A. Présentation des résultats

The results obtained correspond approximately to those expected from the original proposal. Some results are the logical continuation of the previous project “A framework approach to composing heterogeneous applications” (NFS 20-53711.98. Some other results are more closely related to the companion European IST project PECOS (BBW 00.0170/IST-1999-20398).

Component Meta Model

In the context of the Pecos IST project, we are defining a component model for embedded systems [DW01]. In the same time, we are defining a component model that we plan to use to support the extraction of components from legacy system [AD01a]. In the context of our research on reengineering legacy systems, we previously defined FAMIX, a meta-model for representing code entities. We are currently re-designing FAMIX as an instance of a meta-meta-model (see below). From these works and the work we made on Piccola our composition language, we expect to define a component meta model.

Composition Language

Piccola is a small language designed to compose software components. It builds on the principles of *forms*, *agents*, and *channels*. Forms are extensible records, and agents communicate by exchanging forms over channels. We have stabilized the language, defined a core library of Piccola expressions, and specified inter-language bridging from and to Piccola. The library is supported by both current implementation (i.e., Java and Squeak) [Scha01]. We have also developed a partial evaluation algorithm for Piccola [SA01].

We are working on a more efficient implementation to develop bigger applications, on a type system for the Piccola calculus, and on a distributed version of Piccola.

Component Migration

We started to work on the analysis and support for the migration of object-oriented applications into component ones. We experimented with cluster analysis to identify common abstractions in object-oriented code. We then made some manual experiments to evaluate how object-oriented concepts like inheritance could be mapped into component ones which actually do not support inheritance. We are now re-designing our approach to deal with such a problem [AD01].

Reengineering and Evolution

We continued our work on reengineering legacy applications. We proposed new way of visualize class, by defining a class blueprint [LD01a], [LD01b]. Class blueprints provide a lower level of abstraction than our previous work but introduce more semantics.

To increase the expressive power of our tools and environments, we are currently re-designing FAMIX, the meta-model that we designed to represent source code entities, as an instance of a meta-meta-model. Using this meta-meta-model we expect to have more generic tools and be able to represent components.We started to see how the analysis of multiple versions of an applications supports the understanding and the future evolution of such applications [Lanz01].

We applied QSOUL to support the synchronization between code and documentation [Wuyt01]. QSoul is a unification framework symbiotically and reflectively integrated into an object-oriented language [Wuyt01a]. We analysed the run-time information of application to support the extraction of roles and collaborations [RD01].

B. Publications résultant des recherches effectuées

The publications listed below are for the period from the beginning of the project until the 30 september 2000. The joined articles are the publications that appeared in journals, conference proceedings or books (noted in bold below).

Publications

The following papers have appeared in journals, conference proceedings or books, as indicated.

[LS01a] Michele Lanza and Stéphane Ducasse, “A Categorization of Classes based on the Visualization of their Internal Structure: the Class Blueprint,” Proceedings of OOPSLA 2001, 2001, pp. 300-311.

[SLN01a] Jean-Guy Schneider, Markus Lumpe and Oscar Nierstrasz, “Agent Coordination via Scripting Languages,” in *Coordination of Internet Agents*, Andrea Omicini, Franco Zambonelli, Matthias Klusch and Robert Tolksdorf (Eds.), pp. 153-175, Springer, 2001.

[NA00a] Oscar Nierstrasz and Franz Achermann, “Supporting Compositional Styles for Software Evolution,” Proceedings International Symposium on Principles of Software Evolution (ISPSE 2000), IEEE, Kanazawa, Japan, Nov 1-2 2000, pp. 11-19.

[TDDN00a] Sander Tichelaar, Stéphane Ducasse, Serge Demeyer and Oscar Nierstrasz, “A Meta-model for Language-Independent Refactoring,” Proceedings ISPSE 2000, IEEE, 2000.

The following papers have appeared in workshops proceedings as indicated.

[DLT01] Stéphane Ducasse, Michele Lanza and Sander Tichelaar, “The Moose Reengineering Environment,” Smalltalk Chronicles, August 2001.

[LDS01] Michele Lanza, Stéphane Ducasse and Lukas Steiger, “Understanding Software Evolution using a Flexible Query Engine,” Proceedings of the Workshop on Formal Foundations of Software Evolution, 2001.

[Lanz01] Michele Lanza, “The Evolution Matrix: Recovering Software Evolution using Software Visualization Techniques,” Proceedings of IWPSE 2001 (International Workshop on Principles of Software Evolution), 2001.

[LS01b] Michele Lanza and Stéphane Ducasse, “The Class Blueprint: A Visualization of the Internal Structure of Classes,” Workshop Proceedings of OOPSLA 2001, 2001.

[SA01] Nathanael Schärli and Franz Achermann, “Partial evaluation of inter-language wrappers,” Workshop on Composition Languages, WCL'01, September 2001.

[Wuyt01a] Roel Wuyts and Stéphane Ducasse, “Symbiotic Reflection between an Object-Oriented and a Logic Programming Language”, in ECOOP 2001 International workshop on MultiParadigm Programming with Object-Oriented Languages, 2001

[Wuyt01c] Roel Wuyts and Stéphane Ducasse, Composition Languages for Black-Box Components, First OOPSLA Workshop on Language Mechanisms for Programming Software Components, 2001.

Other publications

[Duca01] Stéphane Ducasse, “Reengineering Object-Oriented Application”, habilitation à diriger des recherche de l’Université de Paris 6, 2001.

[Golo01] Georges Golomingi Koni-N'sapu, “A Scenario Based Approach for Refactoring Duplicated Code in Object Oriented Systems,” Masters thesis, University of Berne, June 2001.

[Scha01] Nathanael Schärli, “Supporting Pure Composition by Inter-language Bridging on the Meta-level,” Masters thesis, University of Bern, September 2001.

[Stei01] Lukas Steiger, “Recovering the Evolution of Object Oriented Software Systems Using a Flexible Query Engine,” Masters thesis, University of Bern, June 2001.

[Wuyt01] Roel Wuyts, “A Logic Meta-Programming Approach to Support the Co-Evolution of Object-Oriented Design and Implementation,” Ph.D. thesis, Vrije Universiteit Brussel, 2001.

Working Papers

The following are working documents that have not yet been submitted for publication.

[Arev01] Gabriela Arevalo, “From white box to Black box“, 2001.

[DW01] Stéphane Ducasse and Roel Wuyts, “A component model for field device“, 2001.

[RD01] Tamar Richner and Stéphane Ducasse, “Extracting Roles and Collaborations“, 2001.

[WD01a] Roel Wuyts and Stéphane Ducasse, “Software Classifications: a Uniform Way to Support Flexible IDE’s”, 2001.

[WD01b] Roel Wuyts and Stéphane Ducasse, “Symbiotic Reflection”, 2001.

[Duca01b] Benny Sadeh and Stéphane Ducasse, “Integrating Dynamic Interface in Smalltalk”, 2001

C. Publications à l’impression

Rapport Scientifique Intermédiare

Projet No: FNRS

Titre: Meta-models and Tools for Evolution Towards Component Systems

Résumé :

All successful software systems evolve to meet changing requirements. Without continual reengineering, however, such systems necessarily suffer from *architectural drift*, as their original design no longer matches new business goals and requirements. As a consequence, they become increasingly complex and fragile, leading to ever higher maintenance costs.

We propose a *component-based approach to software evolution*, in which stable software artifacts are identified over time, and are migrated towards components and component architectures. The key to the approach is a *component meta model* for modelling and analysing evolving software systems. This meta model facilitates *component migration* tools and techniques. As a successful software system matures, instead of becoming more complex and fragile, its architecture gradually migrates towards a configuration of software components, which can be more easily reconfigured and adapted than a typical legacy system.

We propose to develop (i) a component meta model for modelling software systems that extends existing standards (such as UML) with concepts required to support evolution, focusing on such issues as non-functional requirements and software dependencies. Based on this meta model, we will develop (ii) component migration tools and methods that will help to identify candidate components, identify and resolve architectural and design drift, and support transformation to component-based software structures. We will focus on software metrics and visualization to support analysis, and language-independent refactorings to support transformation. Component migration methods will be documented as reverse and reengineering patterns. Finally, we propose to develop (iii) a compositional infrastructure to support architectural specification, and run-time configuration and evolution, using the agent-based framework of the Piccola composition language.