Virtual Machines and Intermediate Languages for Emerging Modularization Mechanisms (VMIL 2008)

Hridesh Rajan[λ], Michael Haupt[φ], Christoph Bockisch[β], Robert Dyer[λ]

[λ]Iowa State University, [φ]University of Potsdam, and [β]Technische Universität Darmstadt

[λ]{hridesh,rdyer}@cs.iastate.edu, [φ]michael.haupt@hpi.uni-potsdam.de, and [β]bockisch@cs.tu-darmstadt.de

Abstract

Modern programming languages are compiled to intermediate code preserving the intention of high-level language constructs. Emerging modularization mechanisms, however, lack such handling. Recent research results have shown that deeper support for these modularization mechanisms, e.g., in virtual machines and intermediate languages, is feasible; it allows applying tailored optimizations and radically improves development processes such as incremental compilation, debugging, etc.

The VMIL workshop, second in the series, is a forum for research in virtual machines and intermediate languages with support for emerging modularization mechanisms such as mix-ins, units, open classes, hyper-slices, adaptive methods, roles, composition filters, layers, pointcuts-and-advice, and inter-type declarations. Topics of interest include, but are not limited to: compilation-based and interpreter-based virtual machines as well as intermediate language designs with dedicated support for emerging modularization mechanisms, compilation techniques, optimization strategies, improved techniques for fast predicate evaluation (e.g., of pointcuts) inside virtual machines, and advanced caching and memory management schemes.

Categories and Subject Descriptors D.1.5 [*Programming Techniques*]: Object-oriented Programming; D.3.3 [*Programming Languages*]: Language Constructs and Features — Control structures; Procedures, functions, and subroutines; D3.4 [*Language Processors*]: Compilers, Interpreters, Memory Management, Run-Time Environments

General Terms Design, Languages, Performance

OOPSLA'08, October 19–23, 2008, Nashville, Tennessee, USA.
ACM 978-1-60558-220-7/08/10.

Keywords virtual machine, intermediate Language, dynamic dispatch, separation of concerns, compilation, interpretation, optimization

1. Motivation and Themes

Compilers for modern object-oriented languages generate portable intermediate code. The concepts of the source language are preserved in such intermediate representations. For example, classes, fields, methods, virtual or static dispatch, etc. are found, primarily because the underlying intermediate language directly supports such constructs. The advantage of preserving the information about the source language at the intermediate language level is that this information is now easily accessible to virtual machines, which exploit this information to perform optimizations that may not be enabled otherwise.

In contrast, emerging language constructs for improved separation of concerns [6] are often realized by compiling the new constructs to object-oriented instructions of the conventional intermediate language, which does not preserve the constructs' semantics in a declarative way. Examples of such new constructs include mix-ins [4], units [9], open classes [5], hyper-slices [5], adaptive methods [12], roles [11], composition filters [1], layers, pointcuts-and-advice [10, 14], and inter-type declarations.

The main themes of this workshop are enhancements of existing or development of new intermediate languages that better support new high-level language constructs, development and enhancement of virtual machines, and compilation techniques that take advantage of this newly found information in the intermediate representation for additional optimizations. Recent research results have shown that these new opportunities are valuable for optimization [2, 3], for efficiently supporting dynamic language constructs [8], and for speed of incremental compilation [7, 13].

The areas of interest include, but are not limited to: compilation-based and interpreter-based virtual machines as well as intermediate-language designs with better support for these emerging modularization mechanisms, compilation techniques from high-level languages to enhanced intermediate languages as well as native machine code, op-

timization strategies for reduction of run-time overhead due to either compilation or interpretation, improved techniques for fast predicate evaluation (e.g., of pointcuts) inside virtual machines, and advanced caching and memory management schemes in support of the mechanisms.

2. Relevance to OOPSLA

To date, the advanced modularization constructs researched in various communities are mostly reflected in high-level language design. The workshop's main goal is the discussion of compilation techniques, intermediate languages and execution environments that more naturally support these modularization constructs even within compiled programs. This support will, e.g., facilitate new optimization possibilities and incremental compilation, and improve debugging. Furthermore, it can be expected that advanced modularization constructs benefit from a more efficient execution: increased efficiency will raise the acceptance of the concepts which in turn activates further research at the conceptual level.

After having successfully conducted the VMIL workshop in conjunction with an AOSD conference once, we wish to take the opportunity to address the much wider audience attracted by the OOPSLA conferences. OOPSLA has, traditionally, brought together researchers and practitioners from various communities related to object-oriented and other styles of programming. The language design and implementation communities have frequently been among these.

3. Goals and Expected Results

We expect to receive papers that provide evidence of the benefits of supporting advanced modularization constructs beyond the high-level code of programs. These should act as motivation for new researchers to include the topics of this workshop into their research. We intend to accomplish this goal by publishing the papers in the ACM digital library and by opening the workshop to researchers without accepted papers, as both discussed below. The proceedings of the first workshop in this series have already been published in the ACM digital library.

We also expect to receive position papers from researchers new to the field. For these researchers we want to offer a platform for discussing their ideas and receiving feedback on them. This will be supported by question and answer sessions as well as by a session of group discussions.

4. Organizers

Hridesh Rajan is an Assistant Professor of Computer Science at the Iowa State University. He received his Ph.D. from the University of Virginia in 2005. He has published in several conferences such as ICSE, ESEC/FSE, ASE, AOSD, and IEEE Software. He has served on, or will serve on, the program committees of NWeSP 2009, AOSD 2009, ACP4IS 2008, FOAL 2008, FOAL 2006, and as a referee for top journals in his area such as IEEE Transactions on Software Engineering, ACM Transactions on Software Engineering and Methodology, and IEEE Software. He has also served as an external referee for several international conferences namely IEEE Infocom 2008, ESEC/FSE 2007, ICSE 2007, OOPSLA 2006, Formal Methods 2006, COMPSAC 2006, etc. and program committee of the FOAL 2006, VMIL 2007, FOAL 2008 workshops. He was also the organizer and co-chair of the 2007 edition of this workshop. His research on AOSD and on specification and verification languages has been funded by the US National Science Foundation. He is a member of the IEEE, the IEEE Computer Society, the ACM, SIGSOFT, and SIGPLAN.

Michael Haupt is a post-doctoral researcher in the Software Architecture Group at Hasso-Plattner-Institut in Potsdam. His research interests are in improving the modularity of complex software system architectures as well as in implementing programming languages, in which latter area his main focus is on faithfully regarding programming paradigms' core mechanisms as primary subjects of language implementation effort. Michael holds a doctoral degree from Technische Universität Darmstadt, where he has worked on the Steamloom virtual machine to provide runtime support for AOP languages. He has published papers on this and other AOSD-related subjects in the L'Objet and IEEE Software journals as well as in the AOSD, VEE, OOPSLA, and ECOOP conference series. Michael has served as PC member for ECOOP 2008, as reviewer for TAOSD, and has been supporting reviewer for the AOSD, ECOOP, ICSE, FSE, MODELS, and VEE conference series. He has co-organized the Dynamic Aspects Workshop series in conjunction with the AOSD conferences. Michael is a member of the ACM.

Christoph Bockisch is a post-doctoral researcher with a focus on the implementation of aspect-oriented programming languages. Being significantly involved in developing the Steamloom virtual machine he authored and co-authored several papers about compilation techniques and virtual machine support for aspect-oriented programming languages, published amongst others by the OOPSLA, AOSD, and VEE conferences. To provide virtual machine support, Christoph researches extensions to high-performing Java virtual machines based on just-in time compilation. In 2006 Christoph was member of the program committee of the Open and Dynamic Aspect Languages workshop at the AOSD conference and he was co-organizer of the 2007 edition of the VMIL workshop. He was supporting reviewer for the conferences Compiler Construction, AOSD and ECOOP. Christoph was also involved in teaching courses to graduate students about aspect-oriented software development, concepts of programming languages and virtual machines.

Robert Dyer is a second year Ph. D. student with a research focus on the design of intermediate language models and virtual machine support for advanced modulariza-

tion techniques. He is currently the lead developer on the Nu project, which includes an intermediate language model and virtual machine support for aspect-oriented programming languages. He has authored and co-authored papers about advanced intermediate languages and dedicated virtual machine caching mechanisms. Robert served as an external reviewer for FOAL 2008. He is a student member of ACM, SIGSOFT, and SIGPLAN.

5. Program Committee

The program committee of VMIL 2008 consists of:

- Eric Bodden (McGill University, Canada)
- Juan Chen (Microsoft Research, USA)
- Shigeru Chiba (Tokyo Institute of Technology, Japan)
- Sophia Drossopoulou (Imperial College, UK)
- Eric Eide (University of Utah, USA)
- Matthew Flatt (University of Utah, USA)
- Gregor Kiczales (University of British Columbia, Canada)
- Hidehiko Masuhara (University of Tokyo, Japan)
- Greg Morrisett (Harvard University, USA)
- Angela Nicoara (ETH Zurich, Switzerland)
- Harold Ossher (IBM Research, USA)
- Don Syme (Microsoft Research, UK)

and the organizers.

Acknowledgments

The organization of the workshop was supported in part by the US National Science Foundation under grants CNS 08-08913, CNS-0627354, and the AOSD-Europe Network of Excellence.

References

[1] L. Bergmans and M. Akşit. Principles and design rationale of composition filters. In R. E. Filman, T. Elrad, S. Clarke, and M. Akşit, editors, *Aspect-Oriented Software Development*, pages 63–95. Addison-Wesley, Boston, 2005.

[2] C. Bockisch, M. Arnold, T. Dinkelaker, and M. Mezini. Adapting virtual machine techniques for seamless aspect support. In *OOPSLA '06*, pages 109–124.

[3] C. Bockisch, M. Haupt, M. Mezini, and K. Ostermann. Virtual machine support for dynamic join points. In *AOSD '04*, pages 83–92.

[4] G. Bracha and W. Cook. Mixin-based inheritance. In *OOPSLA/ECOOP '90*, pages 303–311.

[5] C. Clifton, G. T. Leavens, C. Chambers, and T. Millstein. Multijava: modular open classes and symmetric multiple dispatch for java. In *OOPSLA '00: Proceedings of the 15th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, pages 130–145.

[6] E. W. Dijkstra. On the role of scientific thought. *EWD 477*, August 1974.

[7] R. Dyer, H. Narayanappa, and H. Rajan. Nu: preserving design modularity in object code. *SIGSOFT Softw. Eng. Notes*, 31(6):1–2, 2006.

[8] R. Dyer and H. Rajan. Nu: a dynamic aspect-oriented intermediate language model and virtual machine for flexible runtime adaptation. In *AOSD '08: Proceedings of the 7th international conference on Aspect-oriented software development*, pages 191–202.

[9] M. Flatt and M. Felleisen. Units: cool modules for hot languages. In *PLDI '98: Proceedings of the ACM SIGPLAN 1998 conference on Programming language design and implementation*, pages 236–248.

[10] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. V. Lopes, J.-M. Loingtier, and J. Irwin. Aspect-oriented programming. In *Proceedings of the European Conference on Object-Oriented Programming (ECOOP)*.

[11] B. B. Kristensen and K. Osterbye. Roles: conceptual abstraction theory and practical language issues. *Theor. Pract. Object Syst.*, 2(3):143–160, 1996.

[12] K. Lieberherr, D. Orleans, and J. Ovlinger. Aspect-Oriented Programming with Adaptive Methods. *Communications of the ACM*, 44(10):39–41, 2001.

[13] H. Rajan, R. Dyer, Y. Hanna, and H. Narayanappa. Preserving separation of concerns through compilation. In *Software Engineering Properties of Languages and Aspect Technologies (SPLAT 06), A workshop affiliated with AOSD 2006*.

[14] H. Rajan and K. J. Sullivan. Eos: instance-level aspects for integrated system design. In *ESEC/FSE-11*, pages 297–306.