22129

Control Flow and Data Flow: Concepts of Distributed Programming

Edited by

Manfred Broy

Control Flow and Data Flow: Concepts of Distributed Programming

International Summer School directed by F. L. Bauer, E. W. Dijkstra, C. A. R. Hoare

Edited by

Manfred Broy

Fakultät für Mathematik und Informatik Universität Passau, Postfach 2540
8390 Passau, Federal Republic of Germany

Springer-Verlag Berlin Heidelberg New York Tokyo
Published in cooperation with NATO Scientific Affairs Division

Proceedings of the NATO Advanced Study Institute on Control Flow and Data Flow: Concepts of Distributed Programming held at Marktoberdorf, Federal Republic of Germany, Juli 31–August 12, 1984

ISBN 3-540-13919-2 Springer-Verlag Berlin Heidelberg New York Tokyo
ISBN 0-387-13919-2 Springer-Verlag New York Heidelberg Berlin Tokyo

Library of Congress Cataloging in Publication Data.
NATO Advanced Study Institute on Control Flow and Data Flow: Concepts of Distributed Programming (1984: Marktoberdorf, Germany) Control flow and data flow. (NATO ASI series. Series F, Computer and system sciences; vol. 14) "Proceedings of the NATO Advanced Study Institute on Control Flow and Data Flow: Concepts of Distributed Programming held at Marktoberdorf, Federal Republic of Germany, July 31–August 12, 1984"–T. p. verso.
1. Electronic data processing–Distributed processing–Congresses. 2. Electronic digital computers–Programming–Congresses. I. Broy, M., 1949-. II. Title. III. Series: NATO ASI series. Series F, Computer and system sciences; no. 14.
QA76.9.D5N375 1984 001.64'4 85-8046
ISBN 0-387-13919-2 (U.S.)

Printing: Beltz Offsetdruck, Hemsbach; Bookbinding: J. Schäffer OHG, Grünstadt
2145/3140-543210

Series F: Computer and Systems Sciences Vol. 14

Preface

In a time of multiprocessor machines, message switching networks and process control programming tasks, the foundations of programming distributed systems are among the central challenges for computing scientists. The foundations of distributed programming comprise all the fascinating questions of computing science: the development of adequate computational, conceptual and semantic models for distributed systems, specification methods, verification techniques, transformation rules, the development of suitable representations by programming languages, evaluation and execution of programs describing distributed systems. Being the 7th in a series of ASI Summer Schools at Marktoberdorf, these lectures concentrated on distributed systems. Already during the previous Summer Schools at Marktoberdorf aspects of distributed systems were important periodical topics. The rising interest in distributed systems, their design and implementation led to a considerable amount of research in this area. This is impressively demonstrated by the broad spectrum of the topics of the papers in this volume, although they are far from being comprehensive for the work done in the area of distributed systems.

Distributed systems are extraordinarily complex and allow many distinct viewpoints. Therefore the literature on distributed systems sometimes may look rather confusing to people not working in the field. Nevertheless there is no reason for resignation: the Summer School was able to show considerable convergence in ideas, approaches and concepts for distributed systems. A ramifying mesh of theories and concepts of distributed systems starts to be laid bare, and connections between previously isolated ideas and approaches seem to materialize.

The work presented in this Summer School certainly represents a contribution to this development. I hope that this feeling is shared by the participants of the Summer School. And I hope that the Summer School helped them to develop and deepen their views on the nature of computing science. There was one contribution at the summer school that was directly devoted to this issue: the dinner speech by Prof. Dr. E.W. Dijkstra. For all those who could not attend the Summer School and for those who would like to read it the dinner speech is printed as the first contribution in this volume.

Manfred Broy

Passau, December 1984

Table of Contents

On the nature of computing science.

by Edsger W. Dijkstra
Dept. of Computer Sciences
University of Texas at Austin

A U S T I N, Texas 78712-1188
U.S.A.

Now this Summer School draws to a close, it seems appropriate to try to put its topic into some perspective.

Its official theme: "Control Flow and Data Flow: Concepts of Distributed Programming" only determined the flavour, for regularly we were lead to pretty general questions that seem rather central to computing in general. So, what is the nature of computing science, or, perhaps more precisely, what should its nature be?

There is *much confusion* about that, and that confusion should not amaze us, for computers nowadays have so many different aspects:

- You can view computers primarily as industrial products, intended to be sold with profit. A burning question then becomes whether to advertise them with Charlie Chaplin or with Mickey Mouse, and a computing science curriculum should contain a course in "Sales Promotion" as a major component.
- You can view computers primarily as the main battlefield of international competition; in that case a computing science curriculum should mainly consist in courses on "Security" and "Industrial Espionage".
- One day you can view automation as the motor of your economy, the other day you can view it as the greatest threat to the employment situation; so the main courses in our computing science curriculum should be on "Economy" and on "Industrial Relations".
- In recognition of the fact that the new technology has its profoundest impact by adding a totally new dimension to the eternal philosophical questions "Can machines think?" and "What is life?", we conclude that the major chairs in computing should be joint appointments between the departments of Philosophy, of Psychology, and of Biology.

The above enumeration is not exhaustive; I leave it to you to design and justify a dominant rôle in computing for the management scientists, the linguists, the experimental physicists and the educationists. This source of confusion about the nature of computing has now been sufficiently explained, so let us turn to the other component: *science*.

NATO ASI Series, Vol. F14
Control Flow and Data Flow: Concepts of Distributed Programming. Edited by M. Broy
© Springer-Verlag Berlin Heidelberg 1985

To begin with I would like to recall that Science as a whole has dismally failed to meet its original objectives..As you all remember, those original objectives were three.

Firstly, there was the development of the Elixir that would give you eternal youth.

But since there is little point in living in eternal poverty, the second objective was the development of the Philosopher's Stone, by means of which you could make as much gold as you needed.

As you can imagine, the planning of these two grandiose research projects, the Elixir and the Stone, required more foresight than could be provided by the seers of the day and Accurate Prediction of the Future became the third hot scientific topic.

We all know that, as the centuries went by, medicine divorced itself from quackery - at least we hope so! - , that chemistry divorced itself from alchemy and that astronomy divorced itself from astrology. This is just another way of saying that, for tactful reasons, the original objectives were forgotten.

Were they? No, not entirely. Evidently there is still a lingering sense of guilt for, as soon as a promising science or technology emerges, the old objectives are suddenly remembered and the young science is expected to meet them. To which we may add that, the less the new science is understood, the higher these expectations. We all know, how computing is now expected to cure all ills of the world and more, and how, as far as these expectations are concerned, even the sky is no longer accepted as the limit.

The analogy raises, for instance, the questions which current computer-related research will later be identified as computing's alchemy, and whether this identification can be speeded up, but I shall leave these questions for your speculation and shall approach the matter from another direction.

Since the rules of the academic game are remarkably strict and stable, we can pursue the question, which aspects of computing could ensure its viability as an academic discipline. Here I am in a much better position since I developed ten years ago a well-established theory about the viability of academic disciplines.

This theory tells us that if computing is to develop into a viable academic discipline, it has to be made into an unusually formal branch of mathematics, in which knowledge of facts will play a relatively minor rôle but methodological concerns will play an unusually large one. As a consequence there will be no significant distinction between "pure" and "applied" computing science. The current dictionary definition of mathematics "the science of shape, number, and quantity" - which is already obsolete now - will in the long run be replaced by "the art and science of effective reasoning", and when that has happened - a century from now or so - computing science will have had a much profounder influence on mathematics at large than physics has had in the past.

All this is very exciting and also very inviting because mathematical elegance will play such a central rôle. As computing scientists we know that in our area, perhaps more than everywhere else, mathematical elegance is not a dispensable luxury but decides between success and failure.

It is nice to know the dictionary definition for the adjective "elegant" in the meaning "simple and surprisingly effective".

But before above rosy future will have been reached, some hurdles have to be taken. Simplicity is a great virtue but it requires hard work to achieve it and education to appreciate it. And to make matters worse: complexity sells better. The computing industry is not the only one that has discovered that sore truth: so has the academic world. If you deliver a lecture that is crystal-clear from beginning to end, your audience feels cheated and mutters while leaving the lecture hall "That was all rather trivial, wasn't it?" One of our learned journals has rejected a beautiful paper of mine because the solution it presented was "too simple to be of academic interest" and I am waiting for the rejection of a paper on the grounds that it is too short.

Also our academic reward system works against us. One can get credit for some complicated concepts one has introduced, it is hard to get credit for the discovery how some established but complicated concepts had better be avoided: those unaware of these concepts won't notice your discovery and those with vested interest in them will hate you for it. Hence my urgent advice to all of you to reject the morals of the best-seller society and to find, to start with, your reward in your own fun. This is quite feasible, for the challenge of simplification is so fascinating that, if we do our job properly, we shall have the greatest fun in the world.

In short: Two Cheers for Elegance!

Marktoberdorf, 10 August 1984

PART I

Operational Models of Distributed Systems

A significant property of distributed systems is that they work in parallel. Parallelism is mainly an operational concept. Therefore it seems important to study the concepts for describing the operational behaviour of distributed systems. The description of the operational behaviour of parallel systems always comprises the description of sequential systems as a special case It is a heavily discussed question whether true parallelism is to be represented in an operational semantics of distributed systems or whether an interleaving semantics that models parallelism by nondeterminism over sequential behaviour is sufficient. Parallelism can be expressed via partial orderings on event structures.

But when dealing with distributed systems parallelism is not the only issue. The distributed activities generally influence each other by communication or by synchronisation. Therefore for distributed systems it is typical that the distinct parts of distributed systems, that are to be evaluated separately, do not have available all the information they need, and therefore evaluating the parts of distributed systems always means computing with incomplete information about the input. A typical technique for dealing with these problems is mixed computation.

U. Montanari

A. Ershov

DISTRIBUTED SYSTEMS, PARTIAL ORDERINGS OF EVENTS, AND EVENT STRUCTURES

Pierpaolo Degano and Ugo Montanari
Dipartimento di Informatica
Università di Pisa

Abstract

These lecture notes are divided in two parts, dedicated to two models, concurrent histories and Graphs for Distributed Systems (GDS), both based on partial orderings. The models are largely consistent, the latter being a richer version of the former, conceived as a specification formalism for distributed systems. The semantic aspects of the first model are studied in finer detail, including properties of non terminating computations and the definition of an observational equivalence.

Work partially supported by Consiglio Nazionale delle Ricerche, Progetto Finalizzato Informatica, Obiettivo Cnet.

TABLE OF CONTENTS