Experiences from a Brazilian Bank Reengineering Project

Kellyton dos Santos Brito
Exact Sciences Department
Feira de Santana State University
Feira de Santana, Brazil
kellyton.brito@gmail.com

Vinicius Cardoso Garcia
FACITEC - Pernambuco State University, and
Informatics Center – Pernambuco Federal University
Recife, Brazil
vcg@cin.ufpe.br

Silvio Romero de Lemos Meira
Informatics Center - Pernambuco Federal University, and
Recife Center for Advanced Studies and Systems (C.E.S.A.R)
Recife, Brazil
silvio@cesar.org.br

Abstract - The migration of legacy mainframe applications to new web technologies is a challenge faced by several organizations. Since 2005, the Pitang and C.E.S.A.R companies are involved in a large migration project for a bank institution, aiming to migrate NATURAL/ADABAS legacy mainframe source code to a web-based platform. In this paper, we briefly describe the project’s evolution and lessons learned.

Reverse Engineering, Legacy Systems, Software Migration, Software Understanding.

I. INTRODUCTION

Due to the emerging of new technologies, the constant changes in business requirements and the well-known software aging problems [1], organizations are migrating their mainframe-based legacy systems to new modern environments, aiming to reduce costs and to increase the quality and time to market of new features.

In this scenario, since 2005 the Pitang Software Factory and the Recife Center for Advanced Studies and Systems (C.E.S.A.R) are working together with academic institutions in a five-year project to migrate NATURAL/ADABAS legacy systems of a Brazilian bank to new web-based technologies.

In this paper we briefly describe the lessons learned in this project as follows. In Section 2 we present the project. Section 3 describes the experience acquired in the three project phases, and in Section 4 we summarize the lessons learned and future directions.

II. THE PROJECT

The main requirements of this project concern to understanding and re-documenting the legacy NATURAL/ADABAS source code, and then re-implementing it as web applications. The client of this project is a Brazilian bank institution. In order to maintain the consistency with the client’s other development projects, the new web applications must be compliant with the architectural model defined by the bank.

To perform this project, we set up two teams: (a) a senior group of NATURAL/ADABAS developers with previous experience in bank/financial systems to perform the reverse engineering activities (RE team), and (b) a group of JAVA developers with previous experience on web development, to perform the forward engineering activities (FE team).

III. PROJECT PHASES

The project can be divided in three phases, each one with a different approach.

In the first phase, the used approach was based on the COGNOS Business Intelligence (BI) tool [2]. The tool was chosen because it allows the analysis of huge amounts of source code with extreme efficiency, which is similar to the work of Meinen[3]. In this phase, we internally developed a parser to read all source code statements and load it into the tool’s structure, which allowed us to perform several analysis, metrics and reports on the source code. In addition, built-in tools of the NATURAL/ADABAS environment, such as XREF, LIST and SCAN commands, were naturally used, since that the staff is familiar with these commands. Then, with the metrics and reports, the reverse engineering group performed the system re-documentation, which was then transferred to the forward engineering group to help in the re-implementation.

Despite the incontestable assistance of the BI tool, code metrics and reports alone did not lead to a full understanding of the source code, since we noticed that several times the reading of source code was necessary. Therefore, an environment which facilitates this visualization, with previous analysis of the code and suggestions for the user, is essential.
Aiming to resolve this question, in the project’s second phase we developed LIFT – Legacy InFormation retrieval Tool, which first version was already documented elsewhere [4]. The tool’s goal is to aid in reverse engineering and system documentation, and its main functions are (i) program slicing, (ii) generation of call graph and isolation of complete and minimal flows from screens to database modules, (iii) cluster detection, (iv) integrated visualization of source code, (v) identification and visualization of source code comments, (vi) reports generation and (vii) documentation generation. In this phase, the tool supported the reverse engineering activities, while the forward engineering process was not modified.

The analysis of this phase showed that the usage of the LIFT tool increased the productivity of reverse engineering activities in an average of 53.5%, according to the Lines/Hour metric [4]. Nevertheless, we verified that the approach could be improved, by adding better support to screen and database understanding, and by better integrating reverse and forward engineering activities.

Thus, we are performing the project’s third phase. LIFT was enhanced with the functionalities for (i) visualization of screen layouts and (ii) visualization and automatic migration of database structure. In addition, the process was modified in order to integrate some activities of reverse and forward engineering.

The new process, depicted in Figure 1, is performed as follows: First, with the assistance from COGNOS and LIFT, the database and legacy code are analyzed, and model classes automatically generated. Then, the Grails Framework [5] is used to generate a draft application. The framework receives the model classes and automatically generates a web-based MVC application. With these automated steps, a lot of work is avoided, by the generation of a three-tier application (from database to web pages) containing CRUD operations (Create, Retrieve, Update and Delete) of the main application entities. Then, we start the recognition of business logic: again with LIFT’s assistance in detecting clusters, visualizing screens and analyzing source code, the reverse engineers identify, document and transfer the business logic information to the forward engineering team, which modifies the newly generated application, inserting the business logic and adapting it to make it compliant with the client’s architectural model.

Although we still have no data to attest the effectiveness of the current approach, observation demonstrated the increase of global productivity, not only in reverse engineering activities, but by the integration of two teams, and by the automation of the initial forward engineering steps.

IV. LESSONS LEARNED AND FUTURE DIRECTIONS

In this paper we briefly presented the evolution of a project that aim to migrate legacy mainframe NATURAL/ADABAS applications to a web application.

In the project, we identified that the use of a BI tool, which provides metrics and reports about the code, is very important but not sufficient. In addition, tools to help in code visualization and manipulation, as well as automatic inference, can produce substantial productivity increases. Third, the integration between reverse and forward engineering teams and activities presented as a final step to provide good reengineering practices.

Despite these advances, we identified some points of the approach that can be improved:

(i) LIFT tool avoids the need to understand a lot of lines of code by identifying “orphan modules”. However, legacy systems have a lot of clone code, and the previous identification and detection of these clones can reduce the amount of code analyzed, increasing the productivity.

(ii) We identified the need for a tool that manages the migration. For example, such tool could provide visual representation of code pieces that were already migrated or understood, such as modules, clusters or clone code. In addition, it could collect metrics about migration, such as the relation about size, complexity and effort spent in understanding and migration tasks.

REFERENCES