

Linguistic Support for Modern Operating Systems Workshop on Programming Languages and Operating Systems 2006 (PLOS 2006)

Christian W. Probst

Technical University of Denmark
probst@imm.dtu.dk

Andreas Gal

University of California, Irvine, USA
gal@uci.edu

Robert Grimm

New York University, USA
rgrimm@cs.nyu.edu

Olaf Spinczyk

University of Erlangen-Nuremberg, Germany
Olaf.Spinczyk@informatik.uni-erlangen.de

Abstract

This report gives an overview over the Workshop on Programming Languages and Operating Systems (PLOS 2006), which was collocated with ASPLOS XII. It introduces the motivation for the workshop and gives a summary of the workshop contributions.

1. Introduction and Overview

The ASPLOS XII Workshop on Programming Languages and Operating Systems (PLOS) has its root in the ECOOP workshop series on Object-Oriented and Operating Systems (OOOSWS), which started in conjunction with ECOOP'97. Over the past years the number of participants of OOOSWS has grown constantly, and many high quality papers have been presented at this workshop, some of which were later published at relevant conferences and journals. In 2006 the PLOS workshop was held for the third time.

The workshop aims to bring together researchers and developers from the programming languages (PL) and the operating systems (OS) domain. It provides a platform for discussing new visions, challenges, experiences, problems, and solutions arising from the application of advanced programming and software engineering concepts to operating systems construction.

The collocation with ASPLOS and the location in Northern America aimed to make the workshop more visible in the operating systems community, and to reach a different audience than before. Considering these changes, the PLOS 2006 workshop can be considered successful in attracting 20 participants from 9 different countries.

The remaining parts of this report are organized as follows. Section 2 describes the workshop motivation. Section 3 briefly

summarizes the presentations given at the workshop. The report concludes with some final remarks in Section 4.

2. Motivation

Developing operating systems (OSs) is a highly complex task. OS programmers often have to deal with millions of lines of code and common OS issues like concurrency, performance optimization, real-time, deadlocks, and configurability make their work even harder.

Today, the historic language C—created in the early seventies—is still predominantly used to implement operating systems. Only in a few cases have OS implementors switched to more advanced languages like C++ or Java. Developing a new kernel and device drivers from scratch is often rendered impossible by the sheer size and complexity of operating systems. Thus, research is often limited to extend or modify existing systems. Widely deployed general purpose OSs like Linux and the Windows OS family continue to be developed using very conservative methods and languages. Modern software engineering concepts and languages, which are well-known and proven in other domains, are not adopted for the sake of performance optimization and backward compatibility. However, the arousing discussion about security and reliability of OSs, especially with respect to Internet attacks, is an example that shows the drawbacks of the traditional development approach and a demand for new ideas.

In this workshop we address this problem from the programming-language perspective and bring together researchers from both domains. The aim is to facilitate a lively discussion about novel approaches in OS construction based on language concepts in general. Examples are object-orientation, type safety, language support for OS verification, testing, debugging, separation of concerns by aspect-oriented programming, and domain-specific languages.

Prospective participants submit short position papers, covering topics like

- object-oriented OSs, type-safe languages for OS,
- separation of concerns in OS code, AOSD and OS,
- domain-specific languages for OS development, and

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

PLOS 2006, Oct. 22, 2006, San Jose, California, United States
Copyright © 2006 ACM 1-59593-577-0/10/2006...\$5.00

Session 1: Systems and Java OS
Atomicity and Visibility in Tiny Embedded Systems <i>John Regehr, Nathan Coopriker (University of Utah, USA), David Gay (Intel Research Berkeley, USA)</i>
Writing Solaris Device Drivers in Java <i>Hiroshi Yamauchi (Purdue University, USA), Mario Wolczko (Sun Microsystems Laboratories Menlo Park, USA)</i>
OSEK/VDX API for Java <i>Michael Stilkerich, Christian Wawersich, Wolfgang Schroder-Preikschat (University of Erlangen-Nuremberg, Germany), Andreas Gal, Michael Franz (University of California, Irvine, USA)</i>
Session 2: Languages & Language Extensions
SysObjC: C Extension for Development of Object-Oriented Operating Systems <i>Adam Balogh, Zoltan Csornyei (Eotvos Lorand University, Hungary)</i>
Efficient Type and Memory Safety for Tiny Embedded Systems <i>John Regehr, Nathan Coopriker, Will Archer, Eric Eide (University of Utah, USA)</i>
Type Inference for Unboxed Types and First Class Mutability <i>Swaroop Sridhar, Jonathan Shapiro (Johns Hopkins University, USA)</i>
A: An Assertion Language for Distributed Systems <i>Andrew Tjang, Fabio Oliveira, Richard. P. Martin, Thu D. Nguyen (Rutgers University, USA)</i>
Keynote
Programming Language Challenges in Systems Codes Why Systems Programmers Still Use C, and What to Do About It <i>Jonathan Shapiro (Johns Hopkins University, USA)</i>
Session 3: Design & Evolution
Semantic Patches for Documenting and Automating Collateral Evolutions in Linux Device Drivers <i>Rene Rydhof Hansen, Julia L. Lawall (University of Copenhagen, Denmark), Yoann Padioleau, Gilles Muller (Ecole des Mine de Nantes, INRIA, France)</i>
Portability Events: A Programming Model for Scalable System Infrastructures <i>Chris Matthews, Yvonne Coady (University of Victoria, Canada), Jonathan Appavoo (IBM Research)</i>
Issues in Holistic System Design <i>Julia L. Lawall (University of Copenhagen, Denmark), Christian W. Probst (Technical University of Denmark, Denmark), Ulrik Pagh Schultz (University of Southern Denmark, Denmark)</i>

Table 1. Workshop Program.

- language support for OS verification, testing, and debugging, static configuration, dynamic reconfiguration, and specialization.

3. Presentations

The presentations of the position papers were grouped in three sessions and the invited keynote. The complete workshop program is shown in Table 1. The rest of this section gives a short overview of the papers presented.

3.1 Session 1: Systems and Java OS

The workshop started with a session focusing on system properties and Java support for operating systems. The work presented in the first presentation extends the semantics of nesC’s atomic statement to include a visibility guarantee. This is reached by avoiding use of

the volatile qualifier, thereby enabling many optimizations needed to ensure visibility. The second talk presented an experimental implementation of a Java Virtual Machine running inside the kernel of the Solaris operating system. This approach allows to extend operating systems with elements written in type-safe languages, as compared to having to completely re-develop a type-safe OS. A similar approach was discussed in the last presentation of this session. The *KESO* system is a small and adaptable Java execution environment for an OSEK/VDX operating system. One of *KESO*’s properties is a low-overhead system interface, which allows to restrict system services to allow isolation by means of Java types.

3.2 Session 2: Languages & Language Extensions

The second session, which focused on new programming languages for OS development and language extensions, started with the pre-

resentation of a novel language extension called SysObjC. It introduces the concept of predicate classes into the C language and by this means helps to better map object layouts defined by the hardware to elements of the operating system's implementation language. The second presentation also remained in the world of C code. CCured was applied in order to achieve type and memory safety in a tiny embedded system. The authors presented an approach based on extensive static code analysis that reduces the cost of CCured significantly. With the third presented language we left the C domain. BitC is a higher-order programming language in the tradition of ML and Haskell, extended to incorporate both state and the expression of unboxed and low-level datatypes. The presentation concentrated on the fundamental conflict of goals between the ability to infer principal types and to allow freedom of mutability-compatibility at copy boundaries. The final presentation of session 2 introduced a novel assertion language for distributed systems. The language supports a formalized specification of correct behavior of complex distributed systems. It can be used to bolster system understanding, as well as help to flag operator mistakes.

3.3 Keynote

In the keynote session, John Shapiro discussed some of the aspects why systems programmers use programming languages that largely ignore the ongoing development in the programming languages community. Based on his experiences from developing microkernels such as EROS and Coyotos, he presented some of the challenges faced by systems designers that would need to be tackled by "acceptable" programming languages.

After defining the term "systems programming", John went on to present common false conclusions of the PL community, and then presented challenges and opportunities. Finally, he concluded that systems programmers *will* adopt a new language, if it gives them greater ability to understand and maintain the complex programs they write.

3.4 Session 3: Design & Evolution

The last session focused on design and evolution of systems. It started with a presentation of semantic patches, that allow to implement what the authors call *collateral evolution* of device drivers. The idea is to enable driver developers to specify the semantic changes to the interface of, e.g., support libraries. These semantic changes are then used for automated evolution of code depending on these libraries. The second talk presented portability events, which allow to program scalable system infrastructures. The SCOPE framework promotes clustered objects to the user level, as opposed to the specialized, operating-system specific environments usually needed. In the last presentation of this workshop, issues of holistic system design were discussed. This new approach to designing multi-layer systems is supposed to integrate requirements and facilities of the different layers, and to allow explicit reasoning about them. This is in contrast to the rather rigid stratification of current systems.

4. Final Remarks

As mentioned before, the move away from ECOOP and Europe was a result of discussions during PLOS 2005 in Glasgow. As the discussion during PLOS 2006 showed, this move seems to be appreciated by the communities, and as a result we will try to arrange locations alternating between programming language and operating system conferences, and between locations in North America and Europe.

PLOS 2006 repeated last year's interactive setup with short presentations of position papers. To a certain extent this has been a gamble as many workshops in both fields, operating systems and

programming languages, are more focused on paper presentation than actual discussion. Considering the record number of attendees, the interesting discussions resulting from the group presentations, and the feedback we got, it seems that the typical workshop atmosphere at PLOS was well received by the attendees.

From a distant perspective programming languages and operating systems are only small sub-disciplines of computer science. Thus, it might have been a surprise for some participants how different we speak, how different we get our motivation, and how different our solutions are. This diversity is good, but from time to time we should come together and share our latest ideas. This is essential for a research community to come up with mature solutions at the well-established conferences. We are convinced that some of the ideas presented at this PLOS workshop will follow this pattern.

Finally, we would like to thank the program committee members for their reviewing work, all authors for sharing their novel ideas, and all attendees for their participation. We also thank the ASPLOS XII organizers for their support and for hosting a workshop with a scope that—at least on paper—is so close to the main event.