NOREX: Distributed collaborative reengineering

Mihai Balint[1], Petru Florin Mihancea[1], Radu Marinescu[1], and Michele Lanza[2]

[1]LOOSE Research Group, Politehnica University of Timişoara, Romania
[2]Faculty of Informatics, University of Lugano, Switzerland

Abstract

Several reengineering environments have been created to provide a unified infrastructure in which various approaches can be employed together. These environments are typically implemented in different languages and use similar yet distinct architectures. While the collaboration between tools is facilitated within such environments, collaboration across their boundaries happens at most at the level of data exchange using files in a specific format, such as XMI and GXL. Consequently, the different groups of researchers are only collaborating shallowly via data, rather than at the level of analysis. In this paper, we present our vision of a distributed service-based reengineering environment that allows different groups of researchers to transparently use and combine existing analysis techniques shifting focus from tool development towards idea development and facilitating the expression of more complex techniques in a fully reusable manner.

Keywords: reengineering, web services, distributed systems

1 Introduction

Reengineering is a complex task that requires several techniques to be employed together [3]. The larger the data at hand, the more techniques we require to manipulate and to reason about the data.

With new reengineering analyses constantly being proposed and implemented, it is highly desirable for researchers to be in a position to use these state-of-the-art techniques, both for comparison of research results, and for reuse of engineering effort. Hence, several environments have been created to provide a unified infrastructure in which various approaches can be employed together [6].

While environments foster collaboration within their boundaries, they are isolated from one another, analyses built in one of them are not reusable in another one. Communication between environments is confined to data transfer via exchange formats [4, 7]. The main cause of this isolation resides in that environments are oftentimes implemented in different programming languages, and even when they are implemented in the same language they have a dedicated infrastructure (*e.g.,* different meta-models, different front-ends).

In this context, we started the *NOREX* project[1], in order to develop an infrastructure that would support research as it unfolds in the current academic and industrial ecosystem. We aim *NOREX* to have following traits: (a) it should be *distributed i.e.,* it should keep the implementation of reengineering techniques near their creators while making them accessible to everyone without the need of reimplementation; (b) it should be *language independent i.e.,* it should allow the writing of techniques in any programming language while enabling their reuse from any particular programming language; (c) it should be *community focused*, meaning that it centers on the user's needs: researchers combine, compare, develop and enhance techniques; reengineers use techniques for analyzing software systems by employing various techniques developed by researchers. Yet, both sides need the convenience of using their familiar environments and programming languages. Furthermore, we are aware that they wouldn't like to invest into integrating different reengineering tools.

2 Collaborative Reengineering

Reengineering research consists of creating new techniques either from scratch or by combining existing ones and it usually involves the creation or extension of the metamodel entities. These extensions and the fact that sometimes the technique is tightly coupled with the metamodel implementation represent the greatest barrier in the face of approach portability.

To overcome this hurdle, research groups have tried to unify metamodels [1] and to create metamodel transformations [2]. However, metamodel transformations are a temporary solution because metamodels evolve as techniques do, and why would we confine researchers to work with a single metamodel?

Although reengineering is unlikely to adopt a common metamodel, we envision a framework that enables researchers

[1]see http://loose.upt.ro/norex

to *choose* the model of software that best suits them is already here. In this context, *NOREX* allows researchers to keep their programming language and their favorite reengineering environment. It also allows them to use others techniques in their own environment and to publish their techniques so that others may use them in their environments.

To achieve this level of flexibility the architecture of today's reengineering tools (see Figure 1) needs several types of adaptation:

- *Decouple the tool's services*. Reducing the coupling between techniques (parsers, analyses) and metamodel implementations ensures that the techniques can be used on different language implementations of the metamodel. For example, decoupling the browser from a particular metamodel will allow it to navigate any metamodel.
- *Specify explicit interfaces*. If each technique provides a structured specification of it's interfaces and of the data structures it uses, it is easier to reuse and understand what it does, hence it is likely that it will get noticed and used by other researchers.
- *Integrate with the community*. *NOREX* implements a means of publishing the services provided by a tool on a NOREX *server* so that they become available for the whole community. It also provides a means of accessing services published elsewhere in the community.

By reusing instead of reimplementing, reengineering is likely to advance faster and our tools are likely to become less interesting as case studies and more interesting as research tools.

3 Current Achievements

Currently, we have implemented a prototype of the *NOREX* reengineering infrastructure, and defined precise methodologies for defining/adapting reengineering techniques in form of sharable *NOREX* services and for using these services. As part of this process, we have created a *NOREX* server implementation and have published several services (*e.g.,* a set of software metrics and metrics-based rules for detecting design flaws related to size and complexity, coupling and inheritance relations, described in [5]). Additionally, we have published more complex analysis, like polymetric software visualization (*e.g.,* the *System Complexity View*[5]). Furthermore, we are in the process of adapting the two analysis environments developed in the past by our research groups (*i.e.,*MOOSE [6] and IPLASMA [5] to make them fully compliant with the *NOREX* infrastructure.

4 Conclusions

We envision that giving researchers the choice of which model of software to use will inevitably lead to the creation of a metamodel market economy, and in time some metamodels will raise as the most useful for certain applications. In this market metamodels will compete in areas like expressiveness, technique richness, community support and popularity. Also a community such as MOOSE community will be able to share their approaches with other researchers thus tightening relations with other research groups.

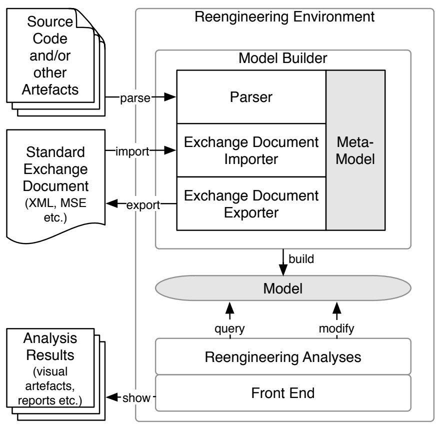
Source Code and/or other Artefacts
Reengineering Environment
Model Builder
parse
Parser
Standard Exchange Document (XML, MSE etc.)
import
Exchange Document Importer
Meta-Model
export
Exchange Document Exporter
build
Model
query
modify
Analysis Results (visual artefacts, reports etc.)
Reengineering Analyses
show
Front End

Figure 1. The Architecture of a Generic Reverse Engineering Environment.

Acknowledgments. We gratefully acknowledge the financial support of the Swiss National Science Foundation for the project "NOREX — Network of Reengineering Expertise" (SNF Project IB7320-110997).

References

[1] A. Alvaro, D. Lucrdio, V. C. Garcia, A. F. do Prado, and L. C. Trevelin. Orion-re: A component-based software reengineering environment. In *WCRE*, 2003.
[2] D. Jin and J. R. Cordy. Integrating reverse engineering tools using a service-sharing methodology. In *Proceedings of ICPC'06*. IEEE Computer Society, 2006.
[3] S. Demeyer, S. Ducasse, and O. Nierstrasz. *Object-Oriented Reengineering Patterns*. Morgan Kaufmann, 2002.
[4] R. C. Holt, A. Winter, and A. Schürr. GXL: Towards a standard exchange format. In *Proceedings WCRE '00*, Nov. 2000.
[5] M. Lanza and R. Marinescu. *Object-Oriented Metrics in Practice*. Springer Verlag, 2006.
[6] O. Nierstrasz, S. Ducasse, and T. Gîrba. The story of Moose: an agile reengineering environment. In *Proceedings of the European Software Engineering Conference (ESEC/FSE 2005)*, pages 1–10. ACM, 2005. Invited paper.
[7] Xml metadata interchange (xmi), v2.0, 2005. http://www.omg.org/cgi-bin/doc?formal/05-05-01.