Application Development using a Meta-repository based Framework

Michel Tilman, System Architect (mtilman@argo.be)

Asbtract

The Argo framework is an object-oriented framework developed in VisualWorks\Smalltalk for modeling the business logic of a wide variety of organizations and application domains, making as few assumptions as possible about particular business models. It offers a set of generic tools for defining, customizing, managing and maintaining the business objects pertaining the workings of a particular organization, be it in the context of database applications, electronic document management or workflow. Its main business areas are large administrations which need to cope with several applications sharing a common business model.

The key features of the architecture are its open-ended nature, relying on an extensible framework of re-usable components containing the generic application logic, and on a meta-repository containing descriptions of the actual business model

 The repository captures both formal and informal knowledge of the business model and of personal and shared work practices. High-level end-user and administration tools consult the repository at run-time, querying the meta-model for dynamic behavior. Changes to the repository can be made at-runtime and are immediately available to clients

End-user applications can be developed interactively and incrementally through modeling and coinfiguration: they are not hard-coded, neither are they generated. Instead, we build increasingly complete specifications of end-user applications that can be executed immediately. This way, applications can evolve more easily with changing needs or organization, be it at the level of functionality or the business model

Context

Argo is responsible for the overall management of Public Schools (non-denominational) within the Flemish community of Belgium. Argo consists of a central administration (which was the focus of the project) and some 350 semi-autonomous local boards and 730 schools. The generic requirements of the Argo project are 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, 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 maket 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 appplication 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 business model of the organization) by modeling and configuration
- uses a meta-repository to capture both formalized knowlegde (e.g. the object model) and less formal knowlegde (e.g. custom scripts) of the business model
- separates this description of the business model from the application logic
- offers generic end-user, 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 manyfold:

- In contrast to most CASE-tool approaches, the business model is not used to *generate* an end-user application, neither do we code (apart from some custom scripting) applications. Rather, the object

model is *consulted* in a dynamic way by generic end-user and administration 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.

- Using a meta-repository in combination with ready-built generic end-user and administration tools containing the necessary functionality bridges the gap between specification, development and use of the applications. We do not use or need throw-away prototypes. Instead, we build increasingly complete specifications of end-user applications. These specifications are available for immediate execution. In a sense, the framework and its built-in tools act like an interpreter for the specifications.

- 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 several generic end-user, administration and configuration tools, as well as a generic process manager.

The generic end-user tools allow 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 event-condition-action 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 object model editor allows to edit an existing or new object model, constraints and behaviour
- the application editor allows the definition of applications, which are coherent views on the object model
- the authorisation editor manages the definition of the authorisation rules
- the graphical workflow editor allows to define workflow templates and visualize and edit running workflow processes
- the action rule editor allows to define both user- and system-triggered rules, which add application-specific functionality and semantics, for instance for setting default values, checking 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
- plug-in import /export tools allow for import of entities and relationships from / to a text file or other formats
- the database generation tool allows administrators to generate the necessary database structures corresponding to the object model.

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 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:

- external applications (typically office tools) using DDE
- 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 suport 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 will highlight 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
- the concepts of the query editor, workflow processes, authorisation and action 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
- show the bootstrapping process, in particular with respect to the meta-model, system objects and action rules editor, and how it benefits from the use of reflection techniques.

A technical overview will accompany the demo.

The presentation will focus on concepts and technical issues of the framework.

References

[Dev96a] Martine Devos and Michel Tilman, Design and Implementation of a Business Modeling Framework using Smalltalk, Object Technology'96, 1996

[Dev96b] Martine Devos and Michel Tilman, Business Modeling using an Object-Oriented Framework and Meta-Repository Architecture, OOPSLA'96 Demonstrations, 1996

[Dev96c] Martine Devos and Michel Tilman, Object-Orientation and Evolutionary Software Engineering, Position paper for the OOPSLA'96 Workshop on Object-Oriented Software Evolution and Reengineering, 1996