing such tasks. Over the projects we have developed we acquired statistics showing that on the average we have consumed 15% of the total hours devoted to the project to evolve and maintain the model transformations. An alternative we have been using to reduce the pressure on consuming the MDA infrastructure team resources for business requirements development is

to share this team among different projects.

- Clearly identify the main benefits MDA can bring to your project

One very import issue is to identify what the development team and mainly the stakeholders of the project are expecting as benefits of the MDA adoption. High productivity, up-to-date documentation, obedience to the established patterns, quality of the code being produced, independent of the set of expected benefits keep one thing in mind: none of them come without focus and effort. The adoption of MDA can indeed provide all the above benefits beside others and we are witnesses of that. However, it is no magic solution. Instead, it will provide the team project with tools to reach the desired benefits. For instance, if you do not work on a good architecture design and implement it in your model transformations should not expect to magically produce a well-structured system. MDA is an elegant way of applying good practices recommended by software engineering without flooding the developers with annoying tasks. MDA has a huge potential, it does have payoff, but it requires hard work to realize it in practice.

- Keep expectations under control, especially when it comes to gains of productivity

Avoid starting your project flooding your sponsors with the promises of commonly seen slide sheets about the wonders of MDA. Obviously you must expose them but keeping everybody feet on the ground. Some benefits only come in the long run and this may frustrate some of the project's stakeholders at earlier stages. For example, when it comes to gains of productivity we have cases confirming the observations of (Hutchinson et al., 2011) having 4 or 5 times increase in productivity. However, at least in our projects we were not able to reach this level on the average. Only developers with high expertise were able to reach those levels. If you have created expectations for such high productivity levels regarding your whole team someone will be frustrated even if you improve the productivity at lower levels.

- Do not try to automate everything using model transformations

Resist to the impulse of automating everything. Current modeling languages do not have enough expressive power to capture every requirement for general information systems. Unless you intend to constraint your domain and work with a domain specific language (DSL), keeping a rate from 60% to 80% of the code being automatically generated is already a good result. The main risk of automation levels near 100% is to imprison the development team and force it to demand too many model transformations evolution to cope with the business requirements flooding the MDA infrastructure team. Moreover, introducing modeling elements at the same level of abstraction that would be used directly into the specific language platform is a signal that you have crossed the optimal level of automation.

- Do not reinvent the wheel

The use of MDA adds a lot of value by bridging the gap between models that represent the application behavior and the platform code that implements it. However, how far we should go with the implementation must be a constant worry as a lot of new tools arise in order to help in different implementation issues. For instance, the Hibernate Java framework helps a lot on the object-relational mapping. A lot of workflow engines (e.g. jBPM) are available, as well as business rule engines (e.g. JBoss Drools). Instead of solving these issues by delivering full solutions built in the model transformations, it is recommended to generate artifacts that can be plugged or fed into the platform specific solutions that already deal with it. By doing so, you can keep the desired information in your models and take advantage of the tools provided in your target platform. Moreover, if you are targeting a new platform and building cartridges from scratch, surveying the available tools and frameworks and selecting the ones that will be used is part of the homework. Summarizing, it is important to think about the MDA approach as a layer above the traditional development which links the models to the platform engines, frameworks and what so ever you would use if you were not using MDA.

- Be prepared for evangelizing the MDA culture

Be prepared to explain one million times what MDA can and cannot do. Unfortunately, MDA concepts are not known in general industry and even the acronym MDA has no meaning for the general IT public outside the academia. Personally, we have been inquired more than once if it was a new kind of MBA (Master of Business Administration).

- Be prepared to be the first one to be blamed for everything that goes wrong

It is sad, but in this case the question "Do you have someone or something to blame?" will always be answered with "Yes, MDA!". As reported before when discussing the use of MDA in the SICONV development, whenever facing any difficulty, the use of MDA will be responsible for it. This issue

enforces the need of a solid MDA infrastructure team that must be ready to quickly identify the origin of problems even if they have nothing to do with the use of MDA. It is extremely critical because not observing and understanding the resistance in a hostile environment will lead your initiative to discredit. Differently, identifying the source of the problem quickly and helping solve it even if it has nothing to do with the use of MDA will strengthen the project as a whole.

6 CONCLUSION AND FUTURE WORK

This paper presented a well-succeeded partnership between the Brazilian Government and Academia, which led to the creation of the MDArte framework, representing a complete MDA development environment used to develop more than a dozen real information systems projects. We presented two cases of success, SICONV and SGDC-P, which were selected for being considered the most challenging ones from different perspectives: technical and cultural. In the sequence we provided a section listing a set of lessons learned from our experience aiming to highlight critical organizational issues that must be observed when deciding to start a project using MDA.

It indeed is a cliché but: "There is no silver bullet!". MDA indeed helps a lot in many different aspects, from establishing standard architecture and patterns, passing through reuse and reaching up-to-date documentation and productivity gains. However, the intrinsic complexity of developing an information system is still there. To avoid creating unrealistic expectation has been a key success factor in our projects allied with hard work of the MDA infrastructure team.

As future work, we intend to compile statistics from all developed projects in order to disclose other successful stories and help to disseminate the use of MDA for information systems development. Furthermore, the universities supporting the MDArte will continually research new capabilities that can be explored and release prototypes through the MDArte community to be evaluated by the adherents projects.

ACKNOWLEDGEMENTS

This article would not have been possible without the support of CNPq, CAPES, FAPERJ, Fluminense Federal University (UFF) and COPPE at the Federal University of Rio de Janeiro (UFRJ).

REFERENCES

MDD, *Model-driven development*, IEEE Software Special Issue, Mellor, S. J., Clark, A. N., and Futagami, T. (eds.), vol 20, n. 5, September 2003.

OMG, Object Management Group. URL: http://www.omg.org. Last visit: Feb 2013.

UML, *Object Management Group, Unified Modeling Language (UML): Superstructure*, version 2.0, August 2005.

Guttman, M., and Parodi, J., *Real-Life MDA: Solving Business Problems with Model Driven Architecture*, 1st ed. Morgan Kaufmann, 2006, p. 224.

AndroMDA. URL: http://www.andromda.org. Last visit: Feb 2013.

PESC, COPPE, UFRJ: http://www.cos.ufrj.br. Last visit: Feb 2013.

MDA, *Object Management Group*, MDA Guide, version 1.0.1, June 2003.

MDArte: http://www.softwarepublico.gov.br/dotlrn/clubs/mdarte. Last visit: Feb 2013.

Bergeron, F., Raymond, L.. "*The advantages of electronic data interchange*". ACM SIGMIS Database, PP 19-31, 1992.

Web Services Architecture (W3C Working Group Note). http://www.w3.org/TR/ws-arch/. Last visit: Feb 2013.

Hutchinson, J., Rouncefield, M., Whittle, J., and Kristoffersen, S: "Empirical assessment of MDE in industry". *Proceeding of the 33rd International Conference on Software Engineering, 2011*.

Rios, E., Bozheva, T., Bediaga, A., and Guilloreau, N.: MDD Maturity Model: *A Roadmap for Introducing Model-Driven Developmen*t, Proc. ECMDA 2006.

Staron, M.: *Adopting Model Driven Software Development in Industry* - A Case Study at Two Companies. Proc. MODELS 2006.

Pinel, R. E. A., Carmo, F. B., Monteiro, R. S., and Zimbrão, G.: *Improving tests infrastructure through a model-based approach*. ACM SIGSOFT Software Engineering Notes 36(1): 1-5 (2011).

Fernandes, L. A., Neto, B. H., Fagundes, V., Zimbrao, G., Souza, J. M., and Monteiro, R. S.: Model-Driven *Architecture Approach for Data Warehouse*. ICAS 2010: 156-161.

Antelio, M.: *Architectural Breaches Detection in a Model Driven Development Scenario*. Master Thesis, COPPE/UFRJ (2011).

Pinel, R. E. A.: *Function Point Analysis Supported by Model Driven Architecture. Master Thesis*, COPPE/UFRJ (2012).