Solutions for Reverse Engineering 4GL Applications, Recovering the Design of a Logistical Wholesale System

Csaba Nagy, László Vidács, Rudolf Ferenc, Tibor Gyimóthy
University of Szeged, Hungary
Department of Software Engineering,
Research Group on Artificial Intelligence
{ncsaba,lac,ferenc,gyimi}@inf.u-szeged.hu

Ferenc Kocsis, István Kovács
SZEGED Software Zrt., Hungary
{kocsis.ferenc, kovacs.istvan}@szegedsw.hu

***Abstract*—Re-engineering a legacy software system to support new, modern technologies instead of old ones is not an easy task, especially for large systems with a complex architecture. The use of reverse engineering tools is crucial for different subtasks of the full process, such as re-documenting the old code or recovering its design. There are many tools available to assist developers, but most of these tools were designed to deal with third generation languages (e.g. Java, C, C++, C#). However, many large systems are developed in higher level languages (e.g. Magic, Informix, ABAP) and current tools are not able to support all the arising problems during re-engineering systems written in fourth generation languages.**

In this paper we present a project whose main goal is the development of a technologically and functionally renewed medicinal wholesale system. This system is developed in Magic 4GL, and its development is based on re-engineering an old Magic (version 5) system to uniPaaS, which is the current release version of Magic. In the early phases of this project we developed a reverse engineering toolset for Magic 4GL to support reverse engineering, recovering the design of the old system, and to support some forward engineering tasks too. Here we present a report on this project that was carried out in cooperation with SZEGED Software Zrt and the Department of Software Engineering at the University of Szeged. The project was partly funded by the Economic Development Operational Programme, New Hungary Development Plan.

***Keywords*-architecture reconstruction, fourth generation languages, Magic 4GL, uniPaaS, re-engineering**

I. Introduction

Fourth generation languages (4GLs) are also referred to as very high level languages. A developer who develops an application in such a language does not need to write 'source code', but he/she can program his/her application at a higher level of abstraction and higher statement level, usually with the help of an application development environment. These languages were introduced and widely used in the mid-1980s. At that time many 4GLs were available (such as Oracle, FOCUS, RAMIS II and DBASE IV), but today most of the information systems are developed in third generation languages. However, large systems developed earlier in a 4GL are still evolving and there is still a continuous need for RADD (Rapid Application Development and Deployment) tools, which are usually based on higher level languages.

There are considerable advances in the design of 4GLs and their development environments. For instance, these languages usually offer ready solutions for common problems in developing a typical business application (e.g. connecting to database, supporting different database management or operating systems, managing data, etc.). On the other hand, one disadvantage of such a language is that it makes the developers depend on a particular vendor. This dependency may prevent the customers from taking advantage of the benefits of new technologies that are not supported by their vendor. New releases of the application development environment may support new technologies, but migrating from a previous version to a new one is not always automated. Performing the migration manually is often not economically feasible and in order to reduce costs of the migration or re-engineering, it is important to develop tools which will automate processes and assist the developers [1].

In this paper we report on a project whose main goal is to develop a re-engineered, technologically and functionally renewed, high market value logistical, medicinal wholesale system. The system was originally developed in Magic version 5 and it is re-engineered to work under uniPaaS, which is today the newest release of the Magic application development environment. Since the re-engineering process includes sophisticated reverse engineering and design recovery technologies, the outcome of the project is not only the re-engineered application, but also techniques and tools for recovering the design of fourth generation languages.

Here we report on the project and briefly introduce the new reverse engineering technology for 4GL as well. In Section II we give a brief overview on related work and related projects, then in Section III we describe the specialties of Magic 4GL and our design recovering methodology. In Section IV we describe the details (funding, duration, etc.) of the project, and finally, in Section V we conclude.

II. Related work and projects

In the literature only few papers are available in the area of reverse engineering fourth generation languages. Some of these papers address software quality, because when 4GLs became popular, some studies argued in favour of their use and some against them. These studies tried to predict the size of a 4GL project and its development effort, for instance by calculating function points [3] or by combining 4GL metrics with metrics for database systems [4]. In our previous paper [5] we introduced a quality-assurance framework (***MAGISTER***) for 4GLs which is able to continuously monitor the development process of a Magic application by measuring metrics and identifying coding rule violations.

An early paper for recovering the design of 4GL information systems was published by Harrison et al. in 1998 [6]. They described a tool to assist the recovery of both the application semantics and the static schema definition from Ingres ABF 4GL applications. They used the recovered design components to migrate from ABF 4GL to Oracle Designer 2000. Another migration tool is named M2J[1] which automatically converts a Magic application (written in newer versions of Magic, such as eDeveloper or uniPaaS) to Java. A solution for converting applications from previous Magic versions to uniPaaS is offered by Kopel Reem Ltd. as a professional service. Their solution is based on similar concepts as our approach, however our main goal is not only to assist conversion. Beyond presenting several different architectural views to developers, our aim is to facilitate general program comprehension by providing a toolset to query and investigate Magic applications. There are some tools available for testing and for optimization purposes too, e.g. Magic Optimizer[2], which also provide a set of views (e.g. cross references, UML charts) of a system.

III. Design recovery in the Magic environment

In this section we place emphasis on the first part of the whole project, where we have the main contributions related to the reverse engineering of Magic 4GL.

A. Design of a Magic application

Magic 4GL was introduced by Magic Software Enterprises (MSE) in the early 80's. It was an innovative technology to move from code generation to the use of an underlying meta model within an application generator. The resulting application was run on popular operating systems including DOS and UNIX. Since then newer versions of Magic have been released called *eDeveloper* and *uniPaaS*. Recent versions support novel technologies including RIA (Rich Internet Applications), SOA.

The heart of a Magic application is the Magic Runtime Engine (MRE), which allows one to run the same application on different operating systems. When one develops an application in Magic, one actually programs the MRE using the unique meta model language of Magic, which is – at a higher level of abstraction – closer to business logic. This meta model is what makes the development in Magic unique and what really makes Magic a RADD (Rapid Application Development and Deployment) tool.

Magic was invented to develop business applications for data manipulating and reporting, so it comes with many GUI screens and report editors. Hence the most important elements of its meta model language are the various entity types of business logic, namely the data tables. A table has its columns and a number of programs (consisting of subtasks) that manipulate it. The programs or tasks are linked to forms, menus, help screens and they may also implement business logic using logic statements e.g. for selecting variables (virtual variables or table columns), updating variables, conditional statements and expressions.

The meta model of a Magic application serves as a 'source code' that can be analyzed.

B. Design recovery in Magic environment

The design recovery process (see Figure 1) involves reverse engineering the Magic application and presenting several different architectural views to the developers. These views can be further investigated to determine functional or logical components or simply query language entities and their relations.

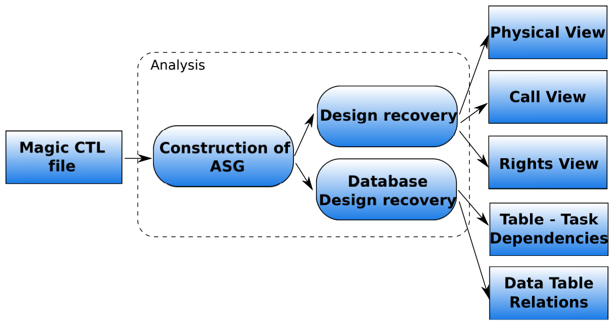


Figure 1. Overview of the design recovery process.

1) Fact extraction: The reverse engineering step is a Magic adaptation of the *Columbus reverse engineering methodology* [7]. In this step we precisely describe the syntactic and semantic characteristics of Magic by defining language elements and their relations with a meta-model, called *Magic Schema*. This schema determines the structure of the *Abstract Semantic Graph* (ASG). The ASG of a Magic application is constructed by a parser called *MagicASG* which takes as its input the CTL (*control*) export of the application development environment. Note, that developers do not write code in the traditional way, but they edit tables of the development environment. Therefore the CTL export represents a textual export of the actual state of the environment. This export is hard to read and comprehend for humans, but it can be parsed and contains all the necessary information for constructing an ASG.

2) Design recovery: The design recovery process involves analysis techniques in order to present the gathered information via higher level views to the developers. These views are the following:

Physical view. This view presents the structure of the application. Here we identify relevant language entities (data

[1]http://www.magic2java.com
[2]http://www.magic-optimizer.com

tables, columns, programs, tasks, logic units, etc.) and their parent-child relation that determine the main structure of the application, like packages, classes and methods define the structure of a Java application.

Call view. This view presents the call-graph of the system.

Menu view. In Magic applications a Menu is an entity of the language too. A Menu can fire system/user events, or can call a program within the application. The executed program can call other programs or subtasks. Hence in this view we extend the Call view with the menu entities and their program call relations.

User rights view. Magic offers ready solutions for user and user role management. In Magic one can define *rights* which describe the role to access menus, programs or data tables. One can also define users and user groups and these users or groups may have a number of previously defined rights. In this view we present the information whether a user has access to a menu/program/table or not.

3) Database design recovery: Magic applications strongly depend on their databases, hence, it is important to take into account the database dependencies during the design recovery process too.

Table-Task dependencies. We identify relations between data tables and all those tasks and programs that use the specified table. We differentiate between *create*, *retrieve*, *update*, and *delete* relations. Using this view, one can easily identify language elements working on the same data table. This can be a powerful tool for identifying e.g. logical components in the system.

Data Table relations. In older Magic versions *foreign key*s were not supported and even in new versions one can develop his application without using them. The only way to determine the relations between data tables is to analyze the application logic and identify those parts of the code where they link together two or more tables. Here we determine if two tables are in *one-to-one*, *one-to-many*, or *many-to-many* relations and we identify the columns that were used for linking them together.

IV. The Project

A. General details

The presented project is a sub-project of the larger research project MAGISTER[3] aiming at defining a framework that provides software quality assurance, reengineering, and testing services for applications developed in Magic fourth generation language (4GL).

The details of the project are the following:

- The ***MAGISTER-ARCH*** project is carried out in cooperation between SZEGED Software Zrt. and the Department of Software Engineering, University of Szeged.
- The project is based on the *"Support of enterprise innovation"* tender of the Economic Development Operational Programme, New Hungary Development Plan. The project is supported by the European Union and by the European Regional Development Fund.
- The project title is *"Development of a technologically and functionally renewed, high market value, logistical, medicinal wholesale system"*, and its registration index is: *GOP-1.3.1-07/1-2008-0026*.
- The total budget of the project is 362 400 EUR. The amount of funding is 181 200 EUR (50%).
- The project started in September, 2008 and lasted for 2 years.

[3]http://www.szegedsw.hu/magister/

B. Main tasks of the project

Figure 2 illustrates the main tasks of the project. First we prepared the introduced reverse engineering toolset and methodology for Magic 4GL (1) then we implemented design recovery tools and applied them on the old system (2). Based on the results and the investigated new technologies (3) we started the specification of the new system (4). After the implementation phases (5) we tested it (6) and started some marketing tasks (7).

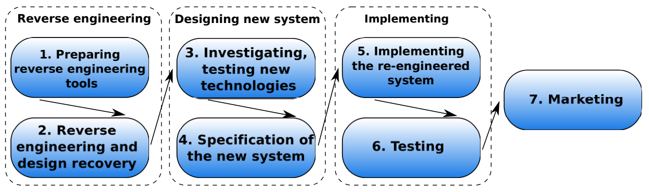

Reverse engineering
Designing new system
Implementing
1. Preparing reverse engineering tools
2. Reverse engineering and design recovery
3. Investigating, testing new technologies
4. Specification of the new system
5. Implementing the re-engineered system
6. Testing
7. Marketing

Figure 2. Main tasks of the project.

Preparing reverse engineering tools. We investigated the specialties of fourth generation languages, especially the specialties of Magic 4GL. We designed a methodology and implemented a tool to reverse engineer Magic 4GL applications. Here we focused on Magic version 5 and our goal was to construct a precise ASG (Abstract Semantic Graph) by analyzing the CTL export of the application development environment.

Reverse engineering and design recovery. We further investigated the specialties of Magic 4GL so we could recover the design of the system being analyzed. We note that these systems are usually database-intensive systems which were designed to store and modify their data model entities using databases. Hence, recovering the design of the data model and its relations to the application is also important here. We implemented the design recovering tools and applied them to the reverse engineered system.

Investigating, testing new technologies. Before finalizing the new specification of the system, it was important to perform tests on its environment too. The old system had many environmental dependencies and some new functionalities of the new system required integration of new tools and technologies too. It was important to see how efficient would the new system be in its new environment. New potential performance or functional problems arose here, some of them could be solved only with the assistance of MSE. We performed performance and stress tests on the selected tools (e.g. database manager) with large data and network traffic.

Specification of the new system. Here we used the results of reverse engineering, the new requirements and the results of testing new technologies. As the new programming environment offered several new features we had to redesign several parts of the system too.

Implementing the re-engineered system. We started the implementation by developing the core framework. That is, the user management, rights, data management, menus, most important background tasks, etc. Then we implemented the selected functionalities of the new system one-by-one.

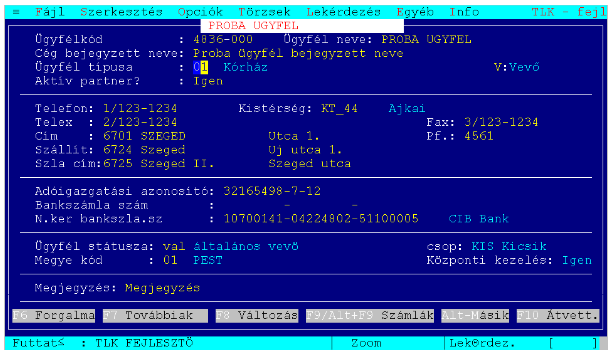
Testing. Testing was particularly important to assure the quality of the system. Here we performed functional, integration and general testing of the whole system as some of the previously implemented functionalities had a work-flow that affected almost the whole system.

Marketing. In the final phases of the project we start direct marketing tasks to advertise our new product.

C. Status of the project

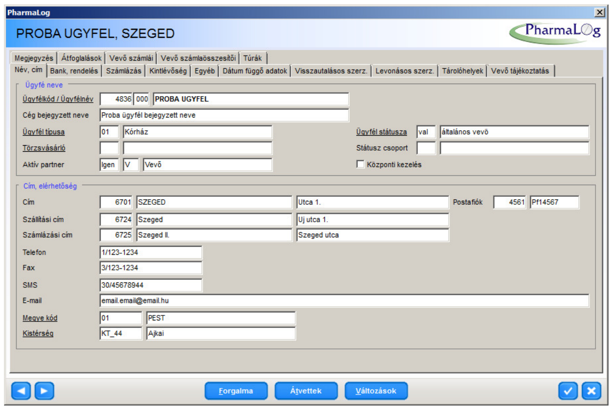
The project was successfully completed within the planned time frame. The participants of the project gained experiences in new technologies, but more importantly collected useful practices in reverse engineering and re-engineering in Magic environment.

Figure 3 is a screenshot of the old, reverse engineered system and Figure 4 is a screenshot of the same input form in the new re-engineered application.



```
Ügyfélkód           : 4836-000    Ügyfél neve: PROBA UGYFEL
Cég bejegyzett neve: Proba ügyfél bejegyzett neve
Ügyfél típusa       : 01  Kórház                               V:Vevő
Aktív partner?      : Igen

Telefon: 1/123-1234        Kistérség: KT_44    Ajkai
Telex  : 2/123-1234                                     Fax: 3/123-1234
Cím    : 6701 SZEGED          Utca 1.                   Pf.: 4561
Szállít: 6724 Szeged          Uj utca 1.
Szla cím:6725 Szeged II.      Szeged utca

Adóigazgatási azonosító: 32165498-7-12
Bankszámla szám        :          -        -
N.ker bankszla.sz      : 10700141-04224802-51100005    CIB Bank

Ügyfél státusza: val általános vevö                     csop: KIS Kicsik
Megye kód      : 01  PEST                               Központi kezelés: Igen

Megjegyzés: Megjegyzés

Futtat≤  : TLK FEJLESZTÖ                      Zoom          Lek©rdez.    [      ]
```

Figure 3. A screenshot of the old, reverse engineered application.



PROBA UGYFEL, SZEGED

Ügyfé neve

Ügyfélkód / Ügyfélnév	4836	000	PROBA UGYFEL		
Cég bejegyzett neve	Proba ügyfél bejegyzett neve				
Ügyfél típusa	01	Kórház	Ügyfél státusza	val	általános vevö
Törzsvásárló			Státusz csoport		
Aktív partner	Igen	V	Vevő	☐ Központi kezelés	

Cím, elérhetőség

Cím	6701	SZEGED	Utca 1.	Postafiók	4561	Pf14567
Szállítási cím	6724	Szeged	Uj utca 1.			
Számlázási cím	6725	Szeged II.	Szeged utca			
Telefon	1/123-1234					
Fax	3/123-1234					
SMS	30/45678944					
E-mail	email.email@email.hu					
Megye kód	01	PEST				
Kistérség	KT_44	Ajkai				

Figure 4. A screenshot of the new, re-engineered application.

V. Conclusions

Our primary goal was to re-engineer a complex Magic 4GL application. Besides the full re-engineering process, in this paper we focused on the reverse engineering steps. In these steps we made the following contributions. We have successfully adapted and tailored for 4GLs the Columbus RE technology, which was originally invented for object oriented languages. Using this technology we developed a methodology for recovering the design of legacy 4GL applications. This methodology defines architectural views which reflect the program structure, semantics and its relations to the underlying databases. Our methodology was considered useful by assisting program comprehension and re-engineering tasks of Magic 4GL applications in a successful project which was partly funded by EU. A remarkable outcome of our work is the tool-chain, which we plan to use for offering services in further projects such as migration, maintenance and program comprehension of Magic programs.

Acknowledgements

This research was supported by the New Hungary Development Plan, Economic Development Operational Programme, GOP-1.3.1-07/1-2008-0026.

Investing in your future
New Hungary Development Plan

References

[1] E. J. Chikofsky and J. H. Cross II, “Reverse Engineering and Design Recovery: A Taxonomy,” in *IEEE Software 7*, Jan. 1990, pp. 13–17.

[2] J. Verner and G. Tate, “Estimating Size and Effort in Fourth-Generation Development,” *IEEE Software*, vol. 5, pp. 15–22, 1988.

[3] G. Witting and G. Finnie, “Using Artificial Neural Networks and Function Points to Estimate 4GL Software Development Effort,” *Australasian Journal of Information Systems*, vol. 1, no. 2, pp. 87–94, 1994.

[4] S. MacDonell, “Metrics for Database Systems: An Empirical Study,” *IEEE International Symposium on Software Metrics*, pp. 99–107, 1997.

[5] C. Nagy, L. Vidács, R. Ferenc, T. Gyimóthy, F. Kocsis, and I. Kovács, “MAGISTER: Quality Assurance of Magic Applications for Software Developers and End Users,” in *Proceedings of the 26th IEEE International Conference on Software Maintenance*. IEEE Computer Society, Sep. 2010.

[6] J. V. Harrison and W. M. Lim, “Automated Reverse Engineering of Legacy 4GL Information System Applications Using the ITOC Workbench,” in *Proceedings of the 10th International Conference on Advanced Information Systems Engineering*. Springer-Verlag, 1998, pp. 41–57.

[7] R. Ferenc, Á. Beszédes, M. Tarkiainen, and T. Gyimóthy, “Columbus – Reverse Engineering Tool and Schema for C++,” in *Proceedings of the 18th International Conference on Software Maintenance (ICSM’02)*. IEEE Computer Society, Oct. 2002, pp. 172–181.