Transferring Software to a New Framework in a Brownfield Environment*

Pieter van der Spek, Steven Klusener
VU University Amsterdam, De Boelelaan 1081, 1081 HV Amsterdam
{pvdspek,steven}@cs.vu.nl

Abstract

This is an experience report on the architectural re-engineering of a large software portfolio. In order to minimize the risks of such an undertaking, companies tend to be conservative when considering their options. We have examined how this risk-averse attitude affects the project and we present the solution we have taken to counter these effects.

Index Terms: Re-engineering, Brownfield, Encapsulation, Migration

1 Introduction

Very few IT projects are delivered on "Greenfield" sites anymore. Most businesses already have a significant and complex IT environment. Some are so complex that they could be considered contaminated or polluted by their complexity [3]. Such environments are called *Brownfield* sites.

Although the costs and, especially, the risks associated with decontaminating such a software system are generally high [1], they cannot always be avoided, for instance when third-party components become end-of-life. In order to minimize the risks, companies tend to be conservative when considering their options.

In the context of an industrial reengineering project we have examined how the conservatism affects the project and chosen solution. Next to that, we provide our approach for countering this effect.

2 A real world modification example

The CLINAPP project was carried out at the business unit magnetic resonance imaging (MRI) of Philips Healthcare. In an effort to consolidate not only multiple user interfaces (300-500 KLOC each), which were in use at the business unit MRI, but also various other in-house developed user interfaces of other business units, a basic user interface framework was developed by an independent department within Philips Healthcare. The goal of the pre-study, which is described in this article, was to explore alternative solutions, identify potential pitfalls and provide the basis for an estimate of the amount of effort required to transfer the clinical post processing applications[1] like diffusion, angiography, spectroscopy and functional MRI to this new framework.

Summary of the system For the purpose of this paper, we have looked at a single clinical application used for post processing images coming from an MRI scanner. This application consists of nearly 90.000 LOC, written primarily in C#, spread over 22 building blocks [5]. The entire software archive consists of about 7 million lines of code mostly consisting of C, C++, and C#, spread over nearly 600 building blocks.

3 Analysis of the re-engineering problems

The approach we followed for this project covers three basic steps. We will be going into each of these steps and describe our experiences together with some of the findings we made.

Survey As legacy systems rarely have an accurately documented architecture [4] other sources of information have to be used as well in order to get an insight into the current system. Although diving into the source code can be useful, the amount of information is usually overwhelming. Therefore it is much more informative to start with interviewing system experts and running different test scenarios in order to get a high-level overview of the current system. Using this information, the current framework can be compared to the new framework.

As part of the survey, two possible scenarios were examined for transferring the existing software: encapsulation (wrapping) and migration. Although migration seemed

*This work has been carried out as a part of the DARWIN project at Philips Healthcare under the responsibilities of the Embedded Systems Institute. This project is partially supported by the Dutch Ministry of Economic Affairs under the BSIK program.

[1] An introduction into MRI and post processing can be found here: `http://en.wikipedia.org/wiki/MRI`.

a much more complex undertaking than wrapping, if successful, migration's long-term benefits are also greater. For example, migration offers more flexibility, better system understanding, easier maintenance, and, in the long term, reduced costs [2].

However, the focus of the survey was on the mismatch between the old and new framework, even though the overlap between the two was much larger. In this way, the risks of migration were exaggerated and the opportunities were overlooked as we will see later.

Isolate After the initial survey, the next step in the project was to isolate the core of the clinical application, which could then serve as a starting point for identifying its dependencies and transferring the application to the new framework.

Figure 1 shows conceptually how the core is surrounded by a layer of dependencies. Depending on the way in which the application is transferred to the new framework, this layer can be thicker (wrapping) or thinner (migration). However, a natural preference exists in favor of keeping this "insulating" layer thick in order to stay away from the business logic of the application and limit any risks due to changes to the core.

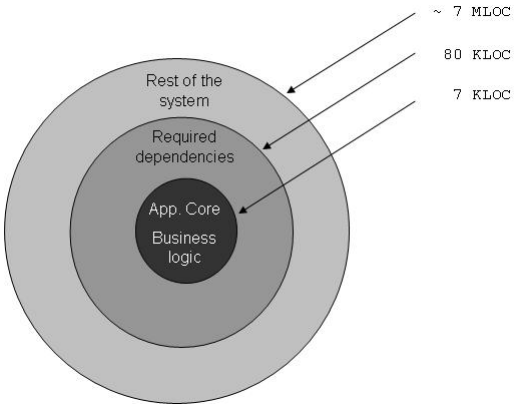

Figure 1. Dependency layers.

Transfer In [6] Sneed mentions several risks involved in reengineering projects. In fact, the bias in favor of wrapping can be considered an instance of "user program rejection". Due to the unfamiliarity with the new framework, the possible benefits of migration and the impact migration would have on the business logic were not clear. In order to overcome this risk, we argued that each scenario should be examined 'hands-on', i.e. users should get hands-on experience with the new framework creating a prototype using wrapping and migration both.

In the case of the clinical application, we performed the migration and showed that the migrated application consisted of 10 KLOC (i.e. the business logic + **some of the** required dependencies in Figure 1) as opposed to 90 KLOC for the wrapped application (i.e. the business logic + **all of the** required dependencies in Figure 1). Although a reduction in size was expected, the extent came as a surprise. As the initial survey had focused on the mismatches, the amount of overlap between the old and new framework, had been underestimated. The new framework thus provided the opportunity of reducing the maintenance effort. However, what the pilot study also showed was that the migration was more time consuming as well as more risky compared to wrapping (although not to the extent which was initially expected).

All things considered, the architects involved agreed that migration could actually be a viable solution. In the end wrapping was still chosen for performing the transfer, but migration is now considered for the clinical applications as part of regular maintenance.

4 Conclusion

In projects such as the CLINAPP project the people involved tend to be risk-averse and choose the safest solution. The potential payoff of other solutions is outweighed by the perceived additional risk. Unfortunately, this also means that only the safest solution is examined in any detail. In the case of the CLINAPP project, migration was not seriously considered until the extent to which the lines of code could be reduced and the architectural match was shown.

Sneed [6] mentioned that one way of reducing the user program rejection is to involve the users in the reengineering effort. From the experiences in the CLINAPP project we have learned that, in order to remove the initial rejection of a reengineering solution, users should be involved in one or more pilot studies to get a first-hand feel for the possible merits of each solution.

References

[1] L. Aversano, G. Canfora, A. Cimitile, and A. de Lucia. Migrating legacy systems to the web: an experience report. In *CSMR'01*, page 148, Washington, DC, USA, 2001. IEEE Computer Society.

[2] J. Bisbal, D. Lawless, B. Wu, and J. Grimson. Legacy information systems: Issues and directions. *IEEE Softw.*, 16(5):103–111, 1999.

[3] R. Hopkins and K. Jenkins. *Eating the IT Elephant - Moving from Greenfield Development to Brownfield*. IBM Press, 2008.

[4] R. Kazman and S. J. Carrière. Playing detective: Reconstructing software architecture from available evidence. *Autom. Soft. Eng.*, 6(2):107–138, 1999.

[5] T. Rötschke. Re-engineering a medical imaging system using graph transformations. In *AGTIVE 2003*, pages 185–201, 2003.

[6] H. M. Sneed. Risks involved in reengineering projects. In *6th Working Conference on Reverse Engineering*, pages 204–212, 1999.