

Design and implementation of a business modeling framework using Smalltalk

*Martine Devos, IS Manager ARGO (mdevos@gate.argo.be)
Michel Tilman, Systems Specialist UNISYS (mtilman@gate.argo.be)*

Abstract

The first part of this paper presents context, rationale and design goals for an integrated workflow environment implemented at ARGO, using object-oriented technology. Subsequently, the paper highlights particular issues involved in implementing the project, and concludes with a summary of some of our main observations.

Keywords: *Business Modeling, EDM, Framework, Prototyping, Relational Database, Reuse, Smalltalk, Workflow.*

Context

Situational factors

The ARGO organization is responsible for the overall management of public schools within the Flemish community in Belgium. ARGO consists of a central administration, managing and coordinating the activities of several hundred semi-autonomous local councils. In order to better achieve its mission objectives through the use of computer support, a five-year project was defined.

The first year was to see the elaboration of the overall infrastructure, as well as the introduction of some form of office automation throughout the central administration, by virtue of tools such as word processing, spreadsheets and E-mail, along simple database applications (using the Oracle database, which was already present at the start of the project). The infrastructure consists of about 300 PC's, which are connected to central database and file servers, and are distributed over three locations.

The next two years -with the end due within 6 months from now- focus on introducing electronic document and workflow management by means of carefully chosen pilot projects, alongside more complex (traditional) database applications, within the central administration

Implementing EDM and workflow management on a larger scale within the organization as well as connecting up schools and local councils to the central administration via Internet are to be the main driving forces during the remainder of the project.

Relatively unburdened by existing legacy systems, the ARGO opted at the onset of the project for a mix of well-known relational database technology (Oracle) and advanced object-oriented technology (VisualWorks\Smalltalk), the former primarily being chosen because it was already present in the organization. Furthermore, object-oriented databases were still considered too immature (certainly from commercial perspective) at the time. The Smalltalk development environment, on the other hand, was deemed necessary to handle the intricacies and complexities of modeling enterprise-wide business automation in a generic and open-ended way.

External factors

Two external events heavily influenced overall the objectives and implementation of the project.

For one, due to political decisions, ARGO was to undergo some rather drastic reorganizations, as well as major changes regarding their main objectives, in the course of the project, though the actual outcome was largely unknown at the onset. As such, the system to be implemented had to provide ample scope for modeling different kinds of organization and workflow process structures. Additionally, in order to allow

for more flexible adaptation whenever the models needed to be re-designed, the concrete implementations had to forego hard-coded assumptions of a particular model as much as possible.

Secondly, as originally planned, implementing the actual end-user applications in the Smalltalk environment was to rely heavily on a set of EDM-libraries well suited to the customer's requirements. Seven months after the start of the project, however, the company developing those libraries went out of business quite suddenly, leaving no practical means to reuse the existing software. A decision was taken, thereafter, to implement the major functionalities offered by said libraries in Smalltalk as well. In this process, the applications under development were to be re-designed as well, due to the opportunities offered by using a more uniform approach and having tighter control over specifications of the basic EDM-functionalities. Despite this dramatic turn of events, the project implementation is, by now, reasonably well on schedule, and is expected to be finished quite close to the original deadline.

Design

Rationale

In order to accommodate the requirements set by a changing organizational environment, the main goals of the system were conceived to be as follows:

- The system had to consist of an open-ended, generic framework, suitable for modeling different kinds of organizational, data and workflow process structures.
- The end-user applications were to be derived as much as possible from this framework, by modeling and configuration, customization being used to capture those specific requirements which make all the difference to the end-user, and give him or her real added value.
- Since the system had to be easily adaptable to changing needs, as few assumptions as possible about details of structuring relevant information had to be hard-coded, with knowledge about these structures being dynamically available from a central repository. As a result, more 'intelligence', rather than more 'automation', can and should be brought to the system.

Not surprisingly, these requirements fit in with following observations quite well:

- Different organizations have different needs, implying the need for both genericity and extensibility.
- Most EDM / workflow systems focus too strictly on managing and routing electronic documents, with little or no explicit support to handle richer modeling in order to capture the essence of the domain analysis. However, modeling what the important structures, relationships, rules and constraints are, should take precedence, as they define the building stones for the entire system.
- Many systems, be it CASE-tools or development environments, suffer from a lack of uniformity and / or integration of the various components, causing impedance mismatches when going from one environment to another. Often, this leads to badly integrated designs, where the programmer has to rely on all sorts of escape mechanisms to get the system up and running. In the longer term, this lack of uniformity severely hampers both maintainability and adaptability.

These observations led to the following design goals.

Design goals

A central datamodel contains a description of the relevant entities, relationships, constraints and rules necessary for modeling the organization, its data and processes. These include:

- system entities, necessary for the proper workings of the system (such as thesaurus components, the structure of electronic documents, storage strategies and users)
- the model of the organization (such as functional structures, roles, responsibilities and authorisations, task and procedure descriptions, policies and availability of resources)
- the model of the data (structured information as well as electronic documents, including relationships and constraints)
- the model of workflow procedures (such as task types, default actions and overall process structures)
- higher-level system entities which actually use all this meta-information (such as queries, authorisation- and constraint-rules).

Generic tools use this meta-information, as well as the actual information, in order to cater for dynamic (run-time) generation and / or modification of, for instance:

- application views
- data entry forms
- query views
- list views
- layouts
- authorisation-, constraint- and default rules
- process definition rules.

Using these tools, end-user applications can be configured quite easily.

Example 1

By capturing relevant knowledge into the central model, more useful intelligence can be included in the framework, as well as in the end-user applications. For instance, workflow procedures which follow the traditional hierarchy, can be defined to follow the hierarchy as specified in the organizational model, rather than having that particular way to be redefined for every similar procedure.

Example 2

In order to go from one business model (e.g. one based on a role model) to another (e.g. one based on a user / user group model), one has to redefine the model of the organization, and replace authorisation-rules based on roles by equivalent rules based on user groups, all this by modeling and configuration.

Obviously, going from one enterprise model to another may sometimes necessitate extra functionalities, or the elaboration of a tool better suited for handling a particular case, but most often this can be managed in a modular way by adding extra classes or refining existing ones. As more experience will be gained, redesigning parts of the existing framework in order to enhance its applicability will become a major issue.

Implementation issues

Development tools

The framework is developed completely in VisualWorks\Smalltalk, with the exception of specific components (available via external modules) such as access to the Oracle DBMS, and image-rendering, FTI and OCR libraries.

Halfway through the project, the Envy development environment was installed, in order to better support multi-user development and keep track of versions and configurations.

We feel the following functionalities of the programming environment to be very important assets:

- the uniformity of language (everything is an object, including classes, which means that meta-information can be consulted or generated at run-time) and environment (e.g. most user interface components and tools are written in the language and environment themselves, implying they are available to the programmer as well as to other tools) alike, allowing for enhanced genericity and reusability
- the highly interactive and dynamic language and environment, supporting an iterative and prototypical style of development
- a rich and mature class library and framework to start with.

Actually, the approach taken in the design of the framework, somewhat parallels these reflections, as is evident from the framework / tools based setting and from the emphasis on a central business (meta)model.

Project structure

Developing the actual framework and tools is managed by a relatively small team (i.e. 7 people, with, on the average, 4-5 people doing actual coding). Most of these people are regularly involved in other activities of the project as well, going from analysis and design to the steering committee, allowing for increased awareness of the customer's real problems. Additionally, 3 other people participate full-time in the analysis, configuration, documentation and training parts of the project.

At the customer's side, depending on the application, representative users, key-users and project-leaders, as well as members of the IS Department are involved. Members of the development team and IS

Department meet on a regular basis with all project leaders. This allows information to be passed across the boundaries of individual projects, making users and developers more aware of the general context of the project.

The overall project implementation is managed on a daily basis by the IS Manager, and by the Steering Committee on a regular basis.

Project implementation

Designing and implementing end-user applications relies on an iterative approach where analysis, design and implementation partially overlap and interact more intensively, with shorter cycles. Furthermore, successive prototypes are presented to and tested by the end-users, each additional prototype refining or extending existing functionalities.

Observations

In the course of the project, the following observations have made lasting impressions:

- Given the scope of the project and the background of the average ARGO computer-literacy prior to the start of the project, it should come as no surprise that training end-users as well as the IS Department presents a major challenge. At times, we felt not enough resources were spent to assist the users on a sufficiently interactive basis.
- In general, the end-users are very cooperative, going through their normal business routines alongside the -sometimes- tedious testing of successive prototypes.
- The IS Department itself is very much understaffed. As such, it is all the more a pity that some members of the department, while having sufficient qualifications to follow the technical aspects of the implementation in more detail, acted too unresponsive to the introduction of this new technology. Furthermore, not all resources available in the department were felt to be really up to the job of coming to grips with the sophistication of this system. Overall, these results are not really surprising, as they tend to agree with the findings of several surveys already conducted in this field.
- Some members of the development team were not acquainted with the Smalltalk environment prior to the start of the project, while others joined the team rather late. Clearly, the VisualWorks/Smalltalk environment is a serious *programming* tool with a rather large library, not a CASE tool or something similar. As such, and although the language itself is indeed very simple, people who do not *want* to make their way into the environment, will quite probably never feel at home with the tool. On the other hand, people *can* become productive in a relatively short time, if they set their minds to it. At this, most people follow the pattern of needing a few months to come to grips with language and elementary knowledge of the environment, followed by a period (of several months) trying to get sufficiently acquainted with the library and -often- a new way of working, in order to feel really at home. After this stage, productivity increases at a fast rate. That said, people with the right background and mind-set can become more productive in a much shorter time-span, even in as little as 5-6 weeks.
- While the development team consists of highly qualified people, we feel the diverse and complementary background of the team members to offer added value in complex projects such as these.
- Implementation-wise, the main problem resulted from having to start all over again after seven months. Quite clearly, adding extra resources to the team does not help solve the problem that all of a sudden too many things had to happen in parallel. At times, this has led to a loss of time, as modules or designs were being used too early. Furthermore, several design options have been identified which could lead to a better implementation, with more scope for reuse, but which had to be solved in a more pragmatic manner, because time was too short for getting enough experience to build a more flexible design.
- In part due to the timing problem, and in part to the goals and scope of the project (for instance, the development of a new framework and some people new to the Smalltalk environment), it was often difficult to accurately estimate the necessary resources to be spent on a particular part of the project. At times, this led to obvious miscalculations. Quite clearly, more experience is needed here. However, once a suitably stable and rich framework will be established and some more experience in configuring applications will have been acquired, planning should become easier. In this context, contemporary

research on software metrics for object-oriented technology is still too far removed to be really useful.

- Having looked at existing OOA / OOD methodologies and tools, we -and other people as well- feel them to be severely lacking in several aspects. Hence, currently, we rely on a mix of proven and new ideas, in combination with some common sense.