

# Architectural concerns and use of a model-driven development framework

G.Koutsoukos, L.Andrade, J.L.Fiadeiro, and J.Gouveia

ATX Software SA, Alameda António Sérgio 7, 2795-023 Linda-a-Velha, Portugal  
jose@fiadeiro.org

**Abstract.** We present and discuss the architecture of a UML and code generation based development framework, and the lessons learnt from its use in an industrial project.

## 1. Introduction

The fierce competition that characterizes the New Economy is giving every CIO two major “headaches”: Time and Change. More and more, the success of IT departments and companies is being judged by their ability to build systems according to ever more strict “time-to-market” constraints, and to evolve their end-products with the agility that is required to respond to new business or technology requirements. However, as most of those engaged in the software industry are aware, these two challenges are not easy to meet. The main problem is that, normally, a large part of any software system consists of infrastructural code responsible for dealing with architectural and “environmental” aspects such as communication protocols, security, or error handling, while only a small percentage of the code implements the business logic for which the system was originally conceived.

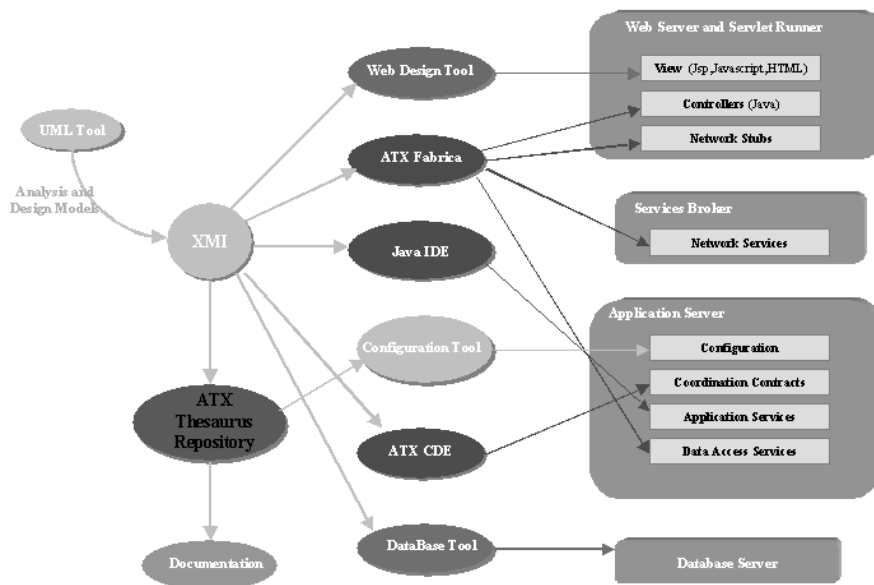
Unfortunately, modeling tools and development environments are not very good in separating architectural and environmental concerns from business considerations. This leads to systems that are very “sensitive” to environmental changes and, at the same time, do not allow for changes performed at the level of analysis/business models to be reflected directly on the implementation layers.

Even worse, this dependency on technical infrastructure requires a high level of expertise from the development teams, which are costly to put together and time consuming to train. We believe that the goals of increasing productivity and, at the same time, maintaining quality and guaranteeing adaptability, can be met by development frameworks that provide methodological and technological support for working directly on the analysis and design models without being dependent on technical and infrastructural decisions. In what follows, we present one such framework that we have developed “in-house” and applied to the development of Web-based financial systems.

## 2. The Framework

The main constituents of the framework that we have developed are shown in the picture below for a typical 4-tier Web distributed architecture following the well-known Model-View-Controller (MVC) logic. This target architecture is shown on the right-hand side of the picture. Although we shall mainly discuss the “logic” of the framework, we must point out that specific decisions taken at the level of the target architecture, e.g. to include a legacy system in the back end, may require additional methodological and tooling support, e.g. for code reengineering.

The system is implemented in Java and consists of a Web server accessible via a browser, an application server running on a different machine, and a database server. To allow access to the system via existing and new clients, calls to the Applications Server are channeled through the “Services Integrator” – a broker that can implement different communication protocols (e.g SOAP, XML).



It is important to notice that, on the Applications Server part, where the business logic is actually concentrated, we enforce a strict separation between the computation, coordination and configuration layers of the system. This separation is based on the methodology that we have developed around the notion of coordination technologies [1], [2], [3] and constitutes one of the distinguishing aspects of our offer. By “coordination technologies” we mean a set of semantic primitives that we have developed over the last years in order to allow for the structure of the business domain to be transposed to the modeling level in terms of basic business entities over which business rules, modeled explicitly as first-class architectural connectors, can be dynamically superposed. Through associated design solutions, we have made it possible for such structures to be directly reflected on the implementation layer so that systems

can be reconfigured dynamically to take into account changes on the business rules with minimal impact on the implementation of the basic business entities. Our experience indicates that such a methodology is absolutely necessary in order to promote the desired levels of architectural compositionality.

Having such an architecture in place, analysts and designers can model the system using any UML tool (e.g Magic Draw, Rational) that supports XMI persistency or generation. XMI is used as a common integration technique among the set of code generators shown in the center of the picture above. A common repository of artifacts, such as the “ATX Thesaurus”, can be used, in option, as a common data model from which the input that is required for the tool generators can be produced. Such a repository is also very useful for storing the software artifacts that are progressively produced, improving management during development.

The code generator tools can be customized according to the target architecture and environment. For instance, given an abstract specification, the “ATX Fabrica” can apply a set of customizable code generation rules to produce the protocol code that is necessary for the communication layer (e.g SOAP, RMI etc). Similarly, a Web Design Tool can generate the necessary front-end pages, a Java Development Environment can import XMI models and generate the application services layer, and so on and so forth. In this way, analysts and designers can concentrate on the core business logic only, using a universal vocabulary (UML-based) to develop their models, keeping away from the infrastructural concerns that tie-up models to technical issues and prevent them to react to changes in the business domain with the agility required by the “Now-Economy”.

### 3. Experiences from using the framework in practice

Based on our experience in a case of a partial instantiation of the target architecture described above, we would like to outline on the table below some of the advantages that we have witnessed and some of the issues that we identified as necessary for putting such a framework successfully in practice. We focus on three major areas, namely the software development process, the technology and the staff resources.

Area	Advantages	Issues
<i>Process</i>	<ul style="list-style-type: none"> <li>– Faster, more uniform and more “industrialized”</li> <li>– Excellent documentation and artifacts management</li> </ul>	There must be a clear view of the process, and the roles played by the various participants, right from the beginning; this view must be communicated and accepted
<i>Technical</i>	<ul style="list-style-type: none"> <li>– Model only once</li> <li>– System closer to business structure</li> <li>– Architectural compositionality</li> <li>– End product is more uniform and easier to maintain</li> </ul>	<ul style="list-style-type: none"> <li>– Good design and analysis techniques and principles</li> <li>– Tool dependency</li> <li>– Very good and reliable UML tool is needed</li> <li>– Tool tuning for different targets</li> <li>– More effort on the analysis/modeling phase</li> <li>– Central Repository is probably necessary</li> </ul>
<i>Personnal</i>	<ul style="list-style-type: none"> <li>– Clearer separation of concerns</li> <li>– Specialization according to skills/domain knowledge</li> </ul>	<ul style="list-style-type: none"> <li>– People involved, especially if an extensive development background, may feel less creative due to less coding and more modeling and tooling work</li> </ul>

## REFERENCES

1. L.F.Andrade and J.L.Fiadeiro. Coordination Technologies for Managing Information System Evolution. In *Proceedings of the 13<sup>th</sup> Conference on Advanced Information System Engineering-CAISE'01*. LNCS 2068, Springer-Verlag 2001, 374--387.
2. Andrade L.F, Fiadeiro J.L, and Wermelinger M. Enforcing business policies through automated reconfiguration. In Feather M, Goedicke M (eds) *Proceedings 16th Int. IEEE Conference on Automated Software Engineering*. IEEE Computer Society Press 2001; 426--429.
3. Gouveia J, Koutsoukos G, Andrade L and Fiadeiro J.L. Tool Support for Coordination-Based Software Evolution. In Pree W(ed) *Technology of Object-Oriented Languages and Systems-TOOLS 38*. IEEE Computer Society Press 2001; 184--196.
4. [www.atxsoftware.com](http://www.atxsoftware.com)