University of Bordeaux 1

Internship Report

Master of Software Engineering (2012-2014)

Migrate Oracle Forms To .NET Windows Forms

Author:
DO Ngoc Cuong

Supervisor:
VO Tran Trong Vu

January 11, 2015

Abstract

Oracle Forms is no longer supported since 2008. The customer wants to do a migration of all these applications to a more modern .NET framework. The goals of the project are to have a single homogenous and flexible environment, obtain clean and transparent processes, and reduce complexity. During my internship with ELCA, as a member of the development team, my tasks are verifying the specifications in order to understand all the business and technical problem thoroughly; I also participated in developing the assigned modules, debugging the code; writing the unit test for the modules as well as providing necessary support to other team members.

Contents

List of Figures

Acknowledgements

This report has been accomplished with my determination and my long-last desire of conducting a study in an authentic setting as a real professional. However, it cannot be made possible by the contribution of many great people who have involved in my study process. I would like to express my gratitude to the following for their invaluable help, care and support. I would like to offer my special thanks to my supervisor and the colleagues, who effectively helped me to finish the internship via helpful suggestions and clarifications. Last but not least, I would like to offer my special thanks to ELCA, the warm environment in where I was lucky enough to spend my internship period.

Ho Chi Minh City, January 2015
DO Ngoc Cuong

Chapter 1

Introduction

1.1 An Overview about PUF

Website: http://www.pufhcm.edu.vn/
The Pôles Universitaires Français (PUF) was created by the intergovernmental agreement of VietNam and French in October 2004. There are two PUF centers in VietNam: PUF HN located in Ha Noi and PUF HCM located in Ho Chi Minh City. PUF has more than 500 students participated in courses in the field of Commerce, Economics, Management and Informatics. Courses in PUF are provided in French, and English and Vietnamese by both the French and Vietnamese professors. Currently, PUF-HCM cooperates with the University of Toulouse I Capitole, University of Paris VI and the University of Bordeaux 1 providing Bachelor's, Masters and PHD courses.

1.2 ELCA introduction

1.2.1 An overview

Website: https://www.elca.vn/
ELCA is Switzerland's largest independent software development company. In 1998 ELCA was one of the first 100% foreign-owned software companies to open an office in Ho Chi Minh City. The company develops on .NET and JAVA platforms and integrates products like SharePoint and CRM. The quality system is appraised at CMMI maturity level 3. ELCA Vietnam started with six people. Today we are a production facility with more than a hundred employees. During the same period ELCA Switzerland tripled its workforce to over 450 engineers.

Our customer has used Oracle Forms as their main application platform

for the last 20 years. During this time, several hundred applications have been developed. Since 2008, Oracle Forms is no longer supported. The customer now wants to migrate all their applications to a more modern .NET framework. The goals of the project are:

- To have a single homogenous and flexible environment;
- To obtain clean and transparent processes;
- To reduce complexity.

The ELCA Company serves as an extension of the development team of the customer and will be tasked with migrating individual applications or packets of applications.

1.2.2 My supervisor

I would like to express the deep appreciation to Mr. VO Tran Trong Vu for giving me the opportunity to work with him. His calm and patience support, guidance, advice in terms of the design, methodology, and how to perform the task in an efficient way helped me in my short period of time in ELCA. His prompt responses and thought-provoking arguments have helped me much in completing this result. In addition, I always feel assured and more confident to explore and figure out the best solutions for myself with his support and suggestions. This is really important for me who am so novice in large-scaled projects.

1.3 An overview of my task

The duration of the internship is six months, starts from July 1, 2014 to December 30, 2014. As a member of the development team, I developed several modules for 6 months. During this period of time, I was a developer. My tasks are verifying the specifications in order to understand all the business and technical problems thoroughly. I also participated in developing the assigned modules, debugging the code; writing the unit test for the modules as well as providing necessary support to other team members.

1.4 The internship report

This report involves six chapters. We firstly give a brief overview of our report with its top most basic goal in the first introductory chapter. In the second

chapter, we introduce the theme of our project and our company policy in finishing a task. The aim of the project and our customers' requirements will be presented in a condense way in this chapter. In this chapter, we also dive into the challenges we encountered when finishing the project.

In the fourth chapter, we present the detailed tasks as well as the solutions applied to tackle the problems highlighted in the previous chapter. To give a deeper understanding about our projects in ELCA, we spend the first section of the chapter to describe the system in more detail. The layer architecture is playing the vital role in our project and a careful understanding in this architecture is necessary for inferring any further related problems and solutions applied in the system. Due to the company's policy, it is forbidden to provide the projects names and its code in a clear way. I write the report in a descriptive way rather than provide a clear example of the whole project. Finally, in the last chapter, we summarize our report not forgetting to refer to some of the future work for the projects in related fields.

Chapter 2

Theme of the project

2.1 Requirements

Since 2008, Oracle no longer supported Oracle form. Our customer found Oracle is slow and quite obsolete at this period of time. The application is previously deployed using the Oracle database. The very first requirement is that the new version of the application must reuse the Oracle database with little changes made in the existing database. The database is a 20-year-old application and there are some inconsistency found when new requirements released. Besides, the newly developed application and the existing one must be operated simultaneously.

The high volume of business rules might be encountered as another drawback in our task. Most of the business rules in the specs cannot be fully understood. Besides, the previous system is provided with some patterns due to the view of the designers at that time. The growth speed of the business requirements soon far exceeds the designers' vision. The code base cannot adapt all the variety of requirements. The previous patterns designed in order to support extensibility and scale up seems to be obsolete in this situation. Our customer showed their interests in the CSLA, Windows form and NHibernate technologies. We marked down this and focus on these technologies to build up our new application. The new application must strictly follow the specs delivered by our company. We as a team of five developers will organize and create the module in the period of three weeks which will be called a spint" (please refer to figure 2.2 in the next page for an illustration). Every three weeks, we receive a set of task sheets. They are modules that we are going to implement. We first read the specs relevant to the delivered task sheets carefully. The implementation starts from the lowest level of the system architecture, which is the database. We might write the database

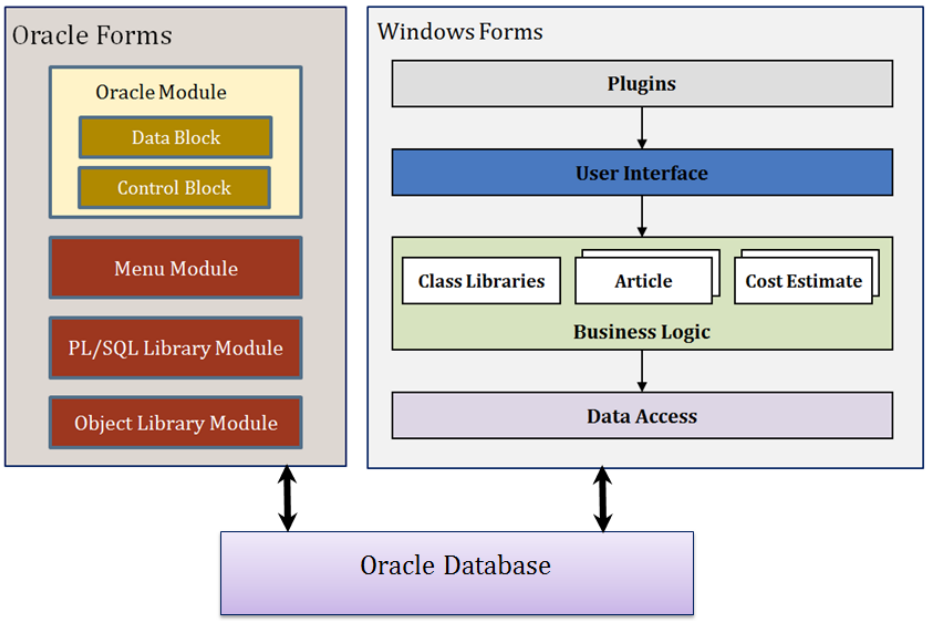

Figure 2.1: Old and new system architecture.

script if there is any necessary change in the database to conform the module described in the task sheet. We then implement the data access layer. After that, we implement the business layer and then the GUI. Our last step is to test and fix the bugs if there is any. The last step is to optimize the functions and code. This is our typical routine in finishing a task in a spint. We have to strictly follow the company architecture. The architecture will be described in chapter 3 because understanding and being able to follow the architecture will lead to the succeed in implementing a new module.

Quality Assurance keeps a high priority in our tasks. All of our code must pass a strictly double checked in both VietNam and Switzerland. The code must first pass the unit test provided by the specs then the test made by our dedicated testers, this phase will be done in three-week time. The successful code will be delivered to the Swiss colleagues for further checked at the fourth week of a sprint.

2.2 Challenges

I found myself a big gap because of the missing experience in CSLA.NET (Component-based Scalable Logical Architecture) and the NHibernate. In the other hand, the application's library is large and is not fully documented.

The old version of the application, as mentioned previously, is using the

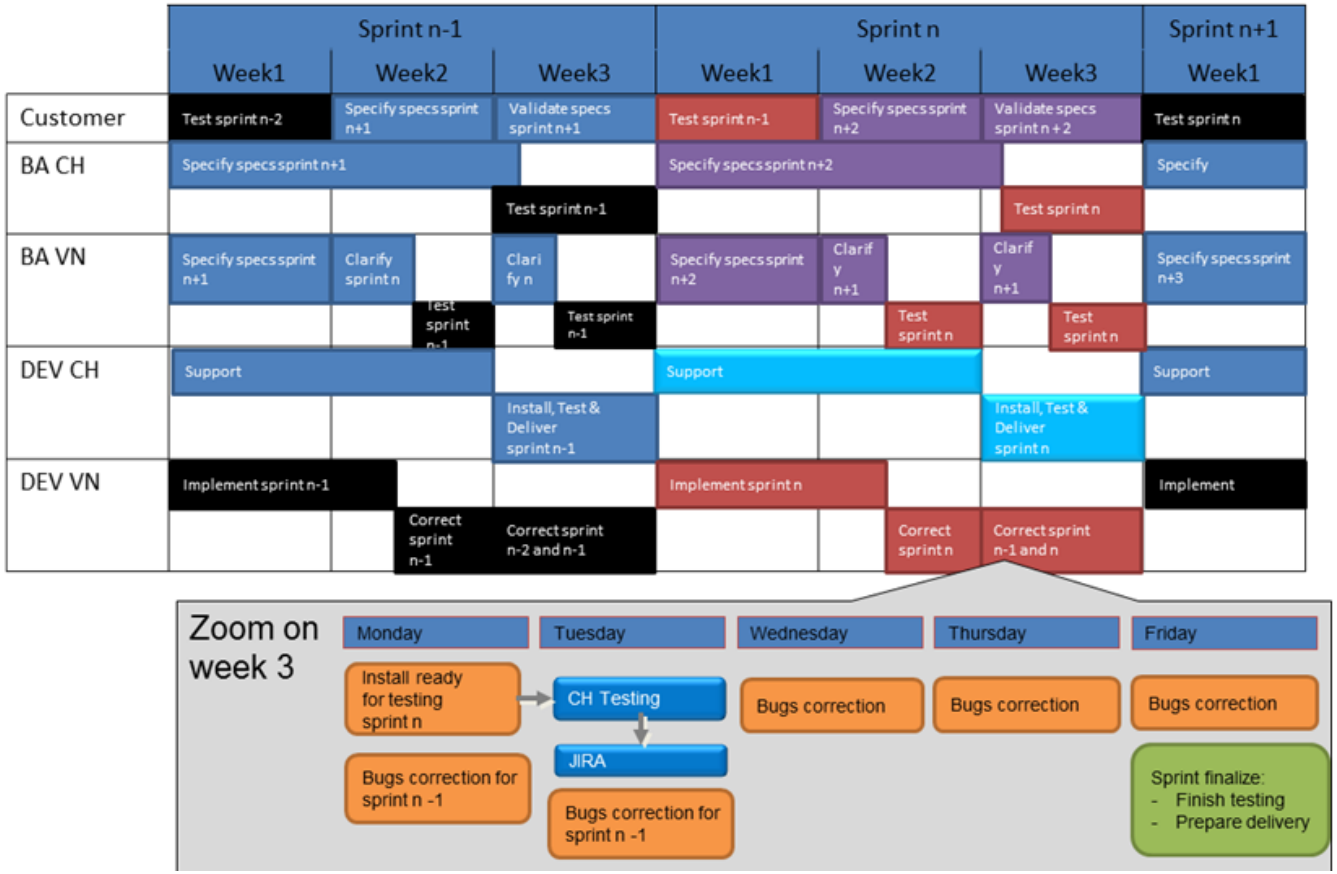

Figure 2.2: An illustration of Sprints

BA: business analyst's task
DEV: developers' task
CH: task will be done in Switzerland
VN: task will be finished in VietNam

Oracle database, we found keys problems with this type of database in migrating. Despite of the Oracle fully supported keys and constraints, there is no foreign key and uniqueness constraints found in this version of the application. We might blame the poor designs of the previous version of the database but this became one of our most concerns in the new design. We must carefully handle this problem due to the fact that foreign key cannot be added in the new version of our application but the new version must persist the data consistency. Besides, the uniqueness constraints are checked using the hard code, which is not recommended in the modern application. We also found many composite keys in this database design which is hard to optimize for NHibernate lazy load as they are designed for.

The old version used package store procedure and native query. The new version is recommended to use the Query Over to generate the query since it is easy to maintain. The problem is that the generated query might not always gives the same results as the previous native queries. The performance is also affected when using the old fashioned native query. In the other hand, the sophisticated queries can only be used in native query.

Never before did I work with the real system then the high volume of data is another problem occurred. The system also consists of large amount of modules and it takes quite long time to build and debug.

As previous mentioned, all the specs and the schedule must be strictly followed. The specs documents are all business requirements. We as the developers will work with the technical requirements and there are some mismatches in the architecture and design document [3].

The main idea of the CSLA is the Parent-child relationship. To take the full advantage of the CSLA, the data model must be mapped into the parent-child relationship. More about this relationship will be described in the book [4, 5].

Chapter 3

Project in details

3.1 Architecture overview

The system consists of multiple different applications running in a single application container (the plugin application) adapted the needs of the different user groups. Every application implements a specific group of aspect of the system. To enforce a consistent user experience and to provide only a single Windows application to the users, the system uses the plugin pattern.

The different applications plugged into the plugin application are all structured according to strict layer architecture, i.e. the layer architecture is the top level structural decomposition used.

The system consists of multiple different applications running in a single application container (the plugin application) adapted the needs of the different user groups. Every application implements a specific group of aspect of the system. To enforce a consistent user experience and to provide only a single Windows application to the users, the system uses the plugin pattern.

The different applications plugged into the plugin application are all structured according to strict layer architecture, i.e. the layer architecture is the top level structural decomposition used.

3.1.1 Plugin application concept

All applications are built as plugins running inside the plugin application. At startup, this plugin application will load all the deployed plugin assemblies. These assemblies contain entry point classes for the module: the plugin classes. Then to find those plugin classes, the plugin application searches in the plugin assemblies for the implementer. It will interact with those plugins by calling methods of this interface. The plugin application can also communicate with all the application windows to send them toolbar commands

like close, refresh, etc.

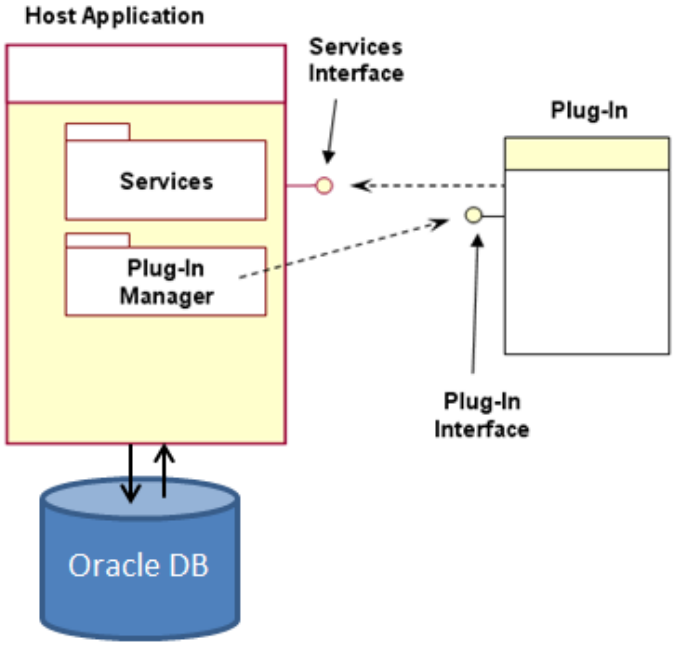

Figure 3.1: Plugin architecture.

3.2 Structural decomposition

N-layer architecture is all about separating different types of functionality. The common logical separation is into a Presentation layer, a Business layer, and a Data layer. These may exist on a single machine or on many separate machines the logical architecture does not define those details. A physical n-tier is quite different from logical n-layer architecture. In n-tier architecture, the application is spread across multiple machines with different functions: a client, a web server, an application server, a database server, and so on. There is a relationship between an application's logical and physical architectures: the logical architecture always has at least as many layers as the physical architecture has tires. There may be more logical layers than physical tiers (because on physical tier can contain several logical layers), but never fewer.

The current application is structured into the layers as the figure 3.2. The layering is strict, i.e. it is not allowed to skip a layer. The only exceptions are the common technical components, which can be used by Presentation, Business and Data Access layer. Below the layer name, the diagram also

shows the main frameworks/ tools used in the corresponding layer, e.g. the Data Access layer is built based on NHibernate.

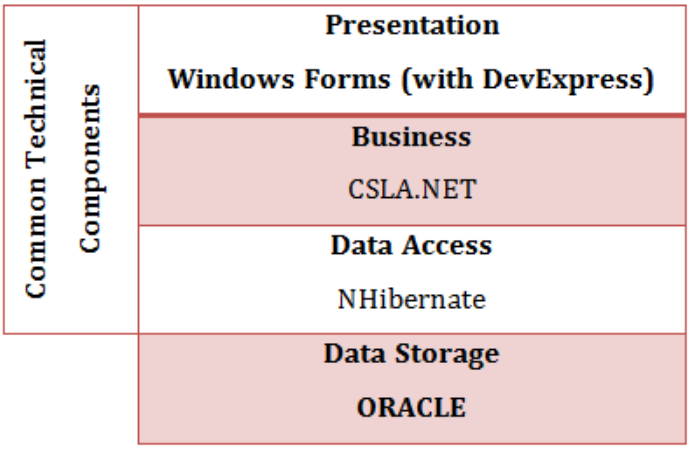

Figure 3.2: Application architecture modeling overview.

The logical layers are deployed as a rich client, i.e. Presentation, Business and Data Access layer are running on one machine as a single application. The database is running on a separate server, i.e. it is a 2-tier physical deployment.

Having clean separation between these layers makes the application more maintainable, because changing one layer often has minimal impact on the other layers. Properly designed logical n-layer architecture provides the following benefits:

- Easier maintenance
- Better reuse of code
- Better team-development experience
- Higher clarity in coding

On the other hand, properly chosen physical n-tier architecture can provide the following benefits:

- Performance
- Scalability
- Fault tolerance
- Security

3.2.1 Presentation layer

The presentation uses Windows Forms with DevExpress controls to significantly improve productivity and provide a better user experience for consumers. DevExpress has made the application development easier by providing easy to use controls for Windows Forms. DevExpress [1] controls use standard Visual Studio development methods. They are one of the most stable and easy to use tools that can cut down your development time up to 50%. The CSLA framework, is used in business layer, also provides some advanced features for data binding and validation. The application uses these features to bind Windows Forms UI to the Business layer.

3.2.2 Business layer

The Business layer is built using a different domain model for different use cases. Business logic includes all business rules, data validation, manipulation, processing, and authorization for the application. The business logic must reside in a separate layer from the presentation code. It must implement all the business logic, because it is the only point of central control and maintainability.

CSLA .NET is the heart of Business layer that provides a standard way to create robust object oriented programs using business objects. Business objects are objects that abstract business entities in an object oriented program. A business object encapsulates all the data and behavior (business logic and rules) associated with the object it represents. For example, an OrderEdit object will contain the data and business rule implementations necessary for the application to correctly allow the user to edit order information. Business objects created using CSLA .NET fully supports data binding for all Microsoft .NET UI technologies, including Windows Forms.

3.2.3 Data Access layer

Data access code interacts with the Data Storage layer to retrieve, insert, update, and delete information. The Data access layer does not actually manage or store data; it merely provides an interface between the business logic and database. By isolating the data access code into a specific layer, the impact of later changes, i.e. data access technology, is limited to a smaller part of the application.

NHibernate [2], an object-relational mapping (ORM) tool, is used in the data access layer to map between the object oriented business logic and the relational data in a data store. It also provides data query and retrieval

facilities. It generates the SQL commands and relieves the developer from manual data set handling and object conversion, keeping the application portable to most SQL databases, with database portability delivered at the very little performance overhead.

Concurrency management

The system uses optimistic locking to detect concurrent modifications on the data. The strategy used is therefore first writer wins. The assumption behind this is that conflicts only occur rarely and user accepts to lose their work under these circumstances. Optimistic locking is implemented by using the version column ROW_VERSION in tables.

Storing data

The mapping between CSLA business object and NHibernate data object is required to store data. To update new data, rehydrating the NHibernate data access object is performed from the database first, and then maps the CSLA business object to the data access object. The data objects only persist themselves to a database if their data has been changed. Data creation, retrieval, updates, and deletes are performed by clearly defined methods of the business object associated with the data. Data access logic is clearly separated from business logic, typically using a repository pattern or other mainstream object-oriented programming techniques.

The data access layer provides a repository per root object of the data object. The Repository is implemented based on NHibernate, which maps between the database and the data object.

A Unit of Work bundles together all the operations on the database belonging to one business action. The unit of work creates and manages the NHibernate session. Every interaction with the database must be part of a Unit of Work.

Data Storage layer

All the data are stored in already existing Oracle schemas. There are many challenges when using existing schemas such as all tables do not have foreign key constraints defined, many tables do not have unique keys or composite keys is already in place. Database changes can be requested such as adding artificial key columns, adding row version columns before attempting to implement an ORM tool. Chances are that the lack of foreign key constraints is masking underlying issues with the integrity of the data. While NHibernate will work without every foreign key being defined with a constraint, it makes

the database vulnerable to integrity violations, and it slows the database down because the Query Optimizer uses this information in its task of determining the best query strategy.

3.2.4 Solutions in short

During the six-month working at ELCA, I solved most of the difficulties mentioned in section 2.2 using my knowledge from courses in PUF as well as the colleagues' effective support. In this section, some of our solutions proposed and applied in the project will be discussed.

To solve the data inconsistency problem, no specific solution was proposed. There is no conflict with the previous version accepted but as mentioned above, the missing foreign key might lead to inconsistency. We currently solve this inconsistency by carefully check the data before implementing ORM tool. We also add the artificial keys to some of the tables in order to solve the composite key problem. The last problem with the oracle database application is the uniqueness constraints. This problem must be solved by checking the data in the customers' database, in case there is any conflict, we need to wait for the official discussion between our team specs and the customers for the more specific requirements. To improve performance, we apply the static cache and indexing for data. The performance is improved dramatically. This makes the system extensibility and scale up in the future. Besides that, I also use NHibernate Profiler regularly to detect in advance possible performance problem such as select N+1 problem. The select N+1 problem is described carefully in the website Hibernating Rhinos [1]. NHibernate Profiler gives us relevant data about the number of queries, session usage, query durations and warnings of possible performance bottlenecks.

The retrieved data model must be mapped to followed parent-child relationship of CSLA model. Some of the design patterns are applied to solve the problem. At the beginning, all of the documents must be strictly followed. Later, the code base is no longer matched to the requirements and new patterns as well as the code must be developed flexibly to follow all the specs.

[1] http://www.hibernatingrhinos.com/products/nhprof/learn/alert/selectnplusone

Chapter 4

Conclusion

Many lessons are learnt and many skills are gained after the internship with ELCA. We found ourselves confident with the Oracle database immigration tasks. The performance and security of the system are not violated despite of the high volume of data accessed. It is our chance to work with queries, system optimizations in a large scale system. This achievement is done via the right choice of technology as well as the well-designed and extensible architecture with an experienced team. We also learnt the rich library with its advantages not only in current projects but future applications.

Moreover, I personally and as a team member successfully developed six modules within the deadline. The rich content of the distributed systems, OOP, architecture design, and advanced database helped me a lot in my internship period. I also gained new knowledge and skill in CSLA, NHibernate and Oracle. The experience in system analysis, database design, architecture design, and data manipulation must also be counted into my achievements.

Thanks to ELCA to provide me the professional international environment. In the future, I might apply quickly to other high level projects. Not only technical skills but also soft skills such as time management, team work, and project management are all embedded into the internship period.

Bibliography

[1] Devexpress document 14.2. https://documentation.devexpress.com/, December 2014.

[2] Nhibernate reference document. http://www.nhibernate.info/doc/index.html, December 2014.

[3] ELCA. *Architecture and Design Document*, September 2013.

[4] Rockford Lhotka. *Expert C# 2008 Business Objects*. Apress, December 2008.

[5] Rockford Lhotka. *Using CSLA 4*. Marimer LLC, 2011.