UNIVERSITY OF BORDEAUX 1

INTERNSHIP REPORT

MASTER OF SOFTWARE ENGINEERING (2012-2014)

**MIGRATE ORACLE FORM TO .NET WINDOWS FORM**

|  |  |
| --- | --- |
| Author: | Supervisor: |
| DO Ngoc Cuong | VO Tran Trong Vu |

December 30, 2014

Contents

* Abstract
* Acknowledgement
* Introduction
* Several chapters
  1. Context
  2. Technical design
  3. …
  4. Results
* Conclusion [and future work]
* Bibliography
* Appendix

# Introduction

The customer, a large retail company 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 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
* Clean and transparent processes
* Reduced complexity

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

As a member of development team, I developed several modules for 5 months.

Trong thời gian khoảng thời gian 5 tháng tham gia vào project, đóng vai trò là một thành viên trong nhóm phát triển, công việc của tôi bao gồm verify specs nhằm nắm rõ các vấn đề liên quan tới business và technical, phát triển các module được giao, sửa các lỗi phát sinh trong quá trình phát triển các module, viết unit test cho module và support các thành viên nếu có thể

**ELCA Vietnam introduction**

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.

The report has the following chapters:

* Context
* Architecture overview
* Structural decomposition

# Context

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.

# 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.

## 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. The find those plugin classes, the plugin application searches in the plugin assemblies for 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. Class diagram of plugin pattern

**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 three 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 following layers:

|  |  |
| --- | --- |
| **Common Technical**  Components | **Presentation**  Windows Forms (with DevExpress) |
| **Business**  CSLA.NET |
| **Data Access**  NHibernate |
|  | **Data Storage**  ORACLE |

Figure. Application architecture modeling overview

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 interface, 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.

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

# Presentation layer

The presentation uses Windows Forms with DevExpress controls to significantly improve productivity and provide a better user experience to consumers. DevExpress has made the application development easier by providing easy to use controls for Windows Forms. DevExpress 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 use these features to bind Windows Forms UI to the Business layer

# Business layer

The Business layer is built using 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.

**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 application.

NHibernate, an object-relational mapping (ORM) tool, is used in 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 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 access object is required to store data. For updating 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 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 centric object model. The Repository is implemented based on NHibernate, which maps between the database and the data access 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 is stored in already existing Oracle schemas with very big data. 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, or performance issues. 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.

**Work to do**

A process for developing modules is sprint. A sprint is 4 weeks long. Team in Vietnamese has 3 weeks to develop, test, and then delivery to Switzerland. In CH, the modules will be tested more before delivering to the customers. Each member in team is tasked for 3 weeks: read specs, coding, testing, fixing bug or improving existing functions in application.

Read specs:

The specs được viết bởi team specs. Các specs được hoàn thành từ các khảo sát với khách hàng và thông qua việc sử dụng app cũ để nắm được các thông tin về business và technical của module cần migrate. Sau đó developer sẽ phát triển module dựa trên specs này. Tuy nhiên, do một số vấn đề về technical, các specs này sẽ được các developer verify thêm một lần nhằm phát hiện sớm các vấn đề liên quan đến technical hoặc các điểm chưa hợp lý cần cải thiện.

Các module được phát triển phải tuân theo Architecture & Design Document. Các phần cần phát triển cần phải tuân theo kiến trúc đa tầng đã nói ở trên. Do đó, một số điểm khi verify specs có thể tính đến như sau:

Về phía data model:

The application migrates from Oracle Forms to Windows Forms based on existing Oracle database schemas. There are several issues with these schemas as the following:

* Some tables lack unique keys
* Database does not have foreign key constraints defined.
* Composite keys are already in place
* Database use store procedures or package to read, create, update, delete information.

Các vấn đề về NHibernate:

* Hệ thống yêu cầu hạn chế native sql khi phát triển module. It is a changeling when many migration of application requires convert existing query using Criteria API or LINQ for maintainable but keeping performance.
* Queries will not be as well-optimized for lazy loading when using composite keys. Because the composite keys cannot be replaced by generated keys, there will be more setup involved to configure the mapping classes, and the resulting objects are harder to understand
* Some database changes such as adding artificial key or row version is considered. Nếu database không có key giả, điều tiếp theo cần xem xét đến là trong các bảng cần dùng cho module hiện đã có unique key nào hay không. Trong một số trường hợp việc này không thể thực hiện được vì có thể dẫn tới các vấn đề về performance
* Khi sử dụng Criteria API hay LINQ, NHibernate yêu cần các bảng cần có unique keys nếu không nó không thể read hay persists data xuống cở sở dữ liệu được.
* Select n+1 problem is considered carefully when developing module. This can be a substantial drain on the application but can easily go unnoticed. Fortunately the problem is relatively easy to fix and will generally be isolated to minor tweaks to your repository classes. The important thing is that none of the business logic needs to be modified to avoid this potential performance problem. This is one of the rare items that can easily be taken care of after the fact without disrupting the rest of your application.

Các vấn đề liên quan tới CSLA.NET:

* The idea of CSLA.NET is child-parent relationship. A child object is contained by a parent object. Conversely, a parent object is an object that contains one or more child objects. The top-most parent of a group of objects is the root object, and that is the object that can be directly retrieved or updated into the database. Any child objects contained by that root object are retrieved or updated at the same time. Nếu data model hiện tại không tuân theo cơ chế này thì cần phải điều chỉnh lại design hoặc cần phải cân nhắc module này có thể được phát tiển mà không tuân theo ADD.