A Domain-Driven Approach for Enterprise Development, using BPM, MDA, SOA and Web Services

Fabio Perez Marzullo

Federal University of
Rio de Janeiro – UFRJ, COPPE
Database Laboratory, Brazil
fpm@cos.ufrj.br

José Roberto Blaschek

State University of Rio de Janeiro –
UERJ – FAF, Brazil
blaschek@attglobal.net

Jano Moreira de Souza

Federal University of
Rio de Janeiro – UFRJ, COPPE
Database Laboratory, Brazil
jano@cos.ufrj.br

***Abstract*— This paper presents preliminary study results of a prototype architecture created with the purpose of using a domain-driven approach to shorten the development of software projects. It is accepted that domain based development is playing an important role on IT projects today. The following discussion presents a way of using different project sites to work together and establish a shared development environment. Using a set of standards and theories such as BPM, MDA, SOA and Web Services, the proposed architecture indicates that, with the aid of standard and controlled techniques, it is possible to obtain significant gains on software scheduling and cost.**

***Index Terms* — Domain Driven Development, Service Oriented Architecture, BPM, MDA, code generation.**

I. INTRODUCTION

Research conducted in recent years has shown that domain-driven approaches are becoming important tools in software development [1]. This paradigm implies new ways of understanding and applying development techniques to conduct software projects. As stated in [2], software development is commonly applied to the automation of processes or to providing solutions for business problems that exist in the real world. We must understand from the beginning that software is originated from and deeply related to its domain. Therefore, domain models are capable of capturing domain-specific knowledge of a certain business application and carefully pinpoint details that should be documented to correctly represent real-life problems.

The heart of software is its ability to solve domain-related problems for its users. However, when the domain is complex, domain modelling become a difficult task calling for the concentrated effort of talented and skilled people [20].

The model is the team's agreed-upon way of structuring business domain rules and distinguishing the elements of most interest. For example, the Business Process Management [17] theory come to join efforts to aid business domain design and, through the automation of business processes, offer better technical solutions to real life problems.

Concurrently, much has been achieved since the introduction of Model Driven Architecture standards and techniques. By releasing their Model Driven Architecture (MDA) [10 and 12], the Object Management Group (OMG) [3] proposed a new development concept toward existing traditional paradigms. It created a new and exciting research area in which it would be possible to develop truly independent and powerful programming environments capable of achieving new levels of productivity, performance and maintainability.

All concepts cover co-related theories that, when used together, might prove powerful in helping project architects do their jobs. Therefore, the real eye-opener here is how to combine these technologies, as well as others like Service Oriented Architecture – SOA, to create an useful development environment capable of orchestrating components and services to solve problems that are currently faced by enterprises.

By pursuing the vision of creating intelligent solutions through the use of BPM, MDA, SOA and Web Services, this paper presents research conducted with the purpose

This work was supported by the Federal University of Rio de Janeiro – UFRJ, COPPE Database Laboratory Database Performance Assessment Using MDA.

Fabio Perez Marzullo, M. Sc. from Federal University of Rio de Janeiro Rio de Janeiro, Brazil. (e-mail: fpm@cos.ufrj.br).

José Roberto Blaschek from the State University of Rio de Janeiro, Rio de Janeiro, Brazil, (e-mail: blaschek@attglobal.net).

Jano Moreira de Souza, Ph.D. from Federal University of Rio de Janeiro, Rio de Janeiro, Brazil, is the head of the Database and Knowledge Management Department (e-mail:jano@cos.ufrj.br).

of proving that domain-driven approaches can promote a more efficient and rich development environment, which lay down an architecture that will adapt to ongoing changes in the business environment [4].

II. Problem Definition

Since 2005, the Ministry of Defence, the Brazilian Navy, the Planning Ministry, and the Database Laboratory of the Federal University of Rio de Janeiro have joined forces to use a new development approach to improve a set of management practices applied to software projects. After shifting from traditional development paradigms to Model Driven Development – MDA, we have obtained important results regarding productivity and project cost. To illustrate, by using MDA tools [19], we have cut our development time by approximately 20%. Basically, we have gained more efficiency in our development process without inflicting on requirement quality.

After these three years, we have seen that the adoption of new technologies, when carefully applied, might prove very useful as the results came up satisfactorily. Along this period, after delivering many software systems, we decided to standardize the way in which domain modelling was done. We felt that by doing it we could use these standardized models to develop new software projects, not from scratch but, starting a few steps ahead.

Given the foregoing, it is necessary to explain the problematic scenario we were faced with in public organization projects. Essentially, it was common practice in public organizations to develop similar software more than once, and wasting taxpayer's money, when a simple generalization of concepts and a cooperative policy could speed up development by using previously conceived domain analysis.

To solve it we began by stating three topics that needed to be immediately addressed:

1. How to standardize domain and services analysis in order to allow different organizations to use the same concepts?
2. How to implement a distributed environment in which different organizations could share their domain analysis?
3. How to automate the development process by using BPM, MDA and SOA theories?

The following sections try to answer each question by detailing what we have done to create the development environment, and the way we have orchestrated this development environment in order to allow a domain-sharing context.

III. Domain Modelling

When embarking on a software project, one should focus on the domain it is operating in [2]. Therefore, the domain modelling process should be standardized in a way that any business domain could be modelled by any project architect in any development environment in the same way.

The idea of software components servicing other software components in a cooperative environment is a long established approach to software design. Web Services, today, is a technological asset that is revitalizing the design of enterprise architectures by enabling the orchestration of different business contexts [6].

Also, in [7] we understand that *"…SOA means that we don't have to build applications any more – all we have to do is invoke other people's services."* Thus, service implementation is merely a collection of service invocations, and the services invoked are implemented by other service invocations, and so on, *ad infinitum*.

The advent of JBPM [18] has allowed software engineers to share business concepts as automated processes and workflows, with business–standards orchestration, with in a flexible and easy-to-use approach.

In this context, we have established a standard Service-Oriented Architecture to model business domains for projects conducted in public organizations.

Basically a business domain should be represented by a set of cooperative business services, either local or remote, while services consist of a set of software components, either local or remote.

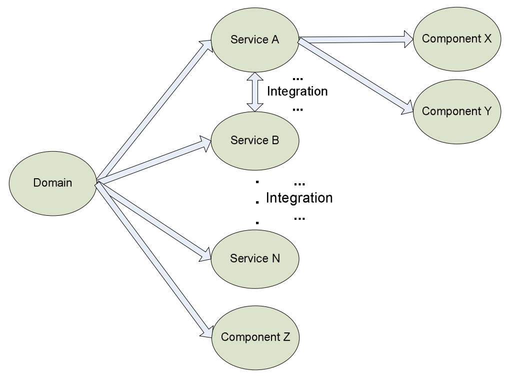

Figure 1. The domain model links to different services and components. Services and components are used to generate software code via the MDA tool. Domain model information is stored in XMI files.

The next topics detail how the project architect should design domain models in such a way that it could be reused by different development teams.

IV. Domain-Driven Development Conception

Our primary goal was to create a development environment in which architects could reuse domain models in different projects.

Throughout ongoing projects, MDA was already being systematically used in software development. The process was similar in every project and both software architecture and coding produced the same software artefacts due to the standardization of tools and analysis processes.

By creating an unified domain modelling process, we were able to broaden our abstraction perspective and share generic business concepts across different public organizations.

To come up with useful domain artefacts we needed to create a domain representation that would indicate the services or components that should be included in the domain model, and that should be used for implementation by the MDA tool [19]. For that purpose we followed a five-step design process, in order to enable domain-driven development:

1. Domain-Modelling. This stage was responsible for identifying the services and software components the domain required. Our goal was to create an infrastructure that could cope with any service or component ever modelled in any project.

2. Business Processes Definition. This was responsible for modelling the domain processes using the JBPM Eclipse plug-in [18]. It was responsible for creating the *process definition* file containing domain information on services and components:

```
<?xml version="1.0" encoding="UTF-8"?>
<process-definition xmlns="urn:jbpm.org:jpdl-3.2"
name="simple">
  <start-state name="start">
     <transition name="to_authentication" to="Authenticate">
     </transition>
  </start-state>
  <state name="Authenticate">
    <transition to="Order Request"
      name="to_order_request"></transition>
  </state>
  <state name="Order Request">
    <transition to="end" name="to_end"></transition>
  </state>
  <end-state name="end"></end-state>
</process-definition>
```

Code Snippet 1. The `processdefinition.xml` file representing the domain model.

The *process definition* file aggregates domain information by modelling its business processes. Each state is used to represent a service responsible for executing the necessary actions. Specifically, each action should point to a XMI file representing the service model.

3. Storing Domain Services and Component models. The third stage implied in storing the models created by project architects in accessible databases. Each project should have its own database and expose their content through a Web service.

4. Exposing and Sharing Domain Models. This stage relates to publishing the models in project Web services. Every project should have its own Web service in order to allow other projects to access domain information as well as services and component models.

5. Automate Code Generation. This stage uses the domain models to generate software code. According to that point, specific business rules might be modelled and incorporated in the final product.

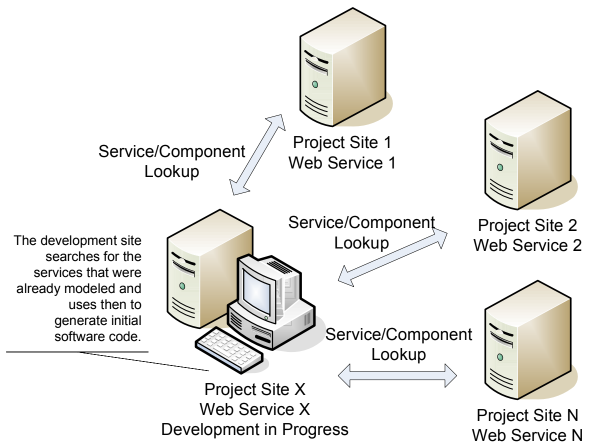

Figure 2. Domain sharing process: The development site seeks domain processes definitions and uses them to generate initial project software code.

V. Domain Sharing

Domain Sharing consists of a set of Web services, allocated to different project sites, with the purpose of exposing services and component models created to support different business domain rules. The sharing process is straightforward: the project architect creates services and component models for the project it is designing; it then stores them on a domain database and exposes them through a Web service. The Web service publishes the services artefacts (basically the process definition file and the services models) via WSDL files, running on a Web application server [16].

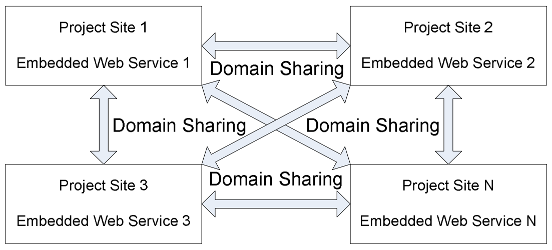

Figure 3. Domain Sharing Architecture.

VI. AUTOMATED CODE GENERATION

Our Domain-Driven Development environment consists of a set of integrated tools. The main development tool is Eclipse IDE [13]; the MDA plug-in called mda4eclipse [19] is responsible for generating software code by reading the UML2Tool [8] use cases and design models and the JBPM [18] process definition.

In every project, artefact development is standardized, both by the way analysis and design models are conceived and the way in which the software is generated. If modelling and code generation are done in a standard way, regardless of the project site, sharing domain models from project to project becomes possible. Therefore, all that the architect needs to do is to locate if the service one needs is available at any project site and download it. After obtaining this model the mda4eclipse will generate the same product as the one in the original project.

For example, whenever a new process definition is modelled, the architect is responsible for publishing it in the projects Web service. Any other project inside the organization might execute a look up query to try to obtain that process definition. If the architect finds the process definition model, it is incorporated within the business domain and might be altered to support specific business services and rules; if the architect does not find any service that matches one’s needs, then a new service model is created and published in the projects Web service.

VII. PRELIMINARY ANALYSIS AND RESULTS

This research tried to prove that the domain-driven development from previously-developed concepts is possible. This concept proof was based on a real-life project conducted at the Brazilian Navy Bureau for Integrated Logistic Support (Núcleo de Apoio Logístico Integrado da Marinha - NALIM).

Currently, NALIM is developing a system to support maintenance logistics of the Brazilian Navy fleet. The system consisted of a set of business services, including the authentication control service. We elected the authentication control service as a prototype for the use of domain sharing, as it was small and was already completed in another software project.

Our goal was to look up the process definition, download the model and generate the initial code for that model.

We expected this initiative would lower the development process of the authentication control by 20% to 30%.

Our code generation rate encompasses from 50% to 60% of the project itself. The MDA framework is limited to that range because we are still researching methods to improve code generation of software views [14 and 15]. The rest of the code involves implementation of specific business rules and enhancement of previously generated code.

This initiative has proved efficient but we know that it can be improved and our goal is to achieve a 70% development by the end of 2008.

The domain-driven prototype consisted of a central storage environment with one Web service. The look up process was direct to the Web Service URL and there was no need to implement the distributed service look up.

NALIM´s domain model consisted of a local service model and the remote authentication model. The following XMI exemplifies the domain model:

```
<?xml version="1.0" encoding="UTF-8"?>
<process-definition xmlns="urn:jbpm.org:jpdl-3.2"
  name="simple">
  <start-state name="start">
    <transition name="to_authentication" to="Authenticate">
    </transition>
  </start-state>
  <state name="Authenticate">
    <event type="task-assign">
      <action name="authService" class="AuthService.uml">
      </action>
     </event>
     <transition to="Order Request" name="to_order_request">
     </transition>
  </state>
  <state name="Order Request">
    <event type="task-assign">
      <action name="orderRequest" class="OrderRequest.uml">
      </action>
    </event>
    <transition to="end" name="to_end"></transition>
  </state>
  <end-state name="end"></end-state>
</process-definition>
```

Code Snippet 2. The processdefinition.xml file representing the concept proof for the NALIM logistic project.

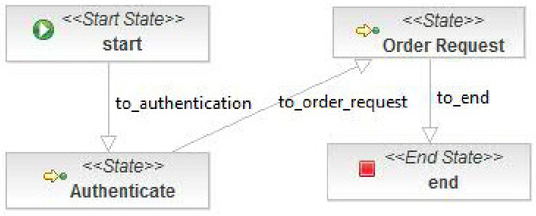

Figure 4. NALIM´s process definition graph.

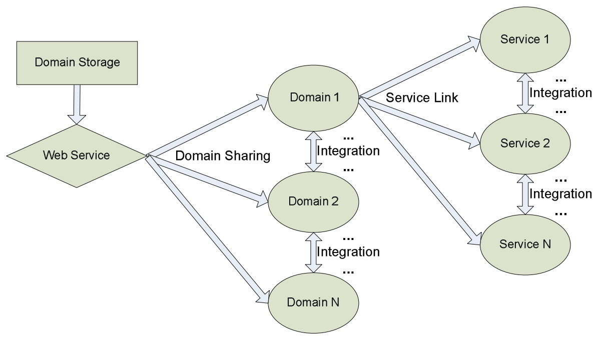

Figure 5. NALIM´s domain sharing architecture.
The prototype architecture used a central remote Web service to store domain models.

Authentication control was designed in a previous project at the Brazilian Ministry of Defence Software Development Laboratory. It took two months from business elicitation rules to code implementation. Our project needed only to verify business rules and authentications rules and use the shared domain model to generate the authentication code. From Business elicitation rules to system implementation we took one month and 15 days approximately, a saving of 25% in development time.

Service	Expected Development Time	Actual Development Time	Development Time Reduction
Authentication Control	2 Months	1 1/2 Month	25%

Table 1. Concept proof results. The use of a shared domain model cut overall development time in the Authentication Control service by 25%.

VIII. Conclusion

This paper presented a different approach for domain-driven development. Through the use of BPM, MDA, SOA and Web Services, we have created a prototype environment in which architects are capable of designing domain models and sharing such information with different projects. Our contribution aimed at creating a MDA environment in which public organizations could solve concurrent development projects with similar or identical business contexts.

Analysis results are still in their early stages but the study proved that domain sharing can improve MDA development using SOA to guarantee coherent system architecture. We believe we have validated the theory, but we are still working to implement a full project development using the above architecture, and also put in production a truly distributed development environment with the cooperation of several project sites.

Finally, the intention was to obtain development gains in scheduling and in cost. According to analysis results, the environment was able to use previously created domain models to give a head start to new projects, and consequently promote the necessary corrections to ensure the code generated is reliable and optimized [9].

References

[1]. Evans, E., "*Domain-Driven Design: Tackling Complexity in the Heart of Software*", Addison Wesley, 2003.
[2]. Evans, E., "*Domain-Driven Design*", Addison-Wesley, 2004.
[3]. "*Model Driven Architecture*", http://www.omg.org/mda.
[4]. Nilsson, J., "*Applying Domain-Driven Design Patterns*", Addison-Wesley, 2006.
[5]. "*AndroMDA, v3.0M3*", http://www.andromda.org/.
[6]. Buschmann, F., Henney, K., Schmidt, D. C., "*Pattern-Oriented Software Architecture - A Pattern Language for Distributed Computing*", John Wiley & Sons, Ltda, 2007.
[7]. Mcgovern, J., Sims, O., Jain, A., Little, M., "*Enterprise Service Oriented Architectures - Concepts, Challenges, Recommendations*", Springer, 2006.
[8]. UML2 Tool, http://www.eclipse.org/modelling/mdt, 2008.
[9]. Rodrigues, G. N., "*A Model Driven Approach for Software System Reliability*", Proceedings of the 26th International Conference on Software Engineering (ICSE'04), IEEE 2004.
[10]. "*Meta Object Facility*", http://www.omg.org/mof.
[11]. "*Eclipse Test & Performance Tools Platform Project*" http://www.eclipse.org/tptp/.
[12]. Frankel D. S., "*Model Driven Architecture – Applying MDA to Enterprise Computing*", OMG Press, Wiley Publications. 2003.
[13]. "*Eclipse Project*". http://www.eclipse.org.
[14]. "*Velocity Project*". http://velocity.apache.org/.
[15]. "*Struts Project*", http://struts.apache.org/.
[16]. "*JBoss Application Server*", http://www.jboss.org/.
[17]. "*Business Process Management Initiative*", http://www.bpmi.org/, 2008.
[18]. "*JBoss JBPM*", http://www.jboss.com/products/jbpm, 2008.
[19]. "*MDA4eclipse Research Project*", www.mda4eclipse.com.br, 2008
[20]. Evans, E., "*Domain-Driven Design: Tackling Complexity in the Heart of Software*", 2003.