# Network Of Reengineering Expertise (NOREX)
# SNF SCOPES/JRP Project IB7320-110997

Scientific Report: November 2005 - October 2006

Michele Lanza[1], Radu Marinescu[2] and Oscar Nierstrasz[3]

[1]University of Lugano, Faculty of Informatics, Switzerland
[2]Institute e-Austria Timişoara, Romania
[3]University of Berne, Software Composition Group, Switzerland

## 1 Overview of Research Activities

The goal of of this joint research project is to provide a comprehensive and extensible support for complex reengineering activities applicable on real-world systems. The project aims to answer the following questions:

1. How can all the heterogenous sets of existing reengineering artifacts (*e.g.*, tools, models, types of analyses) be connected and used together in a uniform reengineering environment?

2. How can a uniform reengineering environment support the automation of complex reengineering activities? Which are the most relevant reengineering "workflows" to support?

Specifically, we address these issues by building a distributed reengineering environment which is able to make all the techniques and models defined and implemented by each of the three research teams complement each other. Furthermore, we use this environment to integrate different reengineering techniques to support complex reengineering techniques and validate, based on large-scale experiments, the feasibility of the approach. In this report we review the first year of the project and reflect the results and prototypes developed so far. The major achievements can be summarized as follows:

1. Conception and implementation of communication protocol for the NOREX reengineering environment.

2. A distributed registration mechanism for heterogenous reengineering tools.

3. A suite of reengineering patterns that combine problem detection and software visualization techniques

4. Tools for visualization (Mondrian and its Java version JMondrian) and in-depth design analysis of class hierarchies (Membrain)

The communication protocol provides a proper means to ensure the needed exchange of information between the provider of a reengineering service (*e.g.*, metrics, concept analysis *etc.*) and the client of that service, *i.e.*, the user of the NOREX environment that wants to apply the service on a given case-study. The registration mechanism together with the communication protocol, both designed and implemented during the first year of this project, ensure the backbone of the NOREX reengineering environment. In parallel with creating the NOREX environment, we worked on finding proper means to correlate metrics-based design flaw detection techniques (*e.g.*, metrics and detection strategies) with software visualization techniques (*e.g.*, polymetric views). Furthermore, the creation of new tools – which are conceptually compatible with the NOREX environment – that support these more complex reengineering activities enable us to come significantly closer towards our vision of supporting large-scale reengineering "workflows". In the following, we discuss these results based on the work package structure of the project.

## 1.1 Research Activities and Results

### 1.1.1 WP1: Creating a Distributed Reengineering Environment

In this workpackage we are building a prototype of the NOREX environment. The first task of the environment is to ensure that different tools can understand each other by understanding the semantics of their models. The semantics of a model are expressed in its meta-model. Therefore, we first specified a common meta-meta-model, using the implementation of MOOSE and iPlasma as a starting point as they already have implemented similar meta-meta-models. Furthermore, the tools need to be implemented in different languages. Consequently, NOREX needs to ensure that the different tools are able to communicate with each other.

**Achievements.** The main achievement of this workpackage is the creation of a prototypical version of NOREX. Our main goal with NOREX is to allow for different groups to build tools in a loose fashion, that is without imposing hard-coded dependencies between these tools. Therefore, the solution that we came up with is based on the following approach: each reengineering tool or service that is provided to the other nodes of the network, will decouple the actual "algorithm" from the data (*i.e.*, the meta-model specific part) on which it performs. Thus, each service will express its dependency on a meta-model in terms of a set of Command objects [GHJV95], each one representing a particular information retrieval action performed on the concrete model. In other words, a service will describe *what* information it needs to run, but it will completely decouple from *how* this information is provided by one or the other of its clients. Part of this achievement is the fact that we defined a common meta-meta-model that allows us to abstract from the particularities of the various meta-models used by our groups. Thus, we can describe any of these meta-models in terms of several major concepts (*e.g.*, *entities* having *entity types*, *group entities* as a special type of *entity* etc.). While we plan to dedicate a paper to the entire construction of NOREX, the idea of a common meta-meta-model is partially described in [DG06]. Furthermore, in [NDG05] we emphasize the benefits of an"agile" reengineering environment.

Using the common meta-model, we defined a so-called *service manifesto* by which a reengineering service describes itself to its clients. The *service manifesto* has the following structure:

1. Name of service (unique identifier)
2. Free text description
3. Entity type identifier on which the service is defined
4. List of commands objects. For each of these, the following information is specified: The name of command (unique identifier), a free text description of the command's semantic, the type of command (*e.g.*, CollectionCommand, FilterCommand, NumericCommand etc.), and the entity type identifier

One important advantage of this approach is that it provides a *"lazy mapping"* between the meta-models used in the different nodes (*i.e.*, by the different groups) of the network. To use a NOREX service, one needs only to bind the entity type and command identifiers from the *service manifesto* to concrete meta-model objects.

The second significant advantage of this approach is its incremental nature *i.e.*, a binding must be done only once. Consequently, the more NOREX services are used the less extra-work is expected before the actual usage of the service (as it is very probable that most, if not all the bindings are already encoded).

**Status of Deliverables**

- (D1.1) The common meta-meta-model was completely specified and implemented. The fact that both MOOSE and iPlasma already use a very similar meta-meta-model has simplified this task.
- (D1.2) The communication protocol is currently completely specified. Concerning its implementation, this is currently accomplished for Java, and is only 1-2 weeks away from being finalized in Smalltalk, due to the straightforward specification and based on the lessons learnt while doing the implementation for Java.
- (D1.3) The design and implementation of the registration mechanism for the reengineering tools has been fully completed. We are entering now a thorough performance testing phase that we expect to lead us to further optimizations (also applicable for deliverable D1.2).

**People.** Dr. Tudor Gîrba (Bern) together with Prof. Lanza (Lugano) and Prof. Marinescu (Timişoara) have designed the common meta-meta-model, the registration mechanism and the communication protocol. Researcher Mihai Balint and Ph.D. students Petru Florin Mihancea and Cristina Marinescu (all from Timişoara) have done most of the implementation work on NOREX. Ph.D. student Dan Cosma (Timişoara) has supervised the WebService specific design and implementation decisions. Mircea Lungu (Lugano) is looking into an implementation of the NOREX mechanisms in Smalltalk.

### 1.1.2 WP2: Enabling Complex Reengineering Activities

Using the NOREX environment we explore the possibilities to build complex analyses by combining tools built by the participating groups. In particular, during the first year we focused our attention on combining problem detection with visualization to get an overview of how design problems are distributed over a system.

**Achievements.** The main achievements of this workpackage are related to a significant series of publications, most of these being joint publication between the partners involved in the current project. The most important publication, co-authored by Prof. Lanza (Lugano) and Prof. Marinescu (Timişoara) is the book (published by Springer Verlag in July 2006) entitled "*Object-Oriented Metrics in Practice*" [LM06] in which metrics-based detection strategies and visualizations are used together in form of a *catalogue of patterns* for understanding and assessing the design quality of object-oriented systems.

Apart from this book, the members of the three teams have published several papers on various types of reengineering activities. In [Mih06] we developed a novel technique for analyzing the quality of class hierarchies. In [TM05] we extended our initial problem detection approach by taking into consideration contextual hints that would lead to an adequate restructuring. In [Mar06] we show how both code understanding and problem detection approaches need to be adapted for enterprise software systems. In [GD06] we present how the history of a system's design can be modelled so that history related analyses can be easily combined with classic analyses based on a single version of a system.

As one of our goals is to integrate in the NOREX environment efficient visualization techniques we developed advanced tools for visualization like Mondrian [MGL06] (and its Java version JMondrian) and published several papers on novel visualization techniques. Thus, in [BGM06] we investigate the correlation between code duplication and the authors of those duplications, and propose a visualization technique that reveals several such correlation patterns. In [LLG06] we show how the architecture of a system can be recovered by means of visual patterns related to the structure of packages using a prototpye [LL06] that we plan to integrate into the NOREX platform.

**Status of Deliverables**

- (D2.1) The catalogue of reengineering patterns defined in [LM06] constitutes the first form of this deliverable. Several other candidates (*e.g.*, combination of evolution analysis and problem detection techniques) are also in advanced stage of being added to the final form of the catalogue.

- (D2.3) In the last month of this first project year, the work on the tool prototype for simulating refactorings on the model level has started. The major discovery that we made so far, is the need to ensure a more detailed (fine-grained) representation of a method's body (we called them *"code stripes"*) than the one currently existing in our meta-models.

**People.** Prof. Lanza and Prof. Marinescu have devised the structure of the catalogue of complex detection techniques for design flaws and helped by Dr. Tudor Gîrba (Bern) and Ph.D. students Petru Florin Mihancea and Cristina Marinescu have defined a first set of such techniques. Dr. Tudor Gîrba and Ph.D. student Mircea Lungu, together with Prof. Lanza have been involved in the development of Mondrian the novel visualization framework that can be much better integrated in the NOREX environment. In Timişoara Mihai Balint and Petru Florin Mihancea worked on creating JMondrian, by porting Mondrian to Java.

### 1.1.3 WP3: Performing Large-Scale Reengineering Experiments

**Achievements.** The main achievement of this workpackage is that we have been able to identify so far two case-studies that are large enough to prove the feasibility of the NOREX platform. One of these is the ArgoUML[1] modelling tool, written in Java. The other one is a very-large C++ proprietary system from one of our industrial partners that we also intend us in our experiments. In addition to these two systems, we identified a third medium-size case-study written in Java that we intend to use especially for the experiments with analyses dedicated for enterprise applications.

**Status of Deliverables.**

- (D3.1) As mentioned, we have successfully found 3 case-studies, and consequently we consider that the deliverable is complete. Yet, we intend to stay open to any new potential case-study that would fit our needs.

**People.** Prof. Lanza and Prof. Marinescu, helped by the other members of their teams have chosen the first two case-studies. The third case-study, *i.e.*, the enterprise system, was chosen by Cristina Marinescu (Timişoara).

## 1.2 Collaboration Progress

The NOREX project must be considered very successful in terms of collaboration: The three involved research groups entertain a closely knit net of personal and professional relationships which is fruitful in terms of scientific collaboration.

# 2 Financial and Practical Issues

In addition to the annexed *"Financial Overview"* form we make a brief summary on the financial situation.

The 2 Swiss partners are handling the budget assigned to them (4800 CHF over 2 years) on their own, and have spent the money mostly on trips between Bern and Lugano, and some trips to Romania.

No major problems were encountered during the first year of the project. Concerning the transfer of funds we proceeded as follows: the entire scientific equipment was bought in Switzerland and also the entire budget for "Accommodation and Meals" and "Transport Costs" was managed by the University Lugano, instead of transferring it to Romania. The rest of the money for the Romanian partner, *i.e.*, the money for "Individual Grants", "Consumables" and "Overheads" was transferred to the e-Austria Institute in Timişoara and was managed from there. The Romanian partner has provided us with all the receipts both for the "Individual Grants" and for their expenses on consumables.

# References

[BGM06] Mihai Balint, Tudor Gîrba, and Radu Marinescu. How developers copy. In *Proceedings of International Conference on Program Comprehension (ICPC 2006)*, pages 56–65, Los Alamitos CA, 2006. IEEE Computer Society Press.

[1] see argouml.tigris.org

[DG06] Stéphane Ducasse and Tudor Gîrba. Using Smalltalk as a reflective executable meta-language. In *International Conference on Model Driven Engineering Languages and Systems (Models/UML 2006)*, pages 604–618, Berlin, Germany, 2006. Springer-Verlag.

[GD06] Tudor Gîrba and Stéphane Ducasse. Modeling history to analyze software evolution. *Journal of Software Maintenance: Research and Practice (JSME)*, 18:207–236, 2006.

[GHJV95] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley, Reading, Mass., 1995.

[LL06] Mircea Lungu and Michele Lanza. Softwarenaut: Exploring hierarchical system decompositions. In *Proceedings of CSMR 2006 (10th European Conference on Software Maintenance and Reengineering)*, pages 351–354, Los Alamitos CA, 2006. IEEE Computer Society Press.

[LLG06] Mircea Lungu, Michele Lanza, and Tudor Gîrba. Package patterns for visual architecture recovery. In *Proceedings of CSMR 2006 (10th European Conference on Software Maintenance and Reengineering)*, pages 183–192, Los Alamitos CA, 2006. IEEE Computer Society Press.

[LM06] Michele Lanza and Radu Marinescu. *Object-Oriented Metrics in Practice*. Springer-Verlag, 2006.

[Mar06] Cristina Marinescu. Identification of design roles for the assessment of design quality in enterprise applications. In *Proceedings of International Conference on Program Comprehension (ICPC 2006)*, pages 169–180, Los Alamitos CA, 2006. IEEE Computer Society Press.

[MGL06] Michael Meyer, Tudor Gîrba, and Mircea Lungu. Mondrian: An agile visualization framework. In *ACM Symposium on Software Visualization (SoftVis 2006)*, pages 135–144, New York, NY, USA, 2006. ACM Press.

[Mih06] Petru Florin Mihancea. Towards a client driven characterization of class hierarchies. In *Proceedings of International Conference on Program Comprehension (ICPC 2006)*, Los Alamitos CA, 2006. IEEE Computer Society Press.

[NDG05] Oscar Nierstrasz, Stéphane Ducasse, and Tudor Gîrba. The story of Moose: an agile reengineering environment. In *Proceedings of the European Software Engineering Conference (ESEC/FSE 2005)*, pages 1–10, New York NY, 2005. ACM Press. Invited paper.

[TM05] Adrian Trifu and Radu Marinescu. Diagnosing design problems in object oriented systems. In *Proceedings of 12th Working Conference on Reverse Engineering (WCRE 2005), 7-11 November 2005, Pittsburgh, PA, USA*, pages 155–164, Los Alamitos CA, 2005. IEEE Computer Society.