# PSI

## From custom developed application to domain specific framework

*Arlette Vercammen, Wilfried Verachtert*
OO Partners bvba
Otto de Mentockplein 19 1853 Strombeek-Bever (Belgium)
Tel: ++32 2 263 04 00 Fax: ++32 2 267 71 51
e-mail: {arlette | wilfried @oopartners.com }

*This paper describes the experience we had at OO Partners with the design and implementation of PSI, a custom developed planning application for a TV-station, and its evolution into a complete domain specific framework that has been successfully installed at several TV-stations in Europe.*

*We report on our experience with object technology, in particular with Smalltalk, and on the barriers we encountered in designing and implementing the PSI framework. Furthermore, we expound on how we have organised ourselves to increase the productivity of our team and to obtain higher quality software that is more adaptable and reusable.*

*Due to our project approach, OO Partners has built up solid domain knowledge, which is the key to our success with PSI. Gathering this domain knowledge constitutes the major part of the investment. Frameworks are essential to leverage this investment. This experience also applies to other vertical markets.*

### Background

At its foundation in 1987, VTM was the first Flemish commercial TV-station. The mission of VTM was —and still is— to be the preferred family broadcaster. Despite its youth, the company has been profitable since its first year of operation, which is unique in the TV-market.

From the start, VTM's IT department was confronted with two major concerns that were of direct strategic importance to the company. First, IT should solely focus on the core business of VTM, namely television broadcasting. Secondly, the IT-department should be capable to evolve in parallel with the business of VTM. Reacting to rapidly evolving user specifications, delivering projects on time and ensuring a short project cycle are crucial for being successful in the broadcast business.

The people of the IT department were attracted by the promises made by object technology. However, at that time —1991— choosing for object technology, and in particular choosing for Smalltalk[1], was like taking a step in the dark. Nevertheless, thanks to the dynamic nature of the young company, the absence of legacy systems and the presence of avid "object supporters" occupying key positions within the company, VTM became an early adopter of Smalltalk. The first pilot projects, using Smalltalk/V, were started with the aid of an external software house. Up until now, nobody at VTM has regretted this strategic decision.

The success of the pilot projects soon led to the decision to automate the entire planning process with Smalltalk. At that time, OO Partners was founded and received a long-term contract to implement the entire planning application. Several

[1] VTM preferred Smalltalk over C++ because of its pure object-oriented nature and believed that the learning curve for C++ was too long for non-C programmers.

employees that were involved in the pilot projects and the analysis conducted earlier at VTM joined the company.

During the first year, OO Partners mainly concentrated on the project at VTM. Gradually, OO Partners became one of the leading Belgian specialists in providing software solutions based on object technology. We guide software developers through the acquisition process of object technology and assist managers in understanding the applicability of OO to their business. The role of OO Partners is to help its customers in building higher quality, reusable and adaptable software systems.

# The PSI Project

## *Description, Goals and Requirements*

The project goal was to create an application to support the planning of TV-programs. More specific, the application should integrate all the aspects of planning, ranging from seasonal planning to daily planning, as well as video tape management, program information management and downloading information to specific transmission hardware. In other words, the application should be accessible by different kinds of users basically manipulating the same data but from a different point of view. Another requirement was that the application should be as close as possible to the planning processes used at VTM. At that time most of the planning was still done with the aid of pencil and paper. The application should improve the quality of the work —e.g. automate the work flow, reduce administrative overhead— and increase productivity, but not influence the basic procedures that were already used for planning. Last but not least, since the application was intended for use by creative "television" people, special devotion should be given to the design of the user interface to improve the ease-of-use.

The project started early 1993 with a young team of 6 software engineers. Our first prototype was immediately taken into production by the project sponsor who played an important role in motivating other users. Since the initial release, several updates were done, mainly to add functionality. The final version was taken into production end 1994.

## *Maxims*

Already from the start, three basic principles were recognised and constantly kept in mind during the project:

- **Maxim 1: Look at the whole picture:** This maxim originated from our initial attempts to automate a part of the work flow of VTM in which planning was not the central theme. It soon became apparent that this was not the right way to do the job and that a global approach was necessary.
- **Maxim 2: Partial deliveries**: This maxim enforced users to give us feedback so that we felt comfortable that we were building the right system. Initially, this maxim was also intended to gain more user confidence in the new type of applications, although this became less important as the project evolved.
- **Maxim 3: Extreme attention to usability**: This maxim was introduced because it was crucial that the users could identify themselves with the new application, in other words the users expected a lot from this application concerning user friendliness.

## *Analysis & Design*

The analysis of the project was performed using a traditional approach —interviews— because at that time object-oriented methodologies for capturing user requirements were not wide-spread and little experience was available about their applicability in such a large project. Today our choice would definitely go to methodologies that capture the user requirements in an object-oriented way such as Object Behaviour Analysis.

During the analysis, which took about six months for 3 persons, the software engineers enlisted on the project became themselves experts in broadcast planning. They closely worked together with the domain experts, observed how schedules were realised, what procedures were used, and how the information flowed through the organisation. At the end of the period this resulted in a huge catalogue of relevant information. As will become apparent later, the acquisition of domain knowledge was the key for success of the PSI-framework, but that was something we didn't realise at that time.

Once the basic requirements for the planning business were known, CRC and OMT were used for specifying the domain objects. The actual design took place in the Smalltalk environment.

### *Process Model*

For actually implementing the application we chose VisualWorks and decided to partition the project in several sub-projects —e.g. program management, video tape management, strategic planning, tactical planning, operational planning and trailer management.

The process model we decided to use was focused on the following issues:

- **Partial Deliveries**. In order to adhere to maxim 2, a partial delivery approach was used to involve key users early in the development process to improve the quality and functionality of the system. At the same time, this approach allowed OO Partners to better understand the business domain. However, in the planning, the overhead of partial deliveries was underestimated. Developers, for example, had to provide significant user support for each delivery, thereby slowing down the overall development of the application. Another example of where the overhead became significant was when we delivered a preliminary version of the seasonal planner. Instead of storing data in a database, this version stored its information directly on a file, using a VisualWorks BOSS-file. The user was so enthusiastic about this prototype that he actually used it for his planning job. This meant, of course, that in the next release, where database access was supported, conversion tools had to be supplied, something that was not foreseen in the initial planning.
- **Incremental addition of features**. The biggest problem here was finding the right balance between user expectations and the global scope of the project (Maxim 1). At several occasions users had to be restrained from adding too specific features. Without our profound domain knowledge of the entire planning process, detecting such too specific features would have been almost impossible. Having a steering committee[2] was a necessity to watch over the global project goals. Its responsibility was to approve additional or altered requirements and to adjust the initial project schedule and budget.
- **Developing subsystems in parallel**. Each sub team was assigned complete authority to design and develop one of the subsystems but nevertheless shared the same domain "vocabulary". During the project the members of the sub teams were subject to rotation. This helped create an egoless development situation. On a regular basis design reviews took place to watch over the quality and discuss the interfaces between the different subsystems.
The initial expectations we had about reuse amongst the sub teams were not completely fulfilled. This was mainly due to the absence of a reuse manager to

[2] The steering committee consisted of the financial director, IT manager, project manager and a user responsible.

plan and control the reuse process. As a consequence the sub teams sometimes implemented the same features —e.g. dirty bit management, button enabling/disabling management— that were not generalised at that time.

## The PSI Framework

Even before the final version of PSI was installed at VTM, another Belgian and several European TV-stations showed interest in the product. Marketing research has revealed that few software solutions are available on the market for scheduling and planning TV-programs and that most of the existing off-the-shelf software is based on older technologies. They lack sophisticated graphical interfaces, don't reflect the dynamic work flow and are not easy to adapt to client-specific needs within limited time and budget. Especially the latter shortcoming is considered a serious problem in the rapidly evolving TV-market. This observation motivated us to build a domain specific framework that is easy customisable.

### *Modifications to PSI*

Our first major modification of PSI was done at VTM for the opening of their second channel. Because the board of directors wanted to keep their ideas for the second channel secret for their direct competitors, only very few people were informed. The official announcement was made only one month before the start of the new channel. Even the planning and IT departments didn’t know of the plans in advance. Needless to say that there was some panic among the users. In one month the existing single-channel planning application had to be transformed into a multi-channel planning application[3]. This had not only implications on the planning strategies used —e.g. ensuring that a film is planned only on one of the channels— but also on some of the domain components —e.g. transmission dates had to be associated with a channel. Despite the complexity of the task, the modifications could be implemented in time!

In the mean time, PSI has been successfully installed at two other broadcast stations, referred to as case 1 and case 2 in the appendix. In case 1 only twenty percent of the classes had to be adapted. These adaptations mostly consisted of adding specific features. In the other case, the existing work flow procedures had to be revised, but the underlying domain objects could be reused. These two extra cases allowed iteration which was very important for OO Partners to improve the quality of the framework.

In all cases, the customers felt like having a custom-made application, tailored towards their specific needs and free from useless and confusing features. Of-the-shelf solutions never could have resulted in the same degree of satisfaction. Because of the short development time, the budget remained within acceptable boundaries. This was only possible thanks to the use of object technology, and in particular the use of framework technology.

### *Our Experience with Building the Framework*

The increasing interest in PSI is for our company a big challenge on the field of software engineering. How to keep the PSI framework manageable —i.e. maintain, support and add new features to it— and in the same time respond to the demands for adapting PSI towards specific needs?

From the start we consequently applied object technology because of its good reputation concerning reusability and adaptability. Practical experience shows that object technology can indeed make significant contributions on these subjects. Our practical experience shows that a company can only take full advantage of OO, and

[3] This is a good example of what it means for the IT department to be able to evolve in parallel with the business.

be successful with it, when it applies OO consequently in every aspect of the process —i.e. business processing, user requirements gathering, analysis, design and implementation. Secondly, we observe that it is often hard to understand what somebody else has developed, not to mention reuse it. During our years of experience, we learned that reusability does not come for free. Merely putting the label OO on an application is not enough. To build large-scale reusable and adaptable systems, a new approach towards software engineering is indispensable to effectively take full advantage of the technology. Our theory is strengthened by what is happening in the world of OO software engineering today —i.e. frameworks and design patterns.

The need to make PSI adaptable requires the definition of a clear architecture expressed in terms that are understandable to experts in the domain. It is the materialisation of our years of experience in the field and includes information about the work flow, the kinds of actions planners can perform and what they expect from other users of the application. In our case the architecture is expressed as a framework consisting of abstract classes, contracts between classes, and numerous hot spots that serve as hooks for future installations of the framework.

A problem we have encountered is the degradation of the architectural design during iterative implementation phases. Often the classes and methods that make up the real design of an architecture are lost in the tangle of implementation classes and methods. It is a hard job to derive the design of an application by merely looking at the implementation code. Documentation and code conventions can help but practical experience reveals their volatile nature. Solving these problems is nearly impossible under the pressure of customer deadlines. Therefore a project was set up to improve the quality of the PSI framework, and to apply existing techniques to cope with this architectural degradation. This project is in co-operation with a university partner (Free University of Brussels) and is sponsored by the government. This project will finish mid 1997.

Building upon the experience with PSI and because of the strategic importance of reuse, OO Partners has founded its internal Reuse Competence Centre (RCC). It specialises in all the aspects of reuse. Having PSI installed at various sites in Europe, gives OO Partners the unique opportunity to build solid experience with respect to reuse, adaptability and flexibility. On the one hand this experience is used to further improve the PSI framework, on the other hand this experience is invaluable in the process of guiding customers with respect to organising reuse, designing and managing reusable components, and building business frameworks.

## Conclusions

In our opinion, the determining factors that turned the first OO-project at VTM into a successful one are:

- Use of iteration —incremental addition of features and partial deliveries— as a means to stabilise the domain components. Object technology supports the use of iteration.
- The incremental approach, together with the presence of a project sponsor resulted in more user confidence. The users were gradually exposed to the new application.
- Combination of available domain knowledge and OO expertise within the project team.

Carefully planning the overhead of partial deliveries in advance is an important lesson we learned from our experience.

Thanks to our rather unique project approach, OO Partners is a domain expert in the field of TV-planning. The availability of domain knowledge within our company largely contributed to the success of the framework. The difficulties that were encountered in capturing this domain information into the PSI-framework were discussed. One key problem that must be solved is the maintenance of a clear architectural model under deadline pressure.

As a general conclusion from our own experience, we want to stress that framework technology is of strategic importance for organisations focusing on software for vertical markets. Frameworks that capture the domain knowledge reduce the development and maintenance costs and give companies a competitive edge.

# Appendix: PSI Expressed in Numbers

Below some project numbers can be found. First, our efforts for the VTM-project in man-days and deviations from the original plan are shown. Next, we compare the efforts spent on the different projects and numbers about applications classes are listed.

## *First Project: VTM*

- **Global Overview**

| | Initial | Actual | Deviation |
|---|---|---|---|
| Requirem. | 360 | 360 | 0% |
| OOA/D | 100 | 100 | 0% |
| Coding + installation + support | 1145 | 1666 | 45% |
| Project management | 115 | 144 | 25% |
| **Total** | **1720** | **2270** | **32%** |

- **Planning modules**

| | Initial | Actual | Deviation |
|---|---|---|---|
| Requirem. | 60 | 72 | 20% |
| Design | 60 | 80 | 33% |
| Prototyping | 80 | 95 | 19% |
| Coding | 450 | 634 | 41% |
| Testing + installation | 250 | 318 | 27% |
| Support | 0 | 65 | - |
| **Total** | **900** | **1264** | **40%** |

- **Program management**

| | Initial | Actual | Deviation |
|---|---|---|---|
| Requirem. | 20 | 18 | -10% |
| Design | 20 | 22 | 10% |
| Prototyping | 10 | 8 | -20% |
| Coding | 30 | 42 | 40% |
| Testing + installation | 15 | 19 | 27% |
| Support | 0 | 5 | - |
| **Total** | **95** | **114** | **20%** |

- **Video tape management**

| | Initial | Actual | Deviation |
|---|---|---|---|
| Requirem. | 20 | 22 | 10% |
| Design | 20 | 42 | 110% |
| Prototyping | 10 | 10 | 0% |
| Coding | 30 | 78 | 160% |
| Testing + installation | 15 | 32 | 113% |
| Support | 0 | 18 | - |
| **Total** | **95** | **202** | 113% |

- **Trailer management**

| | Initial | Actual | Deviation |
|---|---|---|---|
| Requirem. | 10 | 11 | 10% |
| Design | 10 | 13 | 30% |
| Prototyping | 5 | 2 | -60% |
| Coding | 20 | 41 | 105% |
| Testing + installation | 10 | 15 | 50% |
| Support | 0 | 4 | - |
| **Total** | **55** | **86** | **56%** |

Due to our incremental approach, we finally added more functionality to the system than initially was foreseen. Given the latter, we can conclude that most of the deadlines were reached in time. Video tape management is an exception to this rule. It was one of the first experiences we had with users asking too specific features. To solve the problem we had to put extra effort in managing the situation so that we found the right balance with the global scope of the project.

During the year following the final installation, about 5% of the original effort was spent in adapting the application to new needs.

### *Other Cases with Same Complexity*

- **Case 1**

| | VTM | Actual | Deviation |
|---|---|---|---|
| Requirem. | 583 | 19 | **3%** |
| Design | 157 | 13 | **8%** |
| Prototyping | 115 | 2 | **2%** |
| Coding | 795 | 112 | **14%** |
| Testing + installation | 384 | 58 | **15%** |
| **Total** | **2034** | **204** | **10%** |

- **Case 2**

| | VTM | Actual | Deviation |
|---|---|---|---|
| Requirem. | 583 | 118 | **20%** |
| Design | 157 | 102 | **65%** |
| Prototyping | 115 | 56 | **49%** |
| Coding | 795 | 307 | **39%** |
| Testing + installation | 384 | 166 | **43%** |
| **Total** | **2034** | **749** | **37%** |

In case 1 only twenty percent of the classes had to be adapted. These adaptations mostly consisted of adding specific features. In the other case, the existing work flow procedures had to be revised, but the underlying domain objects could be reused.

- **Elapsed time of the different projects**

| | VTM | Case 1 | Case 2 |
|---|---|---|---|
| Duration project | 18 mnths | 7 mnths | 10 mnths |
| User testing | 6 mnths | 2 mnth | 3 mnths |

An application like PSI must be fully proven because the application is used during transmission and bugs can result in severe commercial loss. During the first project, the end-user wanted to be 200% sure before he used the application into production. That explains the long test period. In case 1 and 2 however we noticed that the test period is shorter due to, amongst others, the maturity of the framework.

- **Hardware Architecture**

| | VTM | Case 1 | Case 2 |
|---|---|---|---|
| Client | Mac | Win NT | Win |
| Server | RS6000 | Sun | Win NT |
| Database | Oracle | Ingres | Informix |
| Number of users | 30 | 50 | 100 |

## References

[Gamma95] Design Patterns, Elements of Reusable Object-Oriented Software, Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, Addison Wesley

[Goldberg95] Succeeding with Objects, Decision Frameworks for Project Management, Adele Goldberg & Kenneth S. Rubin, Addison Wesley.

[Rubin 92] Object Behaviour Analysis, Kenneth S. Rubin & Adele Goldberg , Communications of the ACM, Vol 35(9), ACM Press, September 1992.