

Business Modeling using an Object-Oriented Framework and Meta-Repository Architecture

Martine Devos, IS Manager, Argo (mdevos@argo.be)

Michel Tilman, Senior Systems Specialist, Unisys Belgium (mtilman@argo.be)

Introduction

The framework, developed at Argo, allows for modeling the business logic of a wide variety of organizations and application domains, making as few assumptions as possible about particular business models. End-users and administrators get a set of tools for defining, customizing, managing and maintaining the business objects pertaining the workings of a particular organization. The functionality of these tools captures the common patterns when dealing with database applications, electronic document management or workflow. The framework's main focus is large administrations which need to cope with several applications sharing a common business model.

The architecture relies on an extensible framework of re-usable components expliciting the common application logic patterns and on a meta-repository containing descriptions of the actual business logic, which is used at run-time by virtually all components of the framework, such as the authorisation rules. This allows applications to be modeled and configured, rather than hard-coded, and to let applications evolve more easily with changing needs or organization, be it at the level of functionality or the business model.

Context

The framework was developed by Argo and Unisys Belgium. Argo is responsible for the overall management of Public Schools (non-denominational) within the Flemish community of Belgium. The organization consists of a central administration (which was the focus of the project) and some 100 semi-autonomous local boards and 730 schools (which are to be connected to the central system through the Internet). The generic requirements of the Argo project are fairly typical for many large administrations. Most important amongst these was the need for flexibility, as the organization was in a state of major flux at the start of the project.

Design goals of the framework

The framework aims to combine the high-level modeling power of, say, the better CASE-tools with the open-ended nature of an object-oriented framework. At this, it relies on the following observations:

- whereas the business logic is particular to each organisation (or a well-defined vertical market segment), the end-user or administration functionality (such as data-entry, querying, reporting, document management and in / out baskets) is all the more generic
- extensions or enhancements to the functionality can often be made into re-usable assets, independent of the actual business model
- the business model should be allowed to evolve, with limited impact on the application logic and vice versa.

This has resulted into the following approach for the framework, which:

- allows to formalize business structures, data, relationships, processes and rules particular to a specific organization (the *datamodel* of the organization) by modeling and configuration
- uses a meta-repository to store this datamodel
- separates this formalized description of the datamodel from the application logic
- offers a generic end-user application as well as administration, authorisation and configuration tools for managing both structured and unstructured data as well as workflow, requiring only the business model to get the system up and running

The benefits are manifold:

- Using a meta-repository in combination with a ready-built generic application containing the necessary functionality (see below) bridges the gap between development team and end-users. This results in an iterative and incremental style of working, and supports rapid application deployment.
- In contrast to most CASE-tool approaches, the datamodel is not used to *generate* an end-user application, but is *consulted* in a dynamic way by generic end-user application, tools and framework components, which allows for increased flexibility (such as enhancing the functionality of the end-user application without having to modify the existing business model or vice versa) and maintainability.

- As all components, tools and end-user application, as well as structure and functionality of the meta-repository are part of the same framework environment, they can evolve over time within the framework as needed. This allows, for instance, to add new components or functionality to be incorporated in the framework, and to cope with future technologies and media.

Another important design goal of the framework was to allow a kind of *bootstrapping*, whereby administration and configuration tools can gradually be expressed in terms of the framework itself.

Functionality overview

The framework offers standard a generic end-user application and several administration and configuration tools, as well as a generic process manager.

The generic end-user application allows users to:

- login to / logout from the system
- select a login role and the applications he / she wants to use (which may depend on the selected login-role)
- define and execute queries and view the results in overview lists
- edit / view / print / export / database entities using index cards
- access the thesaurus browser
- manage electronic documents (scan / import / edit / annotate / OCR), version lines and alternative representations
- define personalized work environments through the use of layouts, queries and rules
- view incoming and outgoing task assignments by means of in / out baskets
- set various preference settings
- access help functions.

The administration tools include the following components:

- the datamodel editor allows to edit an existing or new datamodel, describing the business model
- the application editor allows the definition of applications, which are coherent views on the datamodel
- the authorisation editor manages the definition of the authorisation rules
- the rule editor allows to define both user- and system-triggered rules, for instance for setting default values, checking application specific validation constraints or creating new task assignments
- the layout editors allow for edition of list and index card layouts, both for individual users and groups of users
- import /export tools allow for import of entities and relationships from / to a text file
- the thesaurus browser allows the use of the thesaurus without the main end-user application
- the database generation tool allows administrators to generate the necessary database structures corresponding to the datamodel.

A generic process manager component allows to schedule various background tasks, which can be defined via rules. Current uses are the notification mechanism, the internet E-mail gateway and the document copy manager, which migrates documents to or from the jukebox.

Technical specifications

The framework is currently implemented in a Windows / Novell / Oracle environment and is developed in VisualWorks\Smalltalk.

Most of the framework components are developed in Smalltalk, apart from some system-specific components, which rely on standard tools:

- scanning and viewing of electronic documents is managed by means of a Kofax image rendering and scanning library, which is transparently integrated into the Smalltalk environment
- the OCR-module is built on top of the Calera module
- E-mail uses the sockets communication library and offers support for SMTP and POP-protocols
- the Mires engine supports full-text-indexing with proximity search
- the jukebox technology (HP) is driven by a document copy and migration process manager.

The environment at Argo contains some 300 client PC's, with future access to the system foreseen for about 730 schools through the use of Internet.

Demo

The demo at OOPSLA'96 highlighted the following aspects:

- the use of the meta-repository and how it allows to separate the business model from the end-user applications and tools
- how the authorisation and configuration tools allow to personalize the work environments for the individual end-user within the context of a central business model
- the essential steps in building an end-user application
- how relationships can be fully exploited by the end-user
- the concepts of the query editor, authorisation and business-rules, and how they can be used to make the system react more intelligently, as they do not rely on a particular business model and can be context- and contents-sensitive
- the bootstrapping process, in particular with respect to the system entities and business rules editor, and how it benefits from the use of reflection techniques.