Gabor Karsai
Eelco Visser (Eds.)

LNCS 3286

Generative Programming and Component Engineering

Third International Conference, GPCE 2004
Vancouver, Canada, October 2004
Proceedings

Springer

Gabor Karsai Eelco Visser (Eds.)

Generative Programming and Component Engineering

Third International Conference, GPCE 2004
Vancouver, Canada, October 24-28, 2004
Proceedings

Springer

Volume Editors

Gabor Karsai
Vanderbilt University
Institute for Software Integrated Systems (ISIS)
Nashville, TN 37235, USA
E-mail: gabor.karsai@vanderbilt.edu

Eelco Visser
Universiteit Utrecht
Institute of Information and Computing Sciences
P.O. Box 80089, 3508 TB Utrecht, The Netherlands
E-mail: visser@acm.org

Library of Congress Control Number: 2004113646

CR Subject Classification (1998): D.2, D.1, D.3, K.6

ISSN 0302-9743
ISBN 3-540-23580-9 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Olgun Computergrafik
Printed on acid-free paper SPIN: 11338000 06/3142 5 4 3 2 1 0

Lecture Notes in Computer Science 3286

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

Preface

Generative and component approaches have the potential to revolutionize software development in a similar way as automation and components revolutionized manufacturing. Generative Programming (developing programs that synthesize other programs), Component Engineering (raising the level of modularization and analysis in application design), and Domain-Specific Languages (elevating program specifications to compact domain-specific notations that are easier to write and maintain) are key technologies for automating program development.

GPCE arose as a joint conference, merging the prior conference on Generative and Component-Based Software Engineering (GCSE) and the Workshop on Semantics, Applications, and Implementation of Program Generation (SAIG). The goal of GPCE is to provide a meeting place for researchers and practitioners interested in cutting edge approaches to software development. We aim to foster further cross-fertilization between the software engineering research community on the one hand, and the programming languages community on the other, in addition to supporting the original research goals of both the GCSE and the SAIG communities.

This volume contains the proceedings of the *Third International Conference on Generative Programming and Component Engineering*, held in Vancouver, Canada from October 24 to 28, 2004, where it was co-located with OOPSLA 2004 and ISSM 2004.

Responding to the call for papers 89 abstracts were submitted, 75 of which materialized as full paper submissions. The papers were reviewed by program committee members and their co-reviewers who together produced a total of 250 reviews, between 3 and 5 per paper. Reviews were often thorough and sometimes actually included the views of multiple co-reviewers. Consensus about the papers to be accepted was reached during the online program committee meeting held in the second week of May 2004. The meeting consisted of a discussion by email of each of the papers by the entire PC so that members could get an overall impression of the quality of the submitted papers, beyond the ones they reviewed themselves. The committee selected 25 of the 75 papers for presentation at the conference and publication in the proceedings. Of the accepted papers, two are co-athored by PC members (from a total of six PC submissions). We tried hard to ensure fairness and held PC submissions to a high standard. Paper submission and reviewing were supported by the open source version of the CyberChair conference system installed at the webserver of the Institute of Information and Computing Sciences of Utrecht University, The Netherlands.

In addition to the technical paper presentations the conferences featured two invited speakers, a panel, four tutorials, five workshops, and six demonstrations.

Invited Speakers. The keynote talk by Jack Greenfield examined the *software factory* approach to rapidly develop domain-specific languages and tools to auto-

mate the production of applications in specific domains, combining innovations in adaptive assembly, software product lines, and model driven development.

The invited talk by Peter Mosses gave an overview of the state of the art in *modular language description*, i.e. the specification of the semantics of programming language features in separate modules such that new languages can be defined by module composition.

Panel. A panel chaired by Gabor Karsai and further consisting of Don Batory, Krzysztof Czarnecki, Jeff Gray, Douglas Schmidt, and Walid Taha examined the current state of the field of generative programming, addressing issues such as its relevance for information technology practice, incorporating generative approaches in education, evaluation and comparison of generative technologies, and research challenges.

Tutorials. The four GPCE tutorials gave introductions to important areas of the generative programming field:

- *Adaptive object-model architecture: Dynamically adapting to changing requirements* by Joe Yoder
- *Multi-stage programming in MetaOCaml* by Walid Taha and Cristiano Calcagno
- *Generative software development* by Krzysztof Czarnecki and Jack Greenfield
- *Program transformation systems: Theory and practice for software generation, maintenance, and reengineering* by Ira Baxter and Hongjun Zheng

Workshops. Prior to GPCE 2004 six workshops were held, providing an opportunity for attendees to exchange views in subareas of generative programming.

With the introduction of software product line approaches into the practice, variants and variability add a new dimension of complexity to the software development process. The combinatorial explosion of possible variants in systems with a high degree of variability requires improved and changed concepts for specifying, modeling, and implementing these systems to assure quality and functionality. In the **OOPSLA/GPCE Workshop on Managing Variabilities Consistently in Design and Code** participants discussed and identified efficient ways for dealing with highly variable software systems on design and code level by evaluating existing approaches and new ideas from the research community and industrial practice.

The **Software Transformation Systems Workshop** was designed to investigate the use of software transformation tools to support generative programming by looking at various generative techniques and suggesting how these may be supported by various general purpose transformation tools, leading to a more general understanding of common principles for supporting generative methods.

MetaOCaml is a multi-stage extension of the widely used functional programming language OCaml. It provides a generic core for expressing macros, staging, and partial evaluation. The **First MetaOCaml Workshop** provided a forum for discussing experience with using MetaOCaml as well as possible future developments for the language.

The **6th GPCE Young Researchers Workshop** provided a platform for young international researchers to present their work and receive feedback from experienced panelists.

The **OOPSLA/GPCE Workshop on Best Practices for Model-Driven Software Development** brought together practitioners, researchers, academics, and students to discuss the best practices for the development of model-driven software, and to discuss the state of the art of tool support for MDSD, including emerging Open Source tool products for model-driven development of software systems.

Demonstrations. The following demonstrations were held in parallel to the technical paper program:

- *Implementation of DSLs using staged interpreters in MetaOCaml* by Kedar Swadi from Rice University
- *MetaEdit+: Domain-specific modeling for full code generation demonstrated* by Juha-Pekka Tolvanen from MetaCase
- *Towards domain-driven development: the SmartTools software factory* by Didier Parigot from INRIA Sophia-Antipolis
- *Xirc: Cross-artifact information retrieval* by Michael Eichberg, and Thorsten Schaefer from Darmstadt University of Technology
- *C-SAW and GenAWeave: A two-level aspect weaving toolsuite* by Jeff Gray, Jing Zhang, and Suman Roychoudhury, from the University of Alabama at Birmingham and Ira Baxter from Semantic Designs
- *The concern manipulation environment* by Peri Tarr, Matthew Chapman, William Chung, and Andy Clement, from the IBM Thomas J. Watson Research Center and IBM Hursley Park.
- *Program transformations for re-engineering C++ components* by Ira Baxter, Larry Akers, Semantic Designs, and Michael Mehlich from Semantic Designs.

The program of this year's conference is proof that the GPCE community is a vibrant, lively group that produces significant new contributions.

August 2004

Gabor Karsai
Eelco Visser

Organization

GPCE 2004 was organized by the Assocation for Computing Machinery (ACM), the OGI School of Science & Engineering at OHSU (USA), Utrecht University (The Netherlands), Vanderbilt University (USA), Intel (USA), University of Alabama at Birmingham (USA), and the University of Waterloo (Canada). The event was sponsored by ACM SIGPLAN, ACM SIGSOFT, and Microsoft and co-located with OOPSLA 2004 and ISSM 2004 in Vancouver, Canada.

General Chair

Tim Sheard, OGI School of Science & Engineering at OHSU, Portland, Oregon, USA

Program Chairs

Gabor Karsai, Vanderbilt University, USA
Eelco Visser, Utrecht University, The Netherlands

Program Committee

Uwe Aßmann (Linköpings Universitet, Sweden)
Don Batory (University of Texas, USA)
Jan Bosch (Universiteit Groningen, The Netherlands)
Jean Bezivin (Université de Nantes, France)
Jim Cordy (Queen's University, Canada)
Krzysztof Czarnecki (University of Waterloo, Canada)
Mathew Flatt (University of Utah, USA)
Robert Glück (Københavns Universitet, Denmark)
George Heineman (Worcester Polytechnic Institute, USA)
Michael Leuschel (University of Southampton, UK)
Karl Lieberherr (Northeastern University, USA)
Simon Peyton Jones (Microsoft Research, UK)
Douglas R. Smith (Kestrel Institute, USA)
Gabriele Taentzer (Technische Universität Berlin, Germany)
Todd Veldhuizen (Indiana University, USA)
Kris de Volder (University of British Columbia, Canada)
Dave Wile (Teknowledge Corporation, USA)
Alexander Wolf (University of Colorado at Boulder, USA)

Workshop Chair

Zino Benaissa, Intel, USA

Tutorial Chair

Jeff Gray, University of Alabama at Birmingham, USA

Demonstrations Committee

Simon Helsen (chair), University of Waterloo, Canada
William Cook, University of Texas at Austin, USA
Frédéric Jouault, Université de Nantes, France

Co-reviewers

Nils Andersen
Anya H. Bagge
Ivor Bosloper
Martin Bravenboer
Niels H. Christensen
Alessandro Coglio
S. Deelstra
Juergen Dingel
Eelco Dolstra
Alexander Egyed
Dan Elphick
Lindsay Errington
Natalya Filatkina
Murdoch Gabbay
Hugh Glaser
Andy Gravell
Jurriaan Hage
Jan Heering
Andre van der Hoek
Kathrin Hoffmann
Yvonne Howard
Anton Jansen
Michel Jaring
Dean Jin
Merijn de Jonge
Frederic Jouault
Kazuhiko Kakehi
Karl Trygve Kalleberg
Markus Klein
Andrei Klimov
Jia Liu
Roberto Lopez-Herrejon
David Lorenz
Andrew Malton
Katharina Mehner
Anne-Francoise Le Meur
Torben Mogensen
Dirk Muthig
Karina Olmos
Scott Owens
Jeffrey Palm
Emir Pasalic
Andrew Pitts
Stephane Lo Presti
Armin Rigo
Matthew Rutherford
Kevin A. Schneider
Jörg Schneider
Ulrik Pagh Schultz
Johanneke Siljee
M. Sinnema
Therapon Skotiniotis
Mike Sperber
Walid Taha
Edward Turner
Mauricio Varea
Andrzej Wasowski
Andreas Winter
Pengcheng Wu

Steering Committee

Don Batory (University of Texas)
Eugenio Moggi (Università di Genova)
Greg Morrisett (Cornell)
Janos Sztipanovits (Vanderbilt University School of Engineering)
Krzysztof Czarnecki (University of Waterloo)
Walid Taha (Rice University)
Bogdan Franczyk (Universität Leipzig)
Ulrich Eisenecker (Fachhochschule Kaiserslautern)

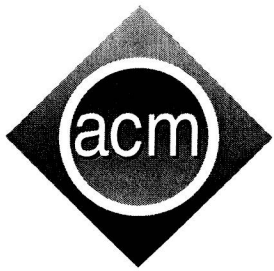
Previous Events

GPCE emerged as the unification of the SAIG workshop series and the GCSE conference series.

GPCE 2003, Erfurt, Germany
GPCE 2002, Pittsburgh, Pensylvania, USA
GCSE 2001, Erfurt, Germany
GCSE 2000, Erfurt, Germany
GCSE 1999, Erfurt, Germany
SAIG 2001, Firenze, Italy
SAIG 2000, Montréal, Canada

See also the permanent website of the conference series `http://www.gpce.org`

Sponsors



ACM SIGPLAN and ACM SIGSOFT

Microsoft®

Lecture Notes in Computer Science

For information about Vols. 1–3181

please contact your bookseller or Springer

Vol. 3305: P.M.A. Sloot, B. Chopard, A.G. Hoekstra (Eds.), Cellular Automata. XV, 883 pages. 2004.

Vol. 3293: C.-H. Chi, M. van Steen, C. Wills (Eds.), Web Content Caching and Distribution. IX, 283 pages. 2004.

Vol. 3286: G. Karsai, E. Visser (Eds.), Generative Programming and Component Engineering. XIII, 491 pages. 2004.

Vol. 3274: R. Guerraoui (Ed.), Distributed Computing. XIII, 465 pages. 2004.

Vol. 3273: T. Baar, A. Strohmeier, A. Moreira, S.J. Mellor (Eds.), <<UML>> 2004 - The Unified Modelling Language. XIII, 454 pages. 2004.

Vol. 3271: J. Vicente, D. Hutchison (Eds.), Management of Multimedia Networks and Services. XIII, 335 pages. 2004.

Vol. 3270: M. Jeckle, R. Kowalczyk, P. Braun (Eds.), Grid Services Engineering and Management. X, 165 pages. 2004.

Vol. 3269: J. López, S. Qing, E. Okamoto (Eds.), Information and Communications Security. XI, 564 pages. 2004.

Vol. 3266: J. Solé-Pareta, M. Smirnov, P.V. Mieghem, J. Domingo-Pascual, E. Monteiro, P. Reichl, B. Stiller, R.J. Gibbens (Eds.), Quality of Service in the Emerging Networking Panorama. XVI, 390 pages. 2004.

Vol. 3265: R.E. Frederking, K.B. Taylor (Eds.), Machine Translation: From Real Users to Research. XI, 392 pages. 2004. (Subseries LNAI).

Vol. 3264: G. Paliouras, Y. Sakakibara (Eds.), Grammatical Inference: Algorithms and Applications. XI, 291 pages. 2004. (Subseries LNAI).

Vol. 3263: M. Weske, P. Liggesmeyer (Eds.), Object-Oriented and Internet-Based Technologies. XII, 239 pages. 2004.

Vol. 3262: M.M. Freire, P. Chemouil, P. Lorenz, A. Gravey (Eds.), Universal Multiservice Networks. XIII, 556 pages. 2004.

Vol. 3261: T. Yakhno (Ed.), Advances in Information Systems. XIV, 617 pages. 2004.

Vol. 3260: I.G.M.M. Niemegeers, S.H. de Groot (Eds.), Personal Wireless Communications. XIV, 478 pages. 2004.

Vol. 3258: M. Wallace (Ed.), Principles and Practice of Constraint Programming – CP 2004. XVII, 822 pages. 2004.

Vol. 3257: E. Motta, N.R. Shadbolt, A. Stutt, N. Gibbins (Eds.), Engineering Knowledge in the Age of the Semantic Web. XVII, 517 pages. 2004. (Subseries LNAI).

Vol. 3256: H. Ehrig, G. Engels, F. Parisi-Presicce, G. Rozenberg (Eds.), Graph Transformations. XII, 451 pages. 2004.

Vol. 3255: A. Benczúr, J. Demetrovics, G. Gottlob (Eds.), Advances in Databases and Information Systems. XI, 423 pages. 2004.

Vol. 3254: E. Macii, V. Paliouras, O. Koufopavlou (Eds.), Integrated Circuit and System Design. XVI, 910 pages. 2004.

Vol. 3253: Y. Lakhnech, S. Yovine (Eds.), Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems. X, 397 pages. 2004.

Vol. 3250: L.-J. (LJ) Zhang, M. Jeckle (Eds.), Web Services. X, 301 pages. 2004.

Vol. 3249: B. Buchberger, J.A. Campbell (Eds.), Artificial Intelligence and Symbolic Computation. X, 285 pages. 2004. (Subseries LNAI).

Vol. 3246: A. Apostolico, M. Melucci (Eds.), String Processing and Information Retrieval. XIV, 332 pages. 2004.

Vol. 3245: E. Suzuki, S. Arikawa (Eds.), Discovery Science. XIV, 430 pages. 2004. (Subseries LNAI).

Vol. 3244: S. Ben-David, J. Case, A. Maruoka (Eds.), Algorithmic Learning Theory. XIV, 505 pages. 2004. (Subseries LNAI).

Vol. 3243: S. Leonardi (Ed.), Algorithms and Models for the Web-Graph. VIII, 189 pages. 2004.

Vol. 3242: X. Yao, E. Burke, J.A. Lozano, J. Smith, J.J. Merelo-Guervós, J.A. Bullinaria, J. Rowe, P. Tiňo, A. Kabán, H.-P. Schwefel (Eds.), Parallel Problem Solving from Nature - PPSN VIII. XX, 1185 pages. 2004.

Vol. 3241: D. Kranzlmüller, P. Kacsuk, J.J. Dongarra (Eds.), Recent Advances in Parallel Virtual Machine and Message Passing Interface. XIII, 452 pages. 2004.

Vol. 3240: I. Jonassen, J. Kim (Eds.), Algorithms in Bioinformatics. IX, 476 pages. 2004. (Subseries LNBI).

Vol. 3239: G. Nicosia, V. Cutello, P.J. Bentley, J. Timmis (Eds.), Artificial Immune Systems. XII, 444 pages. 2004.

Vol. 3238: S. Biundo, T. Frühwirth, G. Palm (Eds.), KI 2004: Advances in Artificial Intelligence. XI, 467 pages. 2004. (Subseries LNAI).

Vol. 3236: M. Núñez, Z. Maamar, F.L. Pelayo, K. Pousttchi, F. Rubio (Eds.), Applying Formal Methods: Testing, Performance, and M/E-Commerce. XI, 381 pages. 2004.

Vol. 3235: D. de Frutos-Escrig, M. Nunez (Eds.), Formal Techniques for Networked and Distributed Systems – FORTE 2004. X, 377 pages. 2004.

Vol. 3232: R. Heery, L. Lyon (Eds.), Research and Advanced Technology for Digital Libraries. XV, 528 pages. 2004.

Vol. 3231: H.-A. Jacobsen (Ed.), Middleware 2004. XV, 514 pages. 2004.

Vol. 3230: J.L. Vicedo, P. Martínez-Barco, R. Muñoz, M.S. Noeda (Eds.), Advances in Natural Language Processing. XII, 488 pages. 2004. (Subseries LNAI).

Vol. 3229: J.J. Alferes, J. Leite (Eds.), Logics in Artificial Intelligence. XIV, 744 pages. 2004. (Subseries LNAI).

Vol. 3226: M. Bouzeghoub, C. Goble, V. Kashyap, S. Spaccapietra (Eds.), Semantics for Grid Databases. XIII, 326 pages. 2004.

Vol. 3225: K. Zhang, Y. Zheng (Eds.), Information Security. XII, 442 pages. 2004.

Vol. 3224: E. Jonsson, A. Valdes, M. Almgren (Eds.), Recent Advances in Intrusion Detection. XII, 315 pages. 2004.

Vol. 3223: K. Slind, A. Bunker, G. Gopalakrishnan (Eds.), Theorem Proving in Higher Order Logics. VIII, 337 pages. 2004.

Vol. 3222: H. Jin, G.R. Gao, Z. Xu, H. Chen (Eds.), Network and Parallel Computing. XX, 694 pages. 2004.

Vol. 3221: S. Albers, T. Radzik (Eds.), Algorithms – ESA 2004. XVIII, 836 pages. 2004.

Vol. 3220: J.C. Lester, R.M. Vicari, F. Paraguaçu (Eds.), Intelligent Tutoring Systems. XXI, 920 pages. 2004.

Vol. 3219: M. Heisel, P. Liggesmeyer, S. Wittmann (Eds.), Computer Safety, Reliability, and Security. XI, 339 pages. 2004.

Vol. 3217: C. Barillot, D.R. Haynor, P. Hellier (Eds.), Medical Image Computing and Computer-Assisted Intervention – MICCAI 2004. XXXVIII, 1114 pages. 2004.

Vol. 3216: C. Barillot, D.R. Haynor, P. Hellier (Eds.), Medical Image Computing and Computer-Assisted Intervention – MICCAI 2004. XXXVIII, 930 pages. 2004.

Vol. 3215: M.G.. Negoita, R.J. Howlett, L.C. Jain (Eds.), Knowledge-Based Intelligent Information and Engineering Systems. LVII, 906 pages. 2004. (Subseries LNAI).

Vol. 3214: M.G.. Negoita, R.J. Howlett, L.C. Jain (Eds.), Knowledge-Based Intelligent Information and Engineering Systems. LVIII, 1302 pages. 2004. (Subseries LNAI).

Vol. 3213: M.G.. Negoita, R.J. Howlett, L.C. Jain (Eds.), Knowledge-Based Intelligent Information and Engineering Systems. LVIII, 1280 pages. 2004. (Subseries LNAI).

Vol. 3212: A. Campilho, M. Kamel (Eds.), Image Analysis and Recognition. XXIX, 862 pages. 2004.

Vol. 3211: A. Campilho, M. Kamel (Eds.), Image Analysis and Recognition. XXIX, 880 pages. 2004.

Vol. 3210: J. Marcinkowski, A. Tarlecki (Eds.), Computer Science Logic. XI, 520 pages. 2004.

Vol. 3209: B. Berendt, A. Hotho, D. Mladenic, M. van Someren, M. Spiliopoulou, G. Stumme (Eds.), Web Mining: From Web to Semantic Web. IX, 201 pages. 2004. (Subseries LNAI).

Vol. 3208: H.J. Ohlbach, S. Schaffert (Eds.), Principles and Practice of Semantic Web Reasoning. VII, 165 pages. 2004.

Vol. 3207: L.T. Yang, M. Guo, G.R. Gao, N.K. Jha (Eds.), Embedded and Ubiquitous Computing. XX, 1116 pages. 2004.

Vol. 3206: P. Sojka, I. Kopecek, K. Pala (Eds.), Text, Speech and Dialogue. XIII, 667 pages. 2004. (Subseries LNAI).

Vol. 3205: N. Davies, E. Mynatt, I. Siio (Eds.), UbiComp 2004: Ubiquitous Computing. XVI, 452 pages. 2004.

Vol. 3204: C.A. Peña Reyes, Coevolutionary Fuzzy Modeling. XIII, 129 pages. 2004.

Vol. 3203: J. Becker, M. Platzner, S. Vernalde (Eds.), Field Programmable Logic and Application. XXX, 1198 pages. 2004.

Vol. 3202: J.-F. Boulicaut, F. Esposito, F. Giannotti, D. Pedreschi (Eds.), Knowledge Discovery in Databases: PKDD 2004. XIX, 560 pages. 2004. (Subseries LNAI).

Vol. 3201: J.-F. Boulicaut, F. Esposito, F. Giannotti, D. Pedreschi (Eds.), Machine Learning: ECML 2004. XVIII, 580 pages. 2004. (Subseries LNAI).

Vol. 3199: H. Schepers (Ed.), Software and Compilers for Embedded Systems. X, 259 pages. 2004.

Vol. 3198: G.-J. de Vreede, L.A. Guerrero, G. Marín Raventós (Eds.), Groupware: Design, Implementation and Use. XI, 378 pages. 2004.

Vol. 3196: C. Stary, C. Stephanidis (Eds.), User-Centered Interaction Paradigms for Universal Access in the Information Society. XII, 488 pages. 2004.

Vol. 3195: C.G. Puntonet, A. Prieto (Eds.), Independent Component Analysis and Blind Signal Separation. XXIII, 1266 pages. 2004.

Vol. 3194: R. Camacho, R. King, A. Srinivasan (Eds.), Inductive Logic Programming. XI, 361 pages. 2004. (Subseries LNAI).

Vol. 3193: P. Samarati, P. Ryan, D. Gollmann, R. Molva (Eds.), Computer Security – ESORICS 2004. X, 457 pages. 2004.

Vol. 3192: C. Bussler, D. Fensel (Eds.), Artificial Intelligence: Methodology, Systems, and Applications. XIII, 522 pages. 2004. (Subseries LNAI).

Vol. 3191: M. Klusch, S. Ossowski, V. Kashyap, R. Unland (Eds.), Cooperative Information Agents VIII. XI, 303 pages. 2004. (Subseries LNAI).

Vol. 3190: Y. Luo (Ed.), Cooperative Design, Visualization, and Engineering. IX, 248 pages. 2004.

Vol. 3189: P.-C. Yew, J. Xue (Eds.), Advances in Computer Systems Architecture. XVII, 598 pages. 2004.

Vol. 3188: F.S. de Boer, M.M. Bonsangue, S. Graf, W.-P. de Roever (Eds.), Formal Methods for Components and Objects. VIII, 373 pages. 2004.

Vol. 3187: G. Lindemann, J. Denzinger, I.J. Timm, R. Unland (Eds.), Multiagent System Technologies. XIII, 341 pages. 2004. (Subseries LNAI).

Vol. 3186: Z. Bellahsène, T. Milo, M. Rys, D. Suciu, R. Unland (Eds.), Database and XML Technologies. X, 235 pages. 2004.

Vol. 3185: M. Bernardo, F. Corradini (Eds.), Formal Methods for the Design of Real-Time Systems. VII, 295 pages. 2004.

Vol. 3184: S. Katsikas, J. Lopez, G. Pernul (Eds.), Trust and Privacy in Digital Business. XI, 299 pages. 2004.

Vol. 3183: R. Traunmüller (Ed.), Electronic Government. XIX, 583 pages. 2004.

Vol. 3182: K. Bauknecht, M. Bichler, B. Pröll (Eds.), E-Commerce and Web Technologies. XI, 370 pages. 2004.

Table of Contents

Aspect Orientation

Staged Programming

Types of Meta-programming

Meta-programming

Model-Driven Approaches

Product Lines

Domain-Specific Languages and Generation

Invited Speakers

Generating AspectJ Programs with Meta-AspectJ

David Zook, Shan Shan Huang, and Yannis Smaragdakis

College of Computing, Georgia Institute of Technology
Atlanta, GA 30332, USA
{dzook,ssh,yannis}@cc.gatech.edu

Abstract. Meta-AspectJ (MAJ) is a language tool for generating AspectJ programs using code templates. MAJ itself is an extension of Java, so users can interleave arbitrary Java code with AspectJ code templates. MAJ is a structured meta-programming tool: a well-typed generator implies a syntactically correct generated program. MAJ promotes a methodology that combines aspect-oriented and generative programming. Potential applications range from implementing domain-specific languages with AspectJ as a back-end to enhancing AspectJ with more powerful general-purpose constructs. In addition to its practical value, MAJ offers valuable insights to meta-programming tool designers. It is a mature meta-programming tool for AspectJ (and, by extension, Java): a lot of emphasis has been placed on context-sensitive parsing and error-reporting. As a result, MAJ minimizes the number of meta-programming (quote/unquote) operators and uses type inference to reduce the need to remember type names for syntactic entities.

1 Introduction

Meta-programming is the act of writing programs that generate other programs. Powerful meta-programming is essential for approaches to automating software development. In this paper we present Meta-AspectJ (MAJ): a meta-programming language tool extending Java with support for generating AspectJ [9] programs. MAJ offers a convenient syntax, while explicitly representing the syntactic structure of the generated program during the generation process. This allows MAJ to guarantee that a well-typed generator will result in a syntactically correct generated program. This is the hallmark property of *structured* meta-programming tools, as opposed to lexical or text-based tools. Structured meta-programming is desirable because it means that a generator can be released with some confidence that it will create reasonable programs regardless of its inputs.

Why should anyone generate AspectJ programs, however? We believe that combining generative techniques with aspect-oriented programming results in significant advantages compared to using either approach alone. MAJ can be used for two general kinds of tasks: to implement generators using AspectJ and to implement general-purpose aspect languages using generation. Specifically,

G. Karsai and E. Visser (Eds.): GPCE 2004, LNCS 3286, pp. 1–18, 2004.

MAJ can be used to implement domain-specific languages (i.e., to implement a generator) by translating domain-specific abstractions into AspectJ code. MAJ can also be used to implement general-purpose extensions of AspectJ (e.g., extensions that would recognize different kinds of joinpoints). Thus, MAJ enables the use of AspectJ as an aspect-oriented "assembly language" [13] to simplify what would otherwise be tedious tasks of recognizing patterns in an existing program and rewriting them. A representative of this approach is our prior work on GOTECH [18]: a system that adds distributed capabilities to an existing program by generating AspectJ code using text templates.

The value and novelty of Meta-AspectJ can be described in two axes: its application value (i.e., the big-picture value for potential users) and its technical contributions (i.e., smaller reusable lessons for other researchers working on meta-programming tools). In terms of application value, MAJ is a useful meta-programming tool, not just for AspectJ but also for Java in general. Specifically:

- For generating either AspectJ or plain Java code, MAJ is safer than any text-based approach because the syntax of the generated code is represented explicitly in a typed structure.
- Compared to plain Java programs that output text, generators written in MAJ are simpler because MAJ allows writing complex code templates using quote/unquote operators.
- MAJ is the only tool for structured generation of AspectJ programs that we are aware of. Thus, to combine the benefits of generative programming and AspectJ, one needs to either use MAJ, or to use a text-based approach.

In terms of technical value, MAJ offers several improvements over prior meta-programming tools for Java. These translate to ease of use for the MAJ user, while the MAJ language design offers insights for meta-programming researchers:

- MAJ shows how to minimize the number of different quote/unquote operators compared to past tools, due to the MAJ mechanism for inferring the syntactic type (e.g., expression, declaration, statement, etc.) of a fragment of generated code. This property requires context-sensitive parsing of quoted code: the type of an unquoted variable dictates how quoted code should be parsed. As a result, the MAJ implementation is quite sophisticated and not just a naive precompiler. An additional benefit of this approach is that MAJ emits its own error messages, independently from the Java compiler that is used in its back-end.
- When storing fragments of generated code in variables, the user does not need to specify the types of these variables (e.g., whether they are statements, expressions, etc.). Instead, a special `infer` type can be used.

The above points are important because they isolate the user from low-level representation issues and allow meta-programming at the template level.

We next present an introduction to the MAJ language design (Section 2), discuss examples and applications (Section 3), describe in more depth the individual interesting technical points of MAJ (Section 4), and discuss related and future work (Section 5).