COOMP 2011: First International Workshop on Combined Object-Oriented Modeling and Programming

Ole Lehmann Madsen  
Alexandra Institute & Aarhus University, Denmark  
ole.l.madsen@cs.au.dk

Birger Møller-Pedersen  
University of Oslo, Norway  
birger@ifi.uio.no

Ragnhild Kobro Runde  
University of Oslo, Norway  
ragnhilk@ifi.uio.no

Abstract

Languages for modeling and programming are diverging, with the implication that developers that would like to model (in order to raise the abstraction level and become independent of implementation platforms) end up with the challenge of maintaining both model and program artifacts. In addition, modeling is hampered by poor tool support compared with programming tools. The trend in programming languages is that less attention is paid to the fact that programming should be a kind of modeling, while executable models will not cover what programs usually cover. The aim of this workshop is to investigate requirements for combined modeling and programming languages, by identifying candidate elements that should be supported by such languages, propose potential new combined language mechanisms, and by investigating implementation techniques for such languages.

Categories and Subject Descriptors  D.3.2 [Object-oriented languages, Specialized application languages, Very high-level languages]

General Terms Languages

Keywords Modeling, programming, Simula, UML

1. Introduction

Languages for modeling and programming are diverging, with the following implications:

• On one hand developers who would like to apply OO design to obtain a suitable model end up with the challenge of maintaining both model and program artifacts. And, since many modeling languages are at the same level of abstraction as programming languages, there is little benefit to using a separate modeling language.

• On the other hand we see that much OO code is written by developers with little appreciation of OO design and development disciplines [10], leading to complex code that is difficult to understand and maintain as the concepts and phenomena of the application domain are not properly reflected in the code.

It has not always been like this. The very first object-oriented programming language, SIMULA, was also considered (and used) as a modeling language. This was a great step forward compared to the dominating methodologies of that time where different languages were used for analysis, design and implementation.

Both modeling and programming have evolved since the days of SIMULA. The aim of this workshop is to investigate requirements for combined modeling and programming languages as of today.

This includes conceptual means as well as language constructs for modeling and programming, identification of modeling constructs that are currently not supported by programming languages, proposals for programming language support for such constructs, new abstraction mechanisms to raise the level of abstraction, graphical versus textual syntax, tools supporting modeling and programming, and implementation techniques.

There are many good reasons for combined modeling and programming languages: Developers will not have to maintain both model and program artifacts; advanced language mechanisms (e.g. generics) that usually are developed for programming languages, become readily available for modeling; tools will require less effort (tools for executable models will have to compete with the best of tools for programming languages).

There a number of issues related to the design of such combined languages:

• Graphical/textual syntax While modeling languages are primarily graphical, and programming languages are primarily textual, an obvious requirement for such a new language is that it should have a mixture of graphical and textual syntax.

• Constraints While there are programming languages with support for e.g. both object-oriented and functional programming, constraints have so far not made it. An approach is described in [2].

• Language mechanisms Obviously some modeling mechanisms do not belong in such a language, or rather they are orthogonal to the (executable) elements of the language. Use Case models and Deployment models belong to this category. They apply to programming as well as to modeling, but with almost no implication on the semantics. Interaction Models (Sequence Diagrams), however, may not only be used to specify requirements, but also traces, and as such they can be used as the basis for the dynamic semantics of a combined language. State Machine models would belong to the language, one approach reported in [3]. However, they may to a large extent be provided through an implementation in a base language, but still should have special syntax. Some parts of Activity Models would also be part of such a language. Associations are obvious candidates, but despite some proposals ([4], [5]) they have not made it into programming languages.
The goals of the workshop are to contribute to the resolution of these issues. We are therefore seeking contributions that e.g.

- Analyze current mainstream programming and modeling languages, identifying candidate elements that should be supported by such a unified approach and how they should be supported by language mechanisms. Similarly, the approach will have to identify and understand programming language mechanisms that do not apply for modeling – and the other way around. Low-level implementation mechanisms may not apply to modeling just as non-executable mechanisms cannot directly become part of a programming language.
- Propose the combination of modeling and programming language mechanisms, and even new language mechanisms that may come about as part of this combination.
- Device ways of implementing such languages, especially how to handle that some of the modeling mechanisms may be implemented as frameworks in an underlying language, but still should be provided by means of special syntax.

Note that we are looking for general purpose modeling –and programming languages, not Domain Specific Languages (DSLs). These have many of the desired properties, but they have these on the expense that they are domain specific. With the exception of embedded DSLs, they are modeling languages where models are executable, usually with a restricted repertoire of language mechanisms. We see DSLs as important in certain situations, so we would also like the general-purpose language to have mechanisms that support the embedding of DSLs, as e.g. demonstrated in [6].

2. Tentative program

The program sketch so far is:

2. Dave Thomas and Thomas Weigert: Positions from a programming and a modeling point of view (titles to be decided)
3. One or all of James Noble, Andrew Black, Kim Bruce: The role of modeling in Next Educational Language [11] (title to be decided)
4. Klaus Østermann: Eliminating modeling from programming [9] (title to be decided)

In addition there will presentations and discussions on subjects such as:

1. Concurrency in relation to COOMP
2. Aspects in relation to COOMP
3. Constraints in relation to COOMP

3. Organisation

Program Chairs
- Ole Lehmann Madsen, Alexandra Institute & Aarhus University, Denmark
- Birger Møller-Pedersen, University of Oslo, Norway
- Ragnhild Kobro Runde, University of Oslo, Norway

Program Committee
- Bjorn Freeman-Benson, New Relic
- Boris Magnusson, Lund University

References

Contact
For questions about the workshop, please contact one of the program chairs or send an email to contact@coomp.org.
Abstract
Multicore programming is both prevalent and difficult. Industry programmers deal with large amounts of legacy code and are increasingly relying on multithreading to provide scalability. For legacy systems, it may not be possible to change this programming model. The Transitioning to MultiCore (TMC) workshop is focused on tools and systems for parallel programming that are interoperable with legacy code, that minimize the annotation burden for developers, and match well with current industry practice. We solicit industry experience reports about working or unworkable examples of such tools or systems, as well as research reports.

Categories and Subject Descriptors D.1.3 [Programming Techniques]: Concurrent Programming—Parallel Programming; D.2.2 [Software Engineering]: Design Tools and Techniques

General Terms Design, Reliability, Human Factors

Keywords Tools, Systems, Usability, Legacy Programs

1. Background
In the early 2000s, we hit a power wall; the energy output of a chip with increased processor speed has become untenable [1]. Today, all major chip manufacturers have switched to producing computers that contain more than one CPU [12]; parallel programming has rapidly moved from a special-purpose technique to standard practice in writing scalable programs. Taking advantage of parallel processors often entails using concurrent software, where multiple threads work simultaneously. However, concurrent software suffers from concurrency-specific errors, such as data races, atomicity violations, determinism violations, and deadlocks [4, 8, 9]. Achieving parallel performance is also difficult. In fact, in previous studies which compared parallel programming models or techniques a large subset of the participants in different groups did not successfully complete a correct solution that exhibited any speedup (e.g. [5, 10]). Furthermore, a large survey on current development practices found that a large portion of developers have to regularly deal with multithreaded code [6].

2. Main Theme and Goals
It is clear from the above discussion that multicore programming is both prevalent and difficult. To address that difficulty, numerous programming models and systems have been proposed, including transactional memory [7, 11], revisions [3], and type systems [2]. However, industry programmers face large amounts of legacy code, and so it may not always be feasible to change the programming model.

The TMC workshop is focused on tools and systems for parallel programming that are interoperable with legacy code, that minimize the annotation burden for developers, and match well with current industry practice. We solicit industry experience reports about working or unworkable examples of such tools or systems, as well as research reports. The topics for these reports may include:

- Surveys or empirical studies focused at measuring the current state of practice for multicore programming in industry
- Field studies identifying barriers and benefits to using existing tools
- Analysis tools focused on correctness, performance, or understandability analysis of existing programs
- New programming models which are interoperable with legacy multithreaded systems

We aim to bring together industry developers and researchers who are interested in improving the current transition to multicore.

3. Participant Preparation
We accept shorter, two to four page experience reports focused on experiences with scalable systems used in industry, or problems with existing systems. We also accept longer, four to six page research reports focused on development
of new systems, tools, or ideas in the multicore space. Additionally, we accept two page position papers focused on proposals for improving existing systems or tools. Although this is a small new workshop, we care about reviewers returning high-quality paper reviews and have picked our program committee accordingly.

4. Activities and Format

We plan to spend the morning on paper presentations. Each presentation slot will be approximately 10-15 minutes long, followed by a five minute question period.

We plan to start the afternoon with a panel presentation, moderated by the workshop organizers, comprising a mix of industry and academic panelists who can describe some of the challenges experienced with transitioning to multicore. The goal of this panel will be to highlight issues that may not be obvious within the research community.

Afterwards, we will facilitate a group discussion about what workshop participants feel are the largest issues raised in the workshop, and any issues they feel are not adequately addressed by current research literature. First we will break into focus groups (containing approximately 4-5 people per group) for about 30-45 minutes. These groups sessions will serve as a networking event so that participants will ideally make some new connections at the workshop. We aim to have at least one industry representative in each group. Each group will come together with 2-3 specific points for future or ongoing research which we will collate on the projected screen.

5. Organizers

Caitlin Sadowski (University of California at Santa Cruz)
Jaeheon Yi (University of California at Santa Cruz)

6. Program Committee

Michael Bond (Ohio State University)
Rachel Brill (IBM Haifa Research Lab)
Sebastian Burckhardt (Microsoft Research)
Joe Devietti (University of Washington)
Eitan Farchi (IBM Haifa Research Lab)
Benedict Gaster (AMD)
Ganesh Gopalakrishnan (University of Utah)
Shan Lu (University of Wisconsin - Madison)
Shankar Pasupathy (NetApp)
Neha Rungta (NASA Ames Research Center)
Koushik Sen (University of California, Berkeley)
Konstantin Serebryany (Google)
Stephen Toub (Microsoft, Parallel Computing Platform)

References

Abstract

Today’s Internet users expect to access Internet resources using increasingly capable and ubiquitous client platforms. This trend has resulted in a wide-ranging diversification of hardware devices supporting various form factors and interaction modes, a choice of web browsers offering varying levels of performance, security and standards compliance, as well as the emergence of domain-specific uses of general-purpose Internet-related technologies, exemplified by Rich Internet Applications (RIAs) and site-specific browsers.

Despite the heterogeneity, all these platforms implement a common set of standards and technologies. While the resulting high level of interoperability can be seen as a major reason for the Internet’s success, its constraints can also be viewed as limiting progress in client technologies. This workshop focuses on both innovative solutions in the area of Internet client software that improves on the current state-of-the-art while respecting the confines dictated by interoperability, as well as bold, new ideas that break with the status quo.

Categories and Subject Descriptors A.0 [GENERAL]: Conference Proceedings

General Terms Languages, Performance, Reliability, Security

Keywords Internet, Programming Languages, Systems, Compilation, Runtimes,

1. Overview

We firmly believe that, while the interest in the Internet-client related topics is on the rise, there is no single venue where people interested in these topics could meet, present their work, and exchange ideas. The main goal of this workshop is to fill out this niche. The workshop seeks contributions in the following (and also related) areas:

• compilation and runtime techniques for Internet client programming languages

• integration with server-side technologies, multi-tier programming languages and environments

• concurrency and parallelism support for Internet clients

• hardware acceleration of Internet client computational capabilities

• support for heterogeneity of Internet client environments (such as desktops, tablets and phones)

• Internet client security

• Internet client application deployment software engineering support (e.g. IDEs, refactoring, frameworks) for client-side Internet applications

• alternative Internet client programming languages and models

• novel approaches to Internet client software stack architecture

We solicit both regular papers (up to 10 pages) and position papers (up to 4 pages). The workshop will consist of a series of sessions where authors of the accepted papers will present ideas described in the papers. For the regular papers, we plan for 20-minute presentations and for the position papers we plan for 10-minute presentations. The intended time interval between the presentations would be at least 10 minutes. The the workshop will also include a panel session.

2. Organizers

The workshop is organized by two general co-chairs, Adam Welc (Adobe System) and Michael Franz (University of California, Irvine), and by the program committee chair, Krzysztof Palacz (Adobe Systems).

Adam Welc is a Senior Researcher at Adobe’s Advanced Technology Lab. Adam’s work is in the area of programming language design and implementation, with specific interests in web technologies, parallel programming and concurrency control, as well as compiler and runtime system optimizations. Some of his recent publications appeared in POPL’11, ECOOP’09 and EUROSYST’09.

Michael Franz is a Professor and the Director of the Secure Systems and Software Laboratory at the University of California, Irvine. He is well known for his research on dynamic compilation and continuous optimization. His work on trace-based compilation was subsequently adopted by Mozilla and became the TraceMonkey JavaScript engine in Firefox. He has been on the PCs of a large number of conferences and was one of the two founders of VEE, the ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments.

Krzysztof Palacz is a Senior Computer Scientist in the ActionScript Engineering group at Adobe, where he is currently the tech lead of the Flash runtime concurrency effort. His previous research work has been focused on efficient implementation of program-
ming languages and virtual machines, as well as reflection and communication frameworks. He is one of the authors of Lively Kernel, an Open Web malleable self-supporting application framework inspired by Smalltalk.

3. Program Committee
We have assembled a diversified program committee, consisting of leading experts in the field, coming from different institutions and backgrounds, and working on two different continents. The full list of program committee members is presented below:

- Ras Bodik (UC Berkeley)
- Andreas Gal (Mozilla)
- Brian Goetz (Oracle)
- Dan Ingalls (SAP)
- Chandra Krintz (UC Santa Barbara)
- Ben Livshits (MSR)
- Bernd Mathiske (Adobe)
- Mark Miller (Google)
- Florian Matthes (TU Munich)
- Tatiana Shpeisman (Intel)
- Laurence Tratt (Middlesex University)
- Jan Vitek (Purdue University)