

ECOOP 2002 Demonstration Proposal

SOUL: Supporting Object-Oriented Software Development with Logic Meta Programming

Primary Contact Person

Tom Tourwe
Tom.Tourwe@vub.ac.be
Tel: +32 2 629 3308 Fax: +32 2 629 3525

Authors and affiliation

Tom Tourwe (Tom.Tourwe@vub.ac.be) and **Johan Brichau** (Johan.Brichau@vub.ac.be)
Programming Technology Lab
Vrije Universiteit Brussel
Pleinlaan 2, B-1050 Brussel
Belgium

Kim Mens (Kim.Mens@info.ucl.ac.be)
Département d'ingénierie Informatique
Université catholique de Louvain
Place Sainte-Barbe 2, 1348 Louvain-la-Neuve
Belgium

Short demo description

At the Programming Technology Lab of the Vrije Universiteit Brussel, research is being conducted on how the technique of logic meta-programming can be used to build state-of-the-art software development support tools. By capturing and expressing the interaction between the higher levels of software development (design, analysis, etc...) and the actual implementation level, we can co-evolve the source code and the higher-level software artifacts. The goal is to keep design, documentation and source code synchronised when the software evolves.

The interaction between source code and higher-level artifacts is expressed using a logic meta-programming language. This approach allows us to express these artifacts in a declarative way as meta-information of the program itself. This information can be actively used by the programming environment to guide the software development process in a variety of ways: to generate design from implementation and vice-versa; to provide support for refactoring, software reuse and evolution, aspect-oriented programming, generative programming, etc...

During this presentation, we will demonstrate SOUL (Smalltalk Open Unification Language), an open, reflective logic programming environment written in Smalltalk VisualWorks 5i4 and ported to various other Smalltalk dialects (such as Squeak). In the first part of the presentation we show how the environment and associated tools can be used in practice to:

- Reason about object-oriented programs (including naming and programming conventions [MMW 2001], design pattern extraction [W 1998], type inferencing, UML model extraction, architectural views [MW 1999], ...)
- Declaratively generate code (including source code templates, aspect-oriented programming, design pattern code generation [MT 2001], ...)
- Support software evolution (including refactoring, software merging [MT 2001], ...)

During the second part of the presentation, we take a closer look at the technical details. We show how high-level software artifacts can be expressed in terms of logic facts and rules. For this, we require that the logic meta programming language lives in symbiosis with the object-oriented base language, allowing base level programs to be manipulated as terms, facts or rules in the meta level. In the case of Smalltalk, we use the reflection facilities to implement a logic meta layer on top of the Smalltalk meta-object protocol.

References

- [W 1998] Roel Wuyts. Declarative Reasoning about the Structure of Object-Oriented Systems, Proc. TOOLS USA '98, pp. 112-124, 1998.
- [MW 1999] Kim Mens, Roel Wuyts. Declaratively Codifying Software Architectures using Virtual Software Classifications, Proc. TOOLS Europe '99, LNCS, pp. 33-45, Springer-Verlag, 1999.
- [DDMW 2000] Theo D'Hondt, Kris De Volder, Kim Mens, Roel Wuyts. Co-Evolution of Object-Oriented Software Design and Implementation, Proc. SACT Symposium, Kluwer Academic Publishers.
- [MT 2001] Tom Mens, Tom Tourwe. A Declarative Evolution Framework for Object-Oriented Design Patterns, Proc. Int. Conf. Software Maintenance, Barcelona, 2001.
- [MMW 2001] Kim Mens, Isabel Michiels, Roel Wuyts. Supporting Software Development Through Declaratively Codified Programming Patterns, Proc. Int. Conf. Software Engineering and Knowledge Engineering, Buenos Aires, June 2001.

ECOOP 2002 Demonstration Proposal

SOUL: Supporting Object-Oriented Software Development with Logic Meta Programming

Hardware and software requirements

1 Macintosh-compatible data projector

1 flipchart + 3 markers

1 overhead projector