

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

Fabio Perez Marzullo¹, Jano M. de Souza¹, José R. Blaschek²

¹ COPPE/Sistemas, Federal University of Rio de Janeiro,
Rio de Janeiro, Brazil. ffpm,jano@cos.ufrj.br.

² FAF, State University of Rio de Janeiro,
Rio de Janeiro, Brazil. blaschek@attglobal.net.

Abstract

It is accepted that domain based development is playing an important role on IT projects today. Following such idea, 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. Our discussion presents a way of sharing business domain models, developed in different project sites, to work together and establish a cooperative development environment. Using a set of development and architectural standards, and modelling theories such as 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.

1. 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.

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.

Both 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 two 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 MDA, SOA and Web Services, this paper presents a research conducted with the purpose of proving that domain-driven approaches can promote a more efficient and rich development environment, which lay down an architecture that is flexible to ongoing changes in the business environment [4].

2. 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 [17], 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 facing 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 MDA and SOA theories?

The following sections try to answer each question by detailing what we have done to create the development environment. However, the overall picture can be perceived as:

1. By creating a generic Service-Oriented Access Bus it becomes easy to integrate different business services.
2. By using Web Services technology it is possible to create a cooperative environment in which different organizations can share architectural standards and conceptual models.
3. By automating code generation different organizations can produce the same software product using the same set of models.

3. 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*.

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.

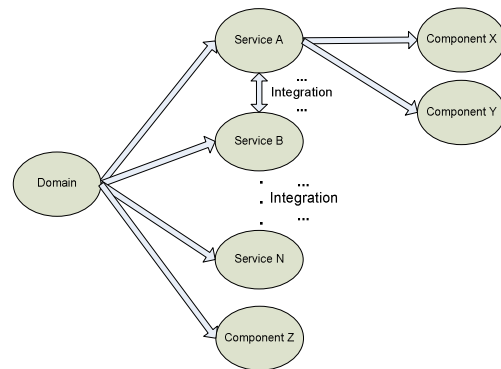


Figure 1. The domain model links to different services and components. Services and components are used to generate software code via the MDA tool. Domain model information is stored in XMI files.

Within this context, we have established a standard Service-Oriented Architecture to model business domains for projects conducted in public organizations. As figure 2 presents, the SOA BUS is used to orchestrate different business services, representing a valuable strategy to integrate older and proprietary platforms. By standardizing the orchestration process it is possible to deliver flexible architectures capable to respond to changes in an affordable manner [18].

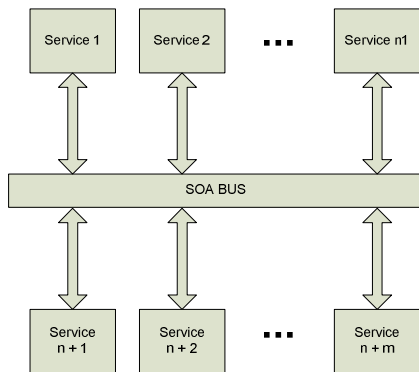


Figure 2. The SOA BUS and service orchestration.

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.

4. 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. For that purpose we followed a five-step design process, in order to enable domain-driven development:

1. *Business 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. *Creating the XMI file.* This stage came up with a modelling style that used the XMI format to design the domain. It was responsible for creating the XMI file containing domain information on services and components:

```
<?xml version="1.0" encoding="UTF-8"?>
<xmi:XMI xmi:version="2.1"
xmlns:xmi="http://schema.omg.org/spec/XMI/2.1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:MZ="http://www.mz-empresarial.com.br/">
<MZ:Domains>
  <MZ:Domain name="DOMAIN" id="ID" >
    <MZ:Description>DOMAIN DESCRIPTION</MZ:Description>
    <MZ:Services>
      <MZ:Service name="SERVICE" id="ID">
        <MZ:Description>SERVICE DESCRIPTION
        </MZ:Description>
        <MZ:Models>
          <MZ:Model name="MODEL NAME" id="ID">
            <MZ:ModelLink file="local_file.xmi"
              url="remote_file_url" id="ID"/>
          </MZ:Model>
          ***
        </MZ:Models>
        <MZ:Components>
          <MZ:Component name="COMPONENT">
            <MZ:Description/>
            </MZ:ComponentLink file="local_file.xmi"
              url="remote_file_url" id="ID"/>
          </MZ:Component>
          ***
        </MZ:Components>
        <MZ:ServiceLink file="service_local_file.xmi"
          url="service_remote_file_url">
        </MZ:ServiceLink>
      </MZ:Service>
    </MZ:Services>
  </MZ:Domain>
</MZ:Domains>
</xmi:XMI>
```

Code Snippet 1. The XMI file representing the domain model.

The XMI file aggregates domain information including local services and component model files as well as URL links to remote files and storage locations.

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.

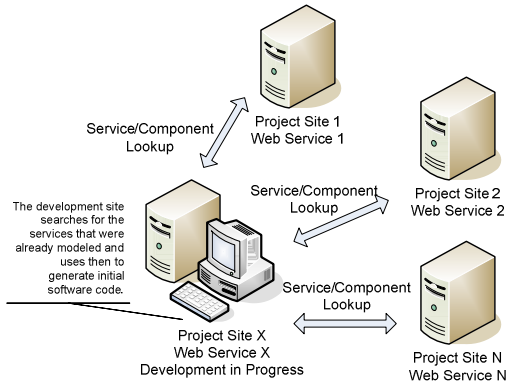


Figure 3. Domain sharing process. The development site seeks services and component models and uses them to generate initial project software code.

5. 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 (XMI files) via WSDL files, running on a Web application server [16].

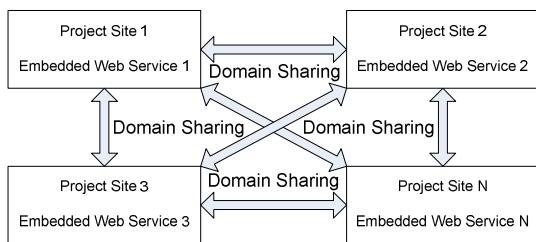


Figure 4. Domain Sharing Architecture.

For example, Project Site 1 models a set of business rules with a UML Case Tool. It then has an automated code generation process identical to every other Project Site, so it creates a set of physical software components using the MDA framework. Project Site 2 has now similar needs and the same business rules to design; what it does is to query all known Project Sites and searches for the business domain models that are more likely to fulfil its needs. It then downloads the models and creates the same software that the first Project Site did.

6. Automated Code Generation

Our MDA environment consists of a set of integrated tools. The main development tool is Eclipse IDE [13]; the Eclipse project contains many other co-related projects for many purposes like testing, modelling, and metrics [11]. The development environment consists of an UML Tool called UML2Tool, available at the Eclipse plug-in home page [8] and by an MDA plug-in developed by our research team at the database laboratory [17].

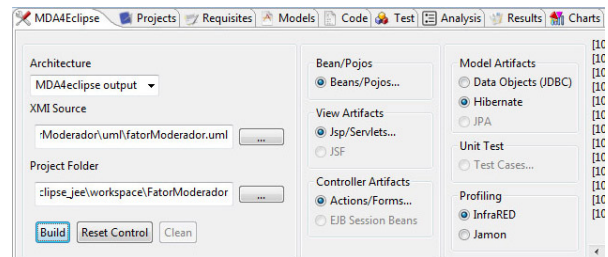


Figure 5. The MDA4Eclipse plug-in.

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 MDA framework will generate the same product as the one in the original project.

For example, whenever a new service 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 service. If the architect finds the service model, it is incorporated within the business domain and might be altered to support specific business 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.

7. 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 service, download the model and generate the initial code for that service.

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"?>
<xmi:XMI xmi:version="2.1"
xmlns:xmi="http://schema.omg.org/spec/XMI/2.1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:MZ="http://www.mz-empresarial.com.br/">
<MZ:Domains>
  <MZ:Domain name="Logistics" id="1" >
    <MZ:Description>A Logistic Domain</MZ:Description>
    <MZ:Services>
      <MZ:Service name="Order Request" id="204">
        <MZ:Description>An Order Request Service
        </MZ:Description>
        <MZ:Models>
          <MZ:Model name="Order Request" id="1023">
            <MZ:ModelLink file="order.xmi" url="none"
            id="2501"/>
          </MZ:Model>
          <MZ:Model name="Authentication" id="10">
            <MZ:ModelLink file="authentication.xmi"
            url="none" id="3104"/>
          </MZ:Model>
        </MZ:Models>
      </MZ:Service>
    </MZ:Services>
  </MZ:Domain>
</MZ:Domains>
</xmi:XMI>
```

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

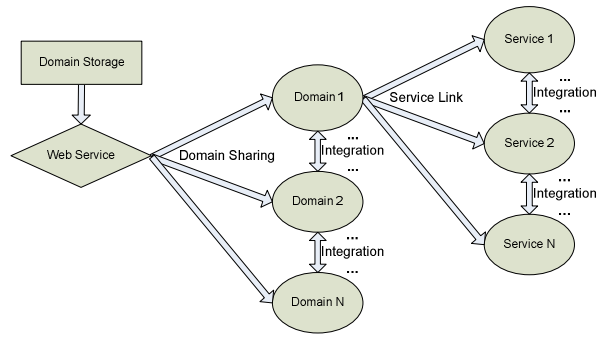


Figure 4. NALIM's domain sharing architecture. The prototype architecture used a central remote Web service to store domain models.

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.

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

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

It is clear that the foregoing results lack more specific analysis to guarantee that our cooperative domain-driven environment can scale to larger systems. However it is possible to state that this research proved that domain-driven development approaches can be used to lower development schedule and budget.

8. Conclusion

This paper presented a different approach for domain-driven development. Through the use of 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].

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] MDA4Eclipse Project, www.mda4eclipse.com.br.
- [18] JBoss Enterprise SOA Platform, <http://www.jboss.com/products/platforms/soa>, 2008.