Adoption of Modern Techniques for the Development Process of the Project Management System PROMAN W®

Daniel Petrik, Oliver Moravcik, Tomas Skripcak

***Abstract*—This article is aimed at the process of re-engineering the architecture of a long established information system and the problems associated with adopting new techniques used as a foundation for the new architecture. The article starts with a brief description of the management system Proman W, which has architecture that needs to be redesigned in order to fit the current needs of customers. Following this we introduce the requirements of the new architecture and describe the decisions behind their implementation. Next we provide insight on how the processes of adopting modern techniques were handled in the re-engineering of the information system Proman W. Finally the opportunities for further modifications within the system's architecture are initiated. The main goal of this article is to provide an overview of the new technologies' implementation process in the Proman W system's architecture and to point out aspects that we found important to note whilst re-engineering the long running applications.**

Keywords- MDA; MVVM; ORM; composite application; testing;

I. INTRODUCTION

The software development field in industry is one of the most rapidly changing branches in the world. Tools and approaches used for developing information systems are developing in order to provide better quality and maintainability of software solutions. Well designed architecture is the basic pre-requirement for a successful information system which can dynamically react to the changing requirements of the end users. But, as time passes, each application becomes obsolete and the need for new implementation is more urgent.

Manuscript received on 26th January 2011; revised on 16th March 2011.
Daniel Petrik. Autor is with the MMS SOFTEC Ltd., Hajdoczyho 1, 917 01 Trnava, Slovakia (e-mail: petrik@mms-softec.sk).
Oliver Moravcik. Author is with the Slovak University of Technology - Faculty of Materials Science and Technology in Trnava, Paulínska 16, 917 24 Trnava, Slovakia (corresponding author to provide phone: +421 33 5511033; fax: +421 33 5511758; e-mail: oliver.moravcik@stuba.sk).
Tomas Skripcak. Author is with the Slovak University of Technology – Faculty of Materials Science and Technology in Trnava, Paulínska 16, 917 24 Trnava, Slovakia (e-mail: tomas.skripcak@stuba.sk).

Development of a new version of a working application can also be seen as the process of new techniques and the adaptation of tools, in order to use and enhance knowledge from an old system to a new one.

Typical end users of software systems are usually conservative thinkers. The re-engineering of a working application has to be done carefully, one must bear this in mind otherwise the transfer of the users' skills to a new version of the information system can be quite expensive.

A. History of Proman W

Proman W is an enterprise information system focused on project and personal costs management, including budget planning and balancing. This system can be easily integrated with other enterprise applications e.g. accounting systems (MACH, WinLine), enterprise resource planning systems (SAP) or personal accounting systems (Best, PAISY,). According to [3], Proman W can be characterized as a sovereign application, which takes the users full attention when working with it.

From an architecture perspective, Proman W was designed as a client-server application with support for two relational database management servers (MS SQL, Oracle). Integrated development environment (IDE) Delhi (first rapid application development which supports Object Pascal as a programming language) was used for the implementation itself. The first version of Proman W was released in 1999 and until last year it was regularly updated [17].

The old architecture of Proman W started to be a limiting factor when it came to the implementation of the newly required features of modern users. This is why, in 2008, a decision was made to develop a new version of the system based on current techniques and methodologies in the software development industry.

B. Requirements

In order to design up to date and maintainable architecture, bare bone techniques and tools have to be chosen, these will be used in the whole life cycle of the software product. According to analysis of recent trends in software development and expectations from the resulting system, we specified a group of requirements; these are described in detail below. Figure 1 shows the graphical representation of such a system.

ISBN: 978-988-18210-9-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

1) Managed application framework and IDE

With consideration to customer needs, Microsoft Windows remains the target operating system. We were looking for a mature application framework, which is continuously developed, has good documentation and enough resources that are available for developers. This is important in order to make the process of adopting the new framework easier and more fluent for developers, which will also lead to a lower total cost of re-engineering. When it comes to Windows development .NET [12], this is usually the first candidate. We had a positive experience with the development on .NET, which was mainly aimed towards web applications. Focusing on this platform was the next logical step. As .NET is a managed environment, it also provides us with an opportunity to deliver an optimized installation of a 32 and also a 64 bit version of the application depending on a single code repository. Microsoft Visual Studio 2008 [13] (later updated to 2010) was chosen as the default IDE for .NET development.

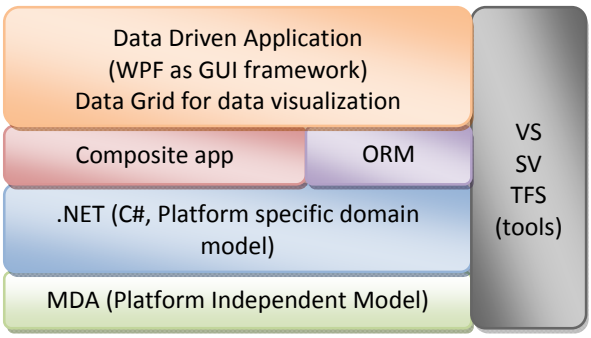

Figure 1. Representation of new architecture depending on requirements

2) Model Driven Architecture (MDA)

The idea of MDA (sometimes called Model Driven Development MDD) is to have a model of the application captured in the platform independent manners (also called platform independent model PIM). This means that, if the model is well designed, it can be used as a base for each platform specific model which can then be transformed by implementing the designed system. A standard way to express a PIM is by the use of a unified modelling language (UML). PIMs are usually used for documentation generation [18].

There are many Computer Aided Software Engineering (CASE) tools on the market, which support ideas of MDA. Some of them provide an option for the whole of the software's life cycle's process management. In our case we used a product called Enterprise Architect [19], it was developed by the Sparx Systems Company. An important feature for us was the synchronization of an independent UML model with a source code repository. In order to stay agile, at the beginning of the development, we needed an automated translation of UML to the initial implementation (code generation), afterwards when the model was refactored into the code it was necessary to publish these changes back to the UML model (reverse modeling).

3) Composite application design

Composition and modularity are principles which when applied to well designed and loosely coupled function blocks, allow for the reusability and convertibility of system parts. The definition of a composite application is as follows:

- *The term composite application expresses a perspective of software engineering that defines an application build by combining multiple modules into a new application* [22].

At the beginning, we were evaluating two possible frameworks for creating rich composite applications. Caliburn [5] developed by Rob Eisenber and Prism [14] (aka Composite WPF) created by the Microsoft Patterns & Practices team in order to provide guidelines for developing a line of business (LOB) application with a new graphic user interface (GUI) technology called Windows Presentation Foundation (WPF). From a functional point of view they both provided similar features, however as it was one of the basic parts of developing the system, we concluded that documentation and further support is not amongst the most important attributes. Caliburn during that time simply did not offer enough learning resources and the adoption process would have had a long learning curve. On the other hand, Prism had completed documentation with code examples and a reference application. This turned out to be a large advantage in the phases that followed.

4) Object Relational Mapping (ORM)

Proman W can be classified as a Data-Driven application. This is true for most of the LOB systems. From a developer's perspective, it is ideal to have a standard method for persistent data manipulation. The relational database server is used for backend data storage. Each customer has his/her own preference regarding the database server, mainly because they want to use the one which is already in use and with which they have experience with. In order to support multiple types of relational databases, we decided to use ORM technology as a primary data manipulation mechanism within the new version of the system. It enables us to hide all data source specific details. The object oriented domain model in the information system can be directly used, by the utilization of ORM, with the relational database server. According to [4], features which are critical when a decision about which ORM technology will be integrated into Proman W is made, were schema generation and update ability, ease of integration, setup and prototyping, reasonable transaction implementation (ideally as one of a standard design pattern e.g. Unit Of Work [8]) and good query mechanism (Language Integrated Query aka LINQ [11] is an advantage). Some simple tests were also made in order to determine the performance of ORM frameworks when dealing with large amounts of data. According to information provided in [20] we had selected a suitable ORM prototype testing application which was built

ISBN: 978-988-18210-9-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

in order to find out how each of these frameworks will work with data from the Proman W system. A summary of impressions for each of the tested ORMs is as follows:

- Data Objects .NET [25]: ORM developed by the X-tensive company was tested in version 3.9 (current version 4.4 is its successor, but it was completely rewritten). Data manipulation was not intuitive enough. Performance was good and it also supports schema generation and updates but its other features do not correspond to initial costs. The implementation of the Unit of Work design pattern is also missing.
- Entity Spaces .NET [6]: This ORM does not support schema generation and updates. It works with a Database Model first approach, where the object model was generated according to an existing database model. Manipulation with data was intuitive. A problem occurs when large amounts of data have to be presented to the user. This framework creates a complete object tree without checking if all of the data will be really presented, thus having a negative impact on performance.
- XPO for .NET [4]: ORM framework developed by the Dev Express Company. It supports all required features. Our prototype application for testing purposes has not discovered any serious problems. Later, during the process of implementing the new version of Proman W, there were a few performance issues which were solved.

5) Data visualization

For Data Driven Applications it is essential to present the user with all kinds of complex structured data. The user control used for this purpose is usually called Data Grid, which displays data in a tabular format. In the beginning of the new Proman W development there was no such user control included in the WPF GUI framework, however Companies like Infragistics [9] and Xceed [24] provide their own implementation of Data Grid control with advanced data manipulation functionalities, namely: hierarchical data (in the form of a tree), searching, multi column ordering, grouping, master-detail view, and automatic error propagation from data objects). We have done some experiment applications for the purpose of finding out how the 3rd party UI components behave together with huge amounts of data selected via the ORM framework. We found out that a bottle neck in this process was created due to the graphical representation (rendering) for each column and row of dataset and not due to the operations on top of the data as it was expected. The root of the problem is that graphical elements are created for all rows of records and if there is a million records it takes a significant amount of time. To solve this issue in WPF, an approach called UI virtualization is used. By definition, UI Virtualization is a concept where UI elements are only created and maintained when they are visible on the screen [23]. The resulting performance then mainly depends on how each vendor implements the UI virtualization internally. The Xceed Data Grid provides a superior virtualization and was used as a default Data Grid control in the new version of the Proman W system.

6) Version Control

The last requirement was aimed at the tool which supports the developer in his daily tasks. A centralized version controlling system SVN (Subversion) [21] was used at the beginning of the development process. After an upgrade to the new version of Visual Studio, we found that this changed our versioning system to the TFS (Team Foundation Server) [15]. In addition to a centralized version controlling, TFS also works like a continuous integration system with an automated build process.

II. The Adoption Process

In this section we are outlining methods used in the re-engineering of Proman W and describe how modern techniques were adopted during the product development life cycle.

A. UML model first approach

In order to make the development process more flexible, we were using a combination of MDA and agile techniques. We were using a UML class model of our information system as a basis for the generation and implementation of the domain model code via the CASE tool. As was stated before, two methods of synchronization exist between the implementation code and the UML model, this means that changes are propagated between code and model. ORM technology allowed us to automatically generate and also update the database schema. All the processes are shown in Figure 2. This approach proved to be very useful from the beginning. It sped up the initial development and maintenance of the working modules.

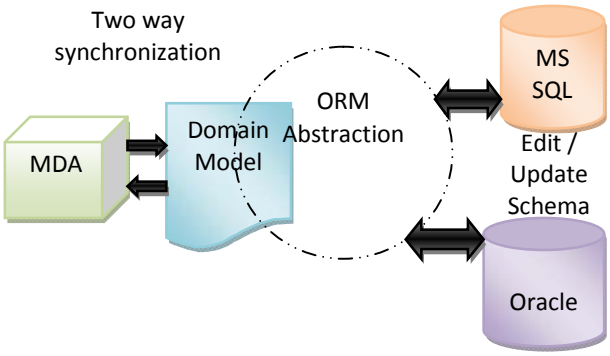

Figure 2. A combination of the agile MDA in the process of Proman W development.

B. New GUI framework and modular design

We decided to use WPF as the default GUI framework in Proman. WPF has a completely new way of dealing with GUI composition. When the re-engineering of Proman started, WPF was a relatively new technology and as we know only a few companies had started to use it for the development of LOB applications. The following is a list of

ISBN: 978-988-18210-9-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

the most important aspects learned during development, which should be considered.

1) UI patterns

With WPF, it is more natural to design an application according to the Model-View-Controller (MVC), Model-View-Presenter (MVP) or Model-View-View Model (MVVM, sometimes also called Model-View-Presentation Model) [7]. Prism, which was used as guidance for creating composite modular applications, has support for these architectonic design patterns. First Proman modules were developed according to MVP, later we switched to MVVM, which fits better to the WPF architecture.

2) Simple CRUD modules (create, read, update, delete) prototiping)

When designing an interactive paradigm in Proman W, we discovered that some of the interaction paradigms used in the old system did not have implementation in WPF. It was necessary to look on new interaction conventions for end users. For this purpose we started development with simple modules with basic CRUD functionality. Here we experimented with possible interaction options and at the end established a convention for designing a common UI for every Proman module.

3) Asynchronuos operations execution

Introduction of a data abstract layer in the form of ORM together with a more complex, loosely coupled architecture of a composite application has a negative effect on the performance of some operations. This is why we have to include a paradigm of an asynchronous operation in the new design of the Proman W system. In WPF it is possible to separate the GUI thread from a working thread so long as the running operations do not block GUI and working with the application is more fluent.

C. Convention over Configuration

Convention over Configuration is a technique focused on simplifying a development task. It is quite common that developers are trying to deliver their solutions of problems as general as possible. This means that in order to use such a general solution, it is necessary to configure it for specific needs. However, the Convention over Configuration technique is defined by specific rules used as a standard for development tasks (e.g. naming convention of modules, handling user interaction, validation, etc.). When these rules are respected, it is much easier to automate configuration according to them. At the end it leads to better code implementation, where the developers do not need to take care about the common infrastructure configuration while writing code based on convention. From our experience, the use of convention can rapidly shorten the development time needed for the introduction of a new prototype module in our composite application oriented system.

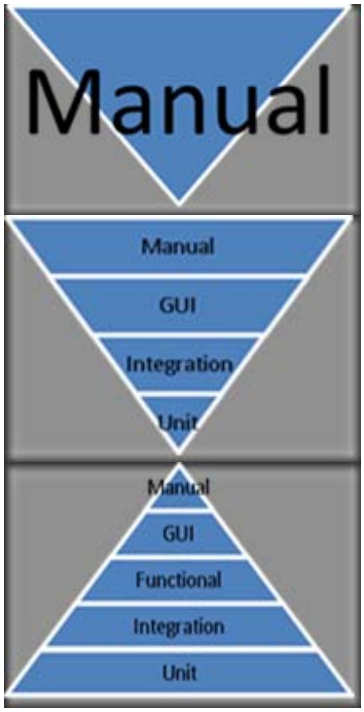

Figure 3. Continuous way for automated testing [1]

D. Test Driven Development (TDD)

In the context of TDD, having reasonable testing sets are essential. The problem is that developers need to learn how to write good tests. This is very important otherwise it could result in a situation where the maintenance of tests takes much more time than the maintenance of the actual product. According to [1] it is possible to have a whole complete test set by the continuous reduction of manual testing and the introduction of other types of automated tests. This process is shown in Figure 3. We are comfortable with this method and in our opinion it is a good way for starting to produce TDD in the correct matter.

III. Feature Ready Architecture

By implementing our set of architecture requirements we are able to make a significant adaptation on the demand. Following is a list of options, which will most likely be required by the end users in the future:

- Change of presentation layer: web based applications are more popular in the LOB field of information systems. MVVM architectonic design pattern used in Proman W, enabled to change View component, so the WPF presentation layer can be replaced with e.g. web based Silverlight technology or HTML 5.
- Flexible data manipulation: usage of ORM does not only hide low level SQL details of the relational storage system, its abstraction allows us to work with data in different formats, e.g. in the context of a web rich client application, where the system is actually running on the client side, we need to have

ISBN: 978-988-18210-9-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

access to data via a standard web protocol. ORM supports OData protocol [2][16]. This flexibility is especially useful when dealing with data-driven applications like Proman W.

- Complete automated testing: one of our future plans is the full automation of tests (unit, functional, integration, and scenario) in our system. Loosely coupled architecture is the basis and necessary condition in order to make this task possible.

IV. Conclusion

The aim of this article is to describe the process of re-engineering the architecture of the obsolete data driven application Proman W. The main requirements for the final information system's architecture were outlined and reasons behind the architecture's decisions were analyzed. The adoption of new development techniques in the process of re-engineering a long term running application is difficult. We provide a description of how this process was handled whilst developing the new version of Proman W and we pointed out a few important aspects which should be taken into account while re-engineering that contains changes in the basis of the architecture's components, such as the application and GUI framework. The resulting implementation of the information system allows for easier maintenance and loosely coupled relations between application components, this makes Proman W ready for further changes.

References

[1] M. Bussa, „Evolution of Automated Testing," [Online 2011], [cit. 2011-03-22]. Available on the Internet <:http://www.matthewbussa.com/2011/01/evolution-of-automated-testing.html.>.

[2] Codeplex, "eXpresss Persistent Object Toolkit," [Online 2011], [cit. 2011-03-23]. Available on the Internet <:http://xpo.codeplex.com/.>.

[3] A. Cooper, R. M. Reimann "About Face 2.0," Wiley, 2003, Indiana, p. 504, ISBN: 978-0-764-52641-1.

[4] DevExpress, „eXpress Persisten Objects," [Online 2011], [cit. 2011-03-09]. Available on the Internet <:http://www.devexpress.com/Products/NET/ORM/.>.

[5] R. Eisenberg, „Calibrun," [Online 2011], [cit. 2011-03-14]. Available on the Internet <:http://www.caliburnproject.org/.>.

[6] Entity Spaces, „Persistance layer and business objects for .NET," [Online 2011], [cit. 2011-03-09]. Available on the Internet <:http://www.entityspaces.net/Portal/Default.aspx.>.

[7] M. Fowler, „GUI Architectures," [Online 2006], [cit. 2011-03-15]. Available on the Internet <:http://martinfowler.com/eaaDev/uiArchs.html.>.

[8] M. Fowler, „Unit of Work," [Online 2002], [cit. 2011-01-09]. Available on the Internet <:http://martinfowler.com/eaaCatalog/unitOfWork.html.>.

[9] Infragistics, „WPF Controls," [Online 2011], [cit. 2011-03-22]. Available on the Internet <:http://www.infragistics.com/dotnet/netadvantage/wpf.aspx#Overview.>.

[10] P. H. Kuaté, T. Harris, C. Bauer, G King, "NHibernate in Action," Manning, 2009, p. 400, ISBN: 978-1-932394-92-4.

[11] F. Marguerie, S. Eichert, J. Wooley, "LINQ In Action," Manning, 2008, p. 576, ISBN: 1-933988-16-9.

[12] Microsoft, „Microsoft .NET," [Online 2011], [cit. 2011-02-20]. Available on the Internet <:http://www.microsoft.com/net/.>.

[13] Microsoft, „Visual Studio," [Online 2010], [cit. 2011-02-20]. Available on the Internet <:http://www.microsoft.com/visualstudio/en-us/.>.

[14] Microsoft patterns & practices, „Prism," [Online 2010], [cit. 2011-03-04]. Available on the Internet <:http://compositewpf.codeplex.com/.>.

[15] Microsoft, „Team Foundation Server," [Online 2010], [cit. 2011-03-22]. Available on the Internet <:http://msdn.microsoft.com/en-us/vstudio/ff637362.>.

[16] Open Data Protcol [Online 2011], [cit. 2011-03-23]. Available on the Internet <:http://www.odata.org/.>.

[17] D. Petrik, O. Moravcík, "Innovation of the Software Product PROMAN W," International Workshop Innovation Information Technologies – Theory and Practice, 2010, Dresden, pp. 55-58, ISBN: 978-3-941405-10-3.

[18] J. D. Poole., "Model-Driven Architecture: Vision, Standards And Emerging Technologies," ECOOP Worshop on Metamodeling and Adaptive Ojbect Models , 2001, p. 15.

[19] Sparx systems, „Enterprise Architect," [Online 2011], [cit. 2011-02-21]. Available on the Internet <:http://www.sparxsystems.eu/.>.

[20] I. Stanek, "Objektově-relační mapování pro platformu .NET," České vysoké učení technické, diploma thesis, Praha, 2008, p. 107.

[21] Tigris, „Subversion," [Online 2005], [cit. 2011-03-01]. Available on the Internet <:http://subversion.tigris.org/.>.

[22] Wikipedia.org, „Composite application," [Online 2011], [cit. 2011-03-11]. Available on the Internet <:http://en.wikipedia.org/wiki/Composite_application.>.

[23] WPF Wiki, „Virtualization," [Online 2010], [cit. 2011-03-30]. Available on the Internet <:http://www.wpfwiki.com/(X(1)S(wxor0z451rijnueg2yavyu55))/Default.aspx?Page=WPF%20Q4.3&AspxAutoDetectCookieSupport=1.>.

[24] Xceed, „XCEED DataGrid for WPF," [Online 2011], [cit. 2011-03-22]. Available on the Internet <:http://www.xceed.com/Grid_WPF_Intro.html.>.

[25] X-tensive, „DataObjects .NET 3.9," [Online 2011], [cit. 2011-03-09]. Available on the Internet <:http://x-tensive.com/Products/DO39/Default.aspx.>.

ISBN: 978-988-18210-9-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)