Declarative Meta Programming to Support Software Development: Workshop Report

Tom Mens*

Programming Technology Lab
Vrije Universiteit Brussel, Belgium
tom.mens@vub.ac.be

Kris De Volder

Department of Computer Science
University of British Columbia, Canada
kdvolder@cs.ubc.ca

Roel Wuyts

Software Composition Group
University of Bern, Switzerland
roel.wuyts@iam.unibe.ch

Kim Mens

Département d'Ingénierie Informatique
Université catholique de Louvain, Belgium
kim.mens@info.ucl.ac.be

Abstract

This paper reports on the results of the workshop on *Declarative Meta Programming to Support Software Development* in Edinburgh on September 23, 2002. It enumerates the presentations made, classifies the contributions and lists the main results of the discussions held at the workshop. As such it provides the context for future workshops around this topic.

Keywords: meta programming, declarative languages, software development

Introduction

The workshop on *Declarative Meta Programming to Support Software Development* (DMP 02) was co-located with the *17th International Conference on Automated Software Engineering* (ASE 2002), and took place at the Heriot-Watt University in Edinburgh, United Kingdom, on September 23, 2002. There were 13 participants, most of which contributed with a position paper that was reviewed and revised before the workshop. The participants originated from Belgium, Canada, France, Switzerland, Israel, United Kingdom, and the USA.

The workshop focused on declarative meta programming (DMP) techniques and tools to support software development. Such techniques and tools are *meta programming* because they reason about or manipulate program code at a meta level to automate some aspect of the software development process. The fact that they are *declarative* means that they focus on *what* is being done rather than *how* it is done.

The workshop had the following explicit goals:

- Get an overview of existing DMP approaches.

- Delineate for which software development activities DMP could be used.

- Compare existing approaches (tools, techniques and formalisms) and identify commonalities and differences.

- Discuss advantages and shortcomings of DMP for supporting software development.

---

*Tom Mens is a postdoctoral fellow of the Fund for Scientific Research - Flanders (Belgium).

Workshop presentations

The morning session was devoted to four long presentations of 20 minutes and four short presentations of 10 minutes, each followed by 5 minutes of discussion. The long presentations were chosen by the organisers because they offered different or novel perspectives on the workshop topic, and because they had a higher potential for generating issues that would stimulate the discussions.

The papers and their authors were as follows, with the names of the actual presenters during the workshop underlined. The papers were collected in a technical report [WMDM02].

Long presentations:

LP1 Toacy Oliveira, Paulo Alencar, Donald Cowan (University of Waterloo, Canada). Towards a declarative approach to framework instantiation.

LP2 Yann-Gaël Guéhéneuc (École des Mines de Nantes, France). Meta-modelling, logic programming, and explanation-based constraint programming for pattern description and detection.

LP3 Tom Tourwe, Johan Brichau, Tom Mens (Vrije Universiteit Brussel, Belgium). Using declarative metaprogramming to detect possible refactorings.

LP4 Gopal Gupta (University of Texas, Dallas, USA). A language-centric approach to software engineering: domain specific languages meet software components.

Short presentations:

SP1 Tom Tourwé, Tom Mens (Vrije Universiteit Brussel, Belgium). A declarative meta-programming approach to framework documentation.

SP2 H. Akehurst, Behzad Bordbar, P.J.Rodgers, N.T.G. Dalgliesh (University of Kent, United Kingdom). Automatic normalisation via metamodeling.

SP3 Robert Filman, Klaus Havelund (NASA Ames Research Center, California, USA). Realising aspects by transforming for events.

SP4 Greg Michaelson (Heriot-Watt University, United Kingdom). SML prototypes from Z specifications.

SP5 <u>Cordell Green</u> (Kestrel Institute, USA). SpecWare: Automatic formal specifications into hardware.

According to the workshop topic, the papers could be classified according to two dimensions: the kind of DMP technique they use (see Table 1) and the kind of support for software development they provide (see Table 2).

Presentation	DMP approach used
LP1	annotated UML, XML, XSLT
LP2	meta modelling, logic programming, explanation-based constraint programming
LP3	logic meta programming
LP4	constraint logic programming, denotational semantics
SP1	logic meta programming
SP2	OCL, graph rewriting
SP3	declarative language
SP4	translation scheme
SP5	theorem provers

Table 1: Declarative Meta Programming approach used

In Table 1 we observe that most of the presented declarative meta programming approaches use some variant of logic meta programming (LP2, LP3, LP4, SP1). Other approaches use more trendy languages and standard technologies such as UML, OCL, XMI, XML and XSLT (LP1, SP2).

Presentation	Kind of development support
LP1	framework documentation
LP2	design patterns
LP3	design patterns, refactoring
LP4	domain-specific languages
SP1	framework instantiation and evolution
SP2	database normalisation
SP3	aspect-oriented programming
SP4	program translation
SP5	code generation from formal specifications

Table 2: Kind of software development support

As can be seen from Table 2, the bulk of the presented approaches uses declarative meta programming to provide support for developing *object-oriented* software applications (LP1, LP2, LP3, SP1, SP3). This support includes: documentation, instantiation and evolution of object-oriented application frameworks; description, detection, generation and conformance checking of design pattern instances; object-oriented refactoring; aspect-oriented programming.

Workshop discussions

In order to stimulate discussions, some general important questions were posed to the participants during the workshop:

Q1 What are the main advantages of DMP over other approaches?

The following benefits were mentioned by the participants:

- *Portability* and *platform independence*. For example, if we express domain-specific languages with DMP, they can be automatically translated to any target platform.

- Declarative programs provide an *executable form of documentation*. *Executable*, since they are programs; *documentation*, since the declarative notation is easy to read and understand.

- *Conciseness* and *complexity reduction*. Declarative programs are often significantly smaller and less complex than non-declarative programs. Cordell Green mentioned an experimentally validated factor 2 to 5 reduction of program dependencies.

- *Error reduction*. This is a direct consequence of complexity reduction. Cordell Green cited an experimentally validated error reduction of a factor 2 to 20.

Q2 What are the potential shortcomings of DMP?

- *Performance* and *efficiency* issues were coined as a potential disadvantage of DMP, but most of the participants agreed that this was a non-issue. With the current state-of-the-art in compiler technology, very efficient logic languages can be implemented.

- Declarative meta programming involves a *high degree of sophistication*. It requires a deep understanding of language semantics. This is even more the case with hybrid DMP, for example when a declarative meta language is used on top of an object-oriented base language. In that case, complex issues such as language symbiosis come into play. As a result, DMP is not suited for the average programmer, and it will never find widespread use. This resulted in the third question to be discussed:

Q3 How can DMP achieve more widespread acceptance as a mechanism for supporting software development?

- *Lack of standard technologies* was suggested as a reason why declarative languages have not found wide adoption for software development support. This can be resolved relatively easy by putting an XML-syntax on top of the declarative language, at the expense of losing the more concise and readable notation.

- A second aspect that strongly affect acceptance of DMP is the *quality and usability of the supported tools*. Two powerful and promising tools for DMP were presented at the end of the day, and are discussed later in this paper.

Q4 For which kinds of support for software development is DMP well-suited/unsuited?

This final question was only discussed very briefly due to time constraints. *Parse tree manipulation* was proposed as

something for which DMP is particularly well suited. Indeed, many of the presented approaches used or proposed some kind of parse tree manipulation for generating, transforming or reasoning about code.

Tool demonstrations

Upon explicit request by the workshop participants, a special tool demonstration session was scheduled at the end of the day, where two sophisticated DMP tools for reasoning about object-oriented programs (one for Smalltalk and one for Java) were demonstrated.

The first tool, *Soul* [MMW02] was presented by Johan Brichau. It is a Prolog-like logic meta programming language built on top of, and tightly integrated with, a Smalltalk object-oriented software development environment. It enables support for design patterns, coding conventions, programming styles, refactoring, and software metrics.

The second demonstration was made by Yann-Gaël Guéhéneuc and showed the *Patternsbox* tool (that allows to select and instantiate patterns) and the *PtiDej* tool (that does program architecture visualization and patterns detection). These tools allow to specify (patterns), and then use these specifications to generate code or check the specification against Java source code. One of the very nice features is that it employs a constraint system that gives feedback on how well the patterns match the code. Hence the pattern serves more as a fuzzy definition that can yield partial matches, and it explains these results.

Acknowledgements

References

[ESF02] Fund for Scientific Research - Flanders (Belgium). *Scientific Research Network on Foundations of Software Evolution.*
http://prog.vub.ac.be/FFSE [1 Oct 2002]

[GDJ02] Yann-Gaël Guéhéneuc, Rémi Douence and Narendra Jussien. No Java without Caffeine: A tool for dynamic analysis of Java programs. In *Proc. Int'l Conf. Automated Software Engineering*, pages 117-126, Edinburgh, United Kingdom, September 2002. IEEE Computer Society Press.

[MMW02] Kim Mens, Isabel Michiels, Roel Wuyts. Supporting Software Development through Declaratively Codified Programming Patterns. *Journal on Expert Systems with Applications*, December 2002. Elsevier Publications.

[WMDM02] Roel Wuyts, Tom Mens, Kris De Volder and Kim Mens. Proc. of the *Workshop on Declarative Meta-Programming to Support Software Development*. Technical Report VUB-PROG-TR-??-2002, Programming Technology Lab, Vrije Universiteit Brussel, 2002. http://www.cs.ubc.ca/ kdvolder/Workshops/ASE2002/DMP/ [1 Oct 2002]