M. Tokoro O. Nierstrasz P. Wegner (Eds.)

Object-Based Concurrent Computing

ECOOP '91 Workshop
Geneva, Switzerland, July 15-16, 1991
Proceedings

Springer-Verlag
Berlin Heidelberg New York
London Paris Tokyo
Hong Kong Barcelona
Budapest

Series Editors

Gerhard Goos
Universität Karlsruhe
Postfach 69 80
Vincenz-Priessnitz-Straße 1
W-7500 Karlsruhe, FRG

Juris Hartmanis
Department of Computer Science
Cornell University
5149 Upson Hall
Ithaca, NY 14853, USA

Volume Editors

Mario Tokoro
Department of Computer Science, Keio University
3-14-1 Hiyoshi, Kohoku-ku, Yokohama 223, Japan

Oscar Nierstrasz
Centre Universitaire d'Informatique, University of Geneva
24, rue du Général-Dufor, CH-1211 Geneva 4, Switzerland

Peter Wegner
Department of Computer Sciene, Brown University
P. O. Box 1910, Providence, Rhode Island 02912-1910, USA

CR Subject Classification (1991): D.1.3, D.1.5, D.3.2-3, F.3.2-3

ISBN 3-540-55613-3 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-55613-3 Springer-Verlag New York Berlin Heidelberg

Printed in Germany

Typesetting: Camera ready by author/editor
Printing and binding: Druckhaus Beltz, Hemsbach/Bergstr.
45/3140-543210 - Printed on acid-free paper

Preface

Background

The world we live in is concurrent, persistent, dynamic, distributed, and open-ended in its very nature. Besides, computation can be envisaged as simulation of a part of the real or an imaginary world, by computers. As hardware facilities for parallel and distributed computation, such as distributed computer networks and various multiprocessors, become more and more popular, large demands arise for providing users with computational frameworks, or tools of abstraction, for concurrent, distributed, and open-ended worlds.

The notion of objects gives us the framework of computation that provides the properties of boundary, identity, and persistence. If we examine these properties very carefully, we come to the conclusion that objects are inherently active, concurrent, and distributed, rather than passive, sequential, and centralized. Thus, a world can naturally be mapped onto a collection of objects and simulated as the mutual effects of objects.

Various attempts have been made, starting from sequential objects, to extend them to be concurrent. There has also been a view in which people look at computing as concurrent activities from the beginning, and regard a sequential one as a constrained form. Either way, concurrency has brought about a lot of new problems. However, attacking concurrency is inevitable and indispensable for establishing the basis of new computer science. We believe that concurrent object-based computing will open a new computational paradigm for the 1990s and beyond.

The Workshop

The ECOOP'91 Workshop on Object-Based Concurrent Computing was organized to provide a forum on concurrent, distributed, and open-ended computing. We put some emphasis on conceptual, theoretical, or formal aspects, as well as practical requirements and sound experience, since we deem that such a viewpoint is indispensable at this stage in order to investigate and to establish a common agreed-upon theoretical basis for further development.

The workshop was held on July the 15th and 16th, 1991 in the University of Geneva as a pre-conference workshop of ECOOP'91. The workshop comprised two invited lectures, twenty-five technical presentations, and a panel. We invited Professor Robin Milner of the University of Edinburgh and Professor Joseph Goguen of Oxford University to give special lectures, and they both kindly accepted our invitation.

Professor Milner's lecture was entitled *Concurrent Processes as Objects*. He presented his constructive view of concurrent computing and explained π-calculus which had been extended from CCS. He alluded to passing references, process migration, interpretation of λ-calculus by π-calculus, and the relationship between process creation and linear logic.

Professor Goguen's lecture was entitled *Semantics of the Object-Oriented Paradigm*. Based on his equilibrium view of concurrent computing, Goguen proposed the sheaf theory as a unified theoretical basis of concurrent and parallel computing. He explained how the sheaf theory is practically used by giving an example of logic design.

Unfortunately, we could not include the manuscripts or transcripts of the invited lectures. Professor Milner, however, participated in the panel discussion which was recorded and transcribed, so readers may capture his thought by tracing these arguments. Professor Goguen, upon our request after the workshop, submitted a technical paper on his related work, which was included in this volume. In this way, the readers can learn at least part of what he intended to convey in his lecture.

About this Volume

We selected 12 presentations out of 25 and asked the authors to revise their papers according to editors' comments. Thus, including Professor Goguen's paper, we gathered 13 papers and one transcribed panel discussion in this volume. The papers are classified into four categories: Formal Methods (1), Formal Methods (2), Concurrent Programming, and Models.

Formal Methods (1): The first three papers are concerned with the formal semantics of concurrent objects based on process calculi. The first paper, by Oscar Nierstrasz, is entitled *Towards an Object Calculus*. Nierstrasz first put his stress on the importance of the notions of concurrency, distribution and persistence in the study of object-oriented computing. He also deals with the computational and compositional aspects of concurrent programs and proposes an *object calculus* that integrates the concept of agents in process calculi with that of functions in λ-calculus.

The second paper is *On Asynchronous Communication Semantics* by Kohei Honda and Mario Tokoro. The authors first discuss the similarity and difference between synchronous and asynchronous systems, and then propose an equational theory called *asynchronous bisimulation* which is based on Milner's π-calculus. They also show that asynchronous bisimilarity is strictly more general than its synchronous counterpart.

The third paper is *A Unifying Framework for Process Calculus Semantics of Concurrent Object-Oriented Languages* by Michael Papathomas. From the viewpoint of designing better concurrent object-oriented languages, Papathomas claims the necessity of a common framework to discuss object-oriented features including encapsulation, object identity, class and inheritance, and concurrency. He proposes a framework for semantic definition of concurrent object-based languages by its translation to CCS, so that various concurrent object-oriented languages can be defined and compared. The framework also supports class and inheritance.

Formal Methods (2): The next four papers are concerned with various formal approaches to the semantics of concurrent programs. The first paper is an extended abstract entitled *A Sheaf Semantics for FOOPS Expressions* written by D. A. Wolfram and Joseph A. Goguen. They present a sheaf semantics for concurrent method expression evaluation in a concurrent object-oriented language called FOOPS, so that evaluations of functions, methods, and attributes are treated in a uniform way.

The next paper is *Semantic Layers of Object-Based Concurrent Computing* by Etsuya Shibayama. He proposes a layered semantics model for an object-based concurrent programming language. The bottom layer semantics is defined based on a transition system, the middle layer is defined based on an object diagram, and the top layer is defined based on the notion of program transformation. The model provides a means of reasoning about object composition.

The third paper is *Formal Techniques for Parallel Object-Oriented Languages* by Pierre America. It gives an overview of the formal techniques that have been developed to deal with the family of parallel object-oriented languages which are generally referred to as POOL. This is a slightly revised version of an invited paper for the Concur'91 Conference published in LNCS Vol. 527.

The last paper in this set is by Vasco Vasconcelos and Mario Tokoro, and is entitled *Trace Semantics for Actor Systems.* This paper describes their attempt to give concurrent semantics to Actor-like concurrent systems by using the theory of traces. They also develop a notion of composition of actor systems that allows the derivation of the semantics of a system from the semantics of its components.

Concurrent Programming: The third set is composed of three papers related to concurrent programming. The first one is written by Jean-Marc Andreoli, Remo Pareschi, and Marc Bourgois, and is entitled *Dynamic Programming as Multiagent Programming.* This paper is concerned with the well-known operations research technique of dynamic programming. The authors view the technique as a concurrent and truly dynamic system, and map the problem onto concurrent programs described in their LO (Linear Objects) programming language. For a given directed acyclic graph with weighted edges, the best path is found by cooperation and competition of multiple agents in a concurrent program.

The second paper is *Scheduling Predicate* by Ciaran McHale, Bridget Walsh, Sean Baker, and Alexis Donnelly. The authors describe a new synchronization mechanism intended to provide programmers with a facility for describing the scheduling of operations based on relative arrival times, values of parameters, and built-in synchronization counters.

The last paper in this set is *A Concurrency Control Mechanism for C++ Objects* by Hayssam Saleh and Philippe Gautron. They first study various synchronization mechanisms which were introduced into class-based languages and then propose a mechanism called *conditional wait* which is incorporated with C++. The incorporation is achieved orthogonally to the base language in such a way that it does not interfere with encapsulation, inheritance, and component reusability.

Models: The fourth set is composed of three papers concerned with models for concurrent systems. The first paper is written by Satoshi Matsuoka, Takuo Watanabe, Yuuji Ichisugi, and Akinori Yonezawa and is entitled *Object-Oriented Concurrent Reflective Architectures.* They introduce the notion of *reflection* into a concurrent language. Reflection provides the abilities of reasoning about and altering the dynamic behavior of computation from within the language framework. This paper first discusses the benefits of reflective architectures and then classifies previously proposed architectures into some categories. Then they present the current state of their work on ABCL/R and its future extensions.

The second paper is *Abstract Description of Distributed Object Systems* by Ralf Jungclaus and Thorsten Hartmann. They first propose an object-oriented model to describe distributed systems, called the Basic Object Model. The model consists of base objects for representing entities as processes and channels for communication. A language which describes distributed systems based on the Basic Object Model is presented.

In the last paper, *Design Issues for Object-Based Concurrency* by Peter Wegner, the author first examines the design space for object-based concurrent programming. Then, he considers the role of abstractions, distribution, and synchronization, and introduces the notion of *relative persistence* of operations and data for functions, objects, and trans-

actions.

The Panel: This volume concludes with a panel discussion, transcribed and edited by Kohei Honda and Satoshi Matsuoka. We had Peter Wegner as the chair and Pierre America, Robin Milner, Oscar Nierstrasz, Mario Tokoro, and Akinori Yonezawa as panelists. The title of the panel was *What is an Object?* Discussions ranged over various issues regarding objects including the definition of objects, identity, persistence, and concurrency. The readers of this volume may be able to discover and appreciate the underlying concepts of the panelists who have been leading research on Concurrent Object-Based Computing.

Acknowledgments

Lastly, but not least, we would like to thank Professor Dennis Tsichritzis, who served as the ECOOP'91 General Chair, for his support in organizing this workshop, Michael Papathomas for his devotion to local arrangements, Professors Robin Milner and Joseph Goguen for their insightful lectures, and all the authors and panelists who really materialized this volume . We are also grateful to Kohei Honda and Ichiro Satoh for their help in editing this volume.

April 1992

Mario Tokoro
Oscar Nierstrasz
Peter Wegner

Contents

Models

Towards an Object Calculus

Oscar Nierstrasz
University of Geneva*

Abstract

The development of concurrent object-based programming languages has suffered from the lack of any generally accepted formal foundations for defining their semantics. Furthermore, the delicate relationship between object-oriented features supporting reuse and operational features concerning interaction and state change is poorly understood in a concurrent setting. To address this problem, we propose the development of an *object calculus*, borrowing heavily from relevant work in the area of process calculi. To this end, we briefly review some of this work, we pose some informal requirements for an object calculus, and we present the syntax, operational semantics and use through examples of a proposed object calculus, called OC.

1 Introduction

In order for object-oriented languages to be an effective medium for implementing reusable software components for reactive applications, they must be able to cope with concurrency, distribution and persistence. Although distribution and persistence can arguable be considered as being purely run-time concerns, concurrency cannot, for it directly concerns the semantics of software composition. There have been numerous attempts in recent years to integrate concurrency features into object-oriented languages (see [33] for a survey). As a result of these experiences, a number of difficulties have become apparent:

1. Most concurrent object-oriented languages lack a well-defined semantic foundation. There is no generally accepted semantic domain or computational model for specifying such languages or for comparing their features. This naturally makes it quite difficult to reason about the abstract properties of any software component.

2. The clean integration of concurrency features with object-oriented features supporting encapsulation and reuse is difficult to achieve. In the particular case of inheritance, difficulties that arise in sequential languages due to confusion between

*Mailing address: Centre Universitaire d'Informatique, 12 Rue du Lac, CH-1207 Geneva, Switzerland.
E-mail: oscar@cui.unige.ch