

Lecture Notes in Computer Science

Edited by G. Goos and J. Hartmanis

86

Abstract Software Specifications

1979 Copenhagen Winter School
Proceedings

Edited by D. Bjørner



Springer-Verlag
Berlin Heidelberg New York

Lecture Notes in Computer Science

Edited by G. Goos and J. Hartmanis

86

Abstract Software Specifications

1979 Copenhagen Winter School
January 22 – February 2, 1979
Proceedings

Edited by D. Bjørner



Springer-Verlag
Berlin Heidelberg New York 1980

Editorial Board

W. Brauer P. Brinch Hansen D. Gries C. Moler G. Seegmüller
J. Stoer N. Wirth

Editor

Dines Bjørner

Department of Computer Science, Technical University of Denmark
DK-2800 Lyngby/Denmark

AMS Subject Classifications (1979): 68-02, 68A05, 68A30

CR Subject Classifications (1974): 5.21, 5.23, 4.20

ISBN 3-540-10007-5 Springer-Verlag Berlin Heidelberg New York

ISBN 0-387-10007-5 Springer-Verlag New York Heidelberg Berlin

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically those of translation, reprinting, re-use of illustrations, broadcasting, reproduction by photocopying machine or similar means, and storage in data banks. Under § 54 of the German Copyright Law where copies are made for other than private use, a fee is payable to the publisher, the amount of the fee to be determined by agreement with the publisher.

© by Springer-Verlag Berlin Heidelberg 1980

Printed in Germany

Printing and binding: Beltz Offsetdruck, Hemsbach/Bergstr.
2145/3140-543210

Lecture Notes in Computer Science

- Vol. 1: GI-Gesellschaft für Informatik e.V. 3. Jahrestagung, Hamburg, 8.–10. Oktober 1973. Herausgegeben im Auftrag der Gesellschaft für Informatik von W. Brauer. XI, 508 Seiten. 1973.
- Vol. 2: GI-Gesellschaft für Informatik e.V. 1. Fachtagung über Automatentheorie und Formale Sprachen, Bonn, 9.–12. Juli 1973. Herausgegeben im Auftrag der Gesellschaft für Informatik von K.-H. Böhlting und K. Indermark. VII, 322 Seiten. 1973.
- Vol. 3: 5th Conference on Optimization Techniques, Part I. (Series: I.F.I.P. TC7 Optimization Conferences.) Edited by R. Conti and A. Ruberti. XIII, 565 pages. 1973.
- Vol. 4: 5th Conference on Optimization Techniques, Part II. (Series: I.F.I.P. TC7 Optimization Conferences.) Edited by R. Conti and A. Ruberti. XIII, 389 pages. 1973.
- Vol. 5: International Symposium on Theoretical Programming. Edited by A. Ershov and V. A. Nepomniashchy. VI, 407 pages. 1974.
- Vol. 6: B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow, Y. Ikebe, V. C. Klema, and C. B. Moler, Matrix Eigensystem Routines – EISPACK Guide. XI, 551 pages. 2nd Edition 1974. 1976.
- Vol. 7: 3. Fachtagung über Programmiersprachen, Kiel, 5.–7. März 1974. Herausgegeben von B. Schlender und W. Frielinghaus. VI, 225 Seiten. 1974.
- Vol. 8: GI-NTG Fachtagung über Struktur und Betrieb von Rechensystemen, Braunschweig, 20.–22. März 1974. Herausgegeben im Auftrag der GI und der NTG von H.-O. Leilich. VI, 340 Seiten. 1974.
- Vol. 9: GI-BIFOA Internationale Fachtagung: Informationszentren in Wirtschaft und Verwaltung. Köln, 17./18. Sept. 1973. Herausgegeben im Auftrag der GI und dem BIFOA von P. Schmitz. VI, 259 Seiten. 1974.
- Vol. 10: Computing Methods in Applied Sciences and Engineering, Part 1. International Symposium, Versailles, December 17–21, 1973. Edited by R. Glowinski and J. L. Lions. X, 497 pages. 1974.
- Vol. 11: Computing Methods in Applied Sciences and Engineering, Part 2. International Symposium, Versailles, December 17–21, 1973. Edited by R. Glowinski and J. L. Lions. X, 434 pages. 1974.
- Vol. 12: GFK-GI-GMR Fachtagung Prozessrechner 1974. Karlsruhe, 10.–11. Juni 1974. Herausgegeben von G. Krüger und R. Friehmelt. XI, 620 Seiten. 1974.
- Vol. 13: Rechnerstrukturen und Betriebsprogrammierung, Erlangen, 1970. (GI-Gesellschaft für Informatik e.V.) Herausgegeben von W. Händler und P. P. Spies. VII, 333 Seiten. 1974.
- Vol. 14: Automata, Languages and Programming – 2nd Colloquium, University of Saarbrücken, July 29–August 2, 1974. Edited by J. Loeckx. VIII, 611 pages. 1974.
- Vol. 15: L Systems. Edited by A. Salomaa and G. Rozenberg. VI, 338 pages. 1974.
- Vol. 16: Operating Systems, International Symposium, Rocquencourt 1974. Edited by E. Gelenbe and C. Kaiser. VIII, 310 pages. 1974.
- Vol. 17: Rechner-Gestützter Unterricht RGU '74, Fachtagung, Hamburg, 12.–14. August 1974, ACU-Arbeitskreis Computer-Unterstützter Unterricht. Herausgegeben im Auftrag der GI von K. Brunnstein, K. Haefner und W. Händler. X, 417 Seiten. 1974.
- Vol. 18: K. Jensen and N. E. Wirth, PASCAL – User Manual and Report. VII, 170 pages. Corrected Reprint of the 2nd Edition 1976.
- Vol. 19: Programming Symposium. Proceedings 1974. V, 425 pages. 1974.
- Vol. 20: J. Engelfriet, Simple Program Schemes and Formal Languages. VII, 254 pages. 1974.
- Vol. 21: Compiler Construction, An Advanced Course. Edited by F. L. Bauer and J. Eickel. XIV, 621 pages. 1974.
- Vol. 22: Formal Aspects of Cognitive Processes. Proceedings 1972. Edited by T. Storer and D. Winter. V, 214 pages. 1975.
- Vol. 23: Programming Methodology. 4th Informatik Symposium, IBM Germany Wildbad, September 25–27, 1974. Edited by C. E. Hackl. VI, 501 pages. 1975.
- Vol. 24: Parallel Processing. Proceedings 1974. Edited by T. Feng. VI, 433 pages. 1975.
- Vol. 25: Category Theory Applied to Computation and Control. Proceedings 1974. Edited by E. G. Manes. X, 245 pages. 1975.
- Vol. 26: GI-4. Jahrestagung, Berlin, 9.–12. Oktober 1974. Herausgegeben im Auftrag der GI von D. Siefkes. IX, 748 Seiten. 1975.
- Vol. 27: Optimization Techniques. IFIP Technical Conference. Novosibirsk, July 1–7, 1974. (Series: I.F.I.P. TC7 Optimization Conferences.) Edited by G. I. Marchuk. VIII, 507 pages. 1975.
- Vol. 28: Mathematical Foundations of Computer Science. 3rd Symposium at Jadwisin near Warsaw, June 17–22, 1974. Edited by A. Blikle. VII, 484 pages. 1975.
- Vol. 29: Interval Mathematics. Proceedings 1975. Edited by K. Nickel. VI, 331 pages. 1975.
- Vol. 30: Software Engineering. An Advanced Course. Edited by F. L. Bauer. (Formerly published 1973 as Lecture Notes in Economics and Mathematical Systems, Vol. 81) XII, 545 pages. 1975.
- Vol. 31: S. H. Fuller, Analysis of Drum and Disk Storage Units. IX, 283 pages. 1975.
- Vol. 32: Mathematical Foundations of Computer Science 1975. Proceedings 1975. Edited by J. Bečvář. X, 476 pages. 1975.
- Vol. 33: Automata Theory and Formal Languages, Kaiserslautern, May 20–23, 1975. Edited by H. Brakhage on behalf of GI. VIII, 292 Seiten. 1975.
- Vol. 34: GI – 5. Jahrestagung, Dortmund 8.–10. Oktober 1975. Herausgegeben im Auftrag der GI von J. Mühlbacher. X, 755 Seiten. 1975.
- Vol. 35: W. Everling, Exercises in Computer Systems Analysis. (Formerly published 1972 as Lecture Notes in Economics and Mathematical Systems, Vol. 65) VIII, 184 pages. 1975.
- Vol. 36: S. A. Greibach, Theory of Program Structures: Schemes, Semantics, Verification. XV, 364 pages. 1975.
- Vol. 37: C. Böhm, λ -Calculus and Computer Science Theory. Proceedings 1975. XII, 370 pages. 1975.
- Vol. 38: P. Branquart, J.-P. Cardinael, J. Lewi, J.-P. Delescaille, M. Vanbegin. An Optimized Translation Process and Its Application to ALGOL 68. IX, 334 pages. 1976.
- Vol. 39: Data Base Systems. Proceedings 1975. Edited by H. Hasselmeier and W. Spruth. VI, 386 pages. 1976.
- Vol. 40: Optimization Techniques. Modeling and Optimization in the Service of Man. Part 1. Proceedings 1975. Edited by J. Cea. XIV, 854 pages. 1976.
- Vol. 41: Optimization Techniques. Modeling and Optimization in the Service of Man. Part 2. Proceedings 1975. Edited by J. Cea. XIII, 852 pages. 1976.
- Vol. 42: James E. Donahue, Complementary Definitions of Programming Language Semantics. VII, 172 pages. 1976.
- Vol. 43: E. Specker and V. Strassen, Komplexität von Entscheidungsproblemen. Ein Seminar. V, 217 Seiten. 1976.
- Vol. 44: ECI Conference 1976. Proceedings 1976. Edited by K. Samelson. VIII, 322 pages. 1976.
- Vol. 45: Mathematical Foundations of Computer Science 1976. Proceedings 1976. Edited by A. Mazurkiewicz. XI, 601 pages. 1976.
- Vol. 46: Language Hierarchies and Interfaces. Edited by F. L. Bauer and K. Samelson. X, 428 pages. 1976.
- Vol. 47: Methods of Algorithmic Language Implementation. Edited by A. Ershov and C. H. A. Koster. VIII, 351 pages. 1977.
- Vol. 48: Theoretical Computer Science, Darmstadt, March 1977. Edited by H. Tzschach, H. Waldschmidt and H. K.-G. Walter on behalf of GI. VIII, 418 pages. 1977.

DEDICATION

This volume is dedicated to:

PROFESSOR, DR. TECHN. HEINZ ZEMANEK

The dedication occurs in connection with Professor Zemanek's 60th Anniversary.

The dedication is motivated by Professor Zemanek's inspired founding, and inspiring leadership of the IBM Vienna Laboratory.

The applied scientific work of the Vienna Laboratory could not have taken place had it not been for Zemanek's continued attention. Examples of this work are: the VDL (Vienna Definition Language) based, operational semantics definition of PL/I, of the mid-to-late 1960s; and the therefrom distinct VDM/META-IV (Vienna Development Method / Meta Language) based denotational semantics definition of a PL/I subset, of the early-to-mid 1970s. The editor of this volume, the co-director of the 1979 Copenhagen Winter School on 'Abstract Software Specifications', and many others, derived from - shorter or longer stays with - the Vienna Laboratory of those periods, long and lasting impressions and inspirations.

The editor of this volume is grateful to be able to open and close this volume with papers by Professor Zemanek. Both were presented at the above-mentioned Winter School. The editor is also grateful to the Springer-Verlag for the opportunity and kind permission to present this dedication.

ACKNOWLEDGEMENTS

The present collection of sixteen papers relate to the subject of ABSTRACT SOFTWARE SPECIFICATIONS. This was the theme of a two week Winter School held at the Technical University of Denmark, January 22 - February 2, 1979. The 'School' also featured a number of workshop sessions, and drew some 120 participants from 23, mostly European, countries. Most of the papers derive directly from series of lectures or workshop seminar presentations given.

The event was sponsored by, and held under the auspices of:

- (1) the Commission of the European Communities, in particular the EEC/CREST Subcommittee on Education in Informatics, and
- (2) the Danish Research Councils for the Natural- and the Technical Sciences, colloquially known as SNF & STVF.

Most valuable direct, as well as indirect (participant scholarship) financial support was also received from:

(3) the edp group ('dif data') of the Danish Engineering Society, and from the following, national IBM Companies:

- (4) IBM Denmark,
- (5) IBM Finland,
- (6) IBM Norway,
- (7) IBM Sweden, and
- (8) The Netherlands IBM (Holland).

The School Directors, D.Bjørner & C.B.Jones, wishes hereby to extend their most sincere thanks to the EEC/CREST, the SNF+STVF, the DIF-DATA, and especially to the above IBM national companies whose generous understanding and prompt contributions was most appreciated.

The Director of the Winter School finally thanks the administration & technical services of the Technical University of Denmark for having provided such outstanding facilities and services.

FOREWORD

The subject of abstractly specifying software - before embarking upon costly realizations - is presently being firmly established. Not just in University Computer Science Curricula, but also as an engineering practice in small and large corporations.

This volume records a number of software abstraction and design methods, their mathematical foundations and use. Common to the methods dealt with in this volume is their reliance on mathematical foundations. This also sets these methods apart from most other recorded means of software specification.

Most papers, with the exceptions being papers number 5, 10 and 15, cover rather exactly lectures and shorter seminars given at the 1979 Copenhagen Winter School on Abstract Software Specifications. Professors Liskov and Plotkin lectured on 'Abstractions in CLU', respectively 'Towards a Mathematical Theory of Concurrently Executing Programs'. Plotkin's present paper is an elaboration of only a part of his lectures at the Winter School.

The Winter School was held at the Technical University of Denmark, in the period: January 22 - Februar 2, 1979.

BROAD CLASSIFICATION

Two main streams of definitional styles are identified: the constructive methods based on the Scott-Strachey approach to Mathematical Semantics, and the Algebraic Semantics methods. The papers by Stoy, Jones, Bjørner, Park and Plotkin belong to the former school; with the papers by Zilles, Dahl, Burstall & Goguen, Dömölki and Liskov, the latter indirectly, belonging to the latter school.

Two hitherto separate areas of application are identified: specification of essentially deterministic, sequential, respectively non-deterministic, parallel-process oriented systems. This is admittedly a rather gross delineation. The papers by Dahl, Lauer et.al., Park and Plotkin address the latter issues, while remaining papers primarily, if not exclusively, stay within a simpler, non-power domain of discourse (!).

PAPER OVERVIEWS

The opening paper by ZEMANEK: "ABSTRACT ARCHITECTURE" relates the task of the computer and software systems architects to that of 'conventional' (i.e. building) architects; investigates the nature of design; of systems and their components; analyzes the notions of in- & formality; etc.. It is a thought provoking paper which, in very relevant terms, is a contribution towards the philosophy- and the theory of science of computer science and software engineering. Awareness of the many points brought up by Zemanek should lead to better suited, more appropriately proportioned systems serving satisfied users.

STOYS paper on the "FOUNDATIONS OF DENOTATIONAL SEMANTICS" opens the part on constructive definition methods. It provides an elementary introduction to the mathematical theory underlying such constructive definition methods, & thus, in particular, the papers by Jones and Bjørner. The papers of Park and Plotkin are more advanced treatises, extending & applying these foundations in the search for answers to and characterizations of crucial notions in non-deterministic & parallel programs. Stoys paper also discusses techniques for reasoning about denotational semantics definitions, and for modelling GOTOs via the technique of so-called continuations. A section of Jones' paper ('Escape Mechanisms') and Bjørners 2nd paper (numbered:5) deals with another way of modelling GOTOs - the so-called exit-mechanism. Bjørners paper (no.5) also combines exit & continuation modelling techniques.

JONES's paper is an introduction to techniques, and a notation, for "MODELING PROGRAMMING LANGUAGE CONCEPTS". The paper unfolds the notation and techniques required, by covering concepts in a stepwise, first orthogonal, subsequently combined fashion.

BJØRNERs paper on "FORMALIZATION OF DATA BASE MODELS" provides an alternative introduction, but now to techniques for modelling Data Base concepts. Not that there is any significant difference! But the audience might be different. Jones' paper emphasizes understanding the modelling techniques and motivates (desired properties of) the notational constructs. Bjørners paper emphasizes the application of these techniques to other than the 'classical' area (of programming languages). Jones' paper, in a sense, assumes some familiarity of the exemplified (source) language constructs; while Bjørners paper can be read as an alternative intro=

duction to Data Base concepts for persons not familiar with these, and as an introduction to modelling techniques for Data Base professionals.

BJØRNERs paper on "EXPERIMENTS IN BLOCK-STRUCTURED GOTO LANGUAGE MODELING: EXITS VERSUS CONTINUATIONS" is a mere exercise in expressing GOTO semantics. It starts with the so-called exit-based modelling techniques motivated in Jones' paper. It then exemplifies 'corresponding' continuation-based models; and finally 'merges' these styles!

Summarizing the papers by STOY, JONES and BJØRNER, we can say that with this volume two prominent variations on the theme of expressing mathematical semantics has been brought together: The VDM (Vienna Development Method) and the Oxford Styles of Denotational Semantics. Stoy's paper clearly points out some differences, but is otherwise a contribution to a unified understanding of their foundations.

The mostly theoretical paper by ZILLES, "AN INTRODUCTION TO DATA ALGEBRA", opens the part on Algebraic Semantics. It provides an advanced level introduction to the mathematics underlying algebraic presentations of abstract data types. It is a long expected paper from one of the first researchers of this most fascinating and booming area.

DAHLs paper, "TIME SEQUENCES AS A TOOL FOR DESCRIBING PROGRAM BEHAVIOUR", explores the (time) sequence concept of e.g. programming languages, using techniques akin to those treated by Zilles. The aim is to provide a "tool kit" for speaking about 'operators, functions and predicates on sequences', aiding practicing programmers in program specification, mechanization and proofs. Use of the established tools are then demonstrated by applications to specification & proofs concerning semaphores, mutual exclusion, deadlock, and the classical readers/writers problem.

"THE SEMANTICS OF CLEAR, A SPECIFICATION LANGUAGE", by BURSTALL & GOGUEN, defines a basically algebraic specification language, CLEAR. CLEAR permits the configured, bottom-up, as well as the hierarchical, top-down, construction of abstract models, put together, respectively derived from models of constituent, respectively overall concepts. CLEAR is here defined using 'a blend of denotational semantics with categorical ideas'.

DÖMÖLKIs paper, "AN EXAMPLE OF HIERARCHICAL PROGRAM SPECIFICATION", applies ideas of CLEAR to the development of a program: specification, realization and correctness proofs.

"MODULAR PROGRAM CONSTRUCTION USING ABSTRACTIONS", by LISKOV, 'presents a programming method in which modular decomposition is based on recognition of useful abstractions'. The paper is structured around a very instructive specification & implementation example.

LUCAS's paper "ON THE STRUCTURE OF APPLICATION PROGRAMS" is concerned with the 'parameterization of programs with respect to factual information'. It reports on 'software techniques which can be expected to facilitate programming and maintenance of commercial applications'. The paper also 'sheds light on the rôle of formalization, and the rôles and proper place of abstract data types'.

The joint paper by GERSTMANN & OLLONGREN has been included in this volume since it attempts to analyze, from one viewpoint, basic notions of the VDL-, the VDM- and the Algebraic Schools of Software Specifications.

The editor would here like to take the opportunity to warn the reader of a possible source of confusion. VDL is not VDM! The former stands for the notation language used for the operational semantics definitions of the 1960s. The latter acronym for a whole development method starting with denotational semantics definitions. VDL reads: Vienna Definition Language. VDM reads: Vienna Development Method. The notational system, or the semantics definition meta-language of VDM has been referred to by the acronym: META-IV.

'COSY' is a language for the "DESIGN & ANALYSIS OF HIGHLY PARALLEL & DISTRIBUTED SYSTEMS". It is based on Petri-net like concepts, and is derived from regular expressions. In their paper, LAUER, SHIELDS & BEST, introduces the 'COSY' notation (Lauer), gives the net semantics of 'COSY' (Best), and presents firing sequence- and adequacy properties of 'COSY' (Shields).

The last two technical papers, by Park and Plotkin, focuses on very specific, mathematical problems of dealing with parallelism and non-determinism.

PARK applies the relational semantics variant of denotational semantics in his paper "ON THE SEMANTICS OF FAIR PARALLELISM". In it, he analyzes fairness, or finite delay properties of processes, and 'unbounded, but finite', and 'potentially infinite' attributes of parallel systems specifications.

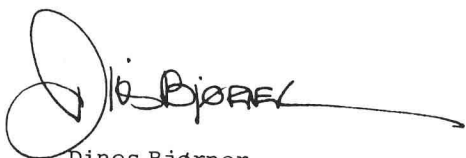
Abstract (Machine, Abstract Operational) Non-deterministic State Transformation Semantics explications of "DIJKSTRAS PREDICATE TRANSFORMERS" have been provided by e.g. de Roever, de Bakker, Wand and Bäck. PLOTKINS paper, whose title finishes with: "& SMYTHS POWER DOMAINS", 'regards this by showing homo- and isomorphisms from the state transformation view to the predicate transformer view'.

As a prerequisite for more fully enjoying the closing "BANQUET TALK" paper by ZEMANEK, the reader should be informed that (i) the above-mentioned winter school lectures took place in auditorium 81 of the Technical University of Denmark, and (ii) that "only" 50, out of a total of more than 130 participants ('students', workshopppers and lecturers) went to a mid-course Royal Danish Ballet evening which, in addition to classical, retrospective, Danish Bournonville ballet, also featured the more abstract 'Serenade' ballet by Balanchine.

CLOSING REMARKS

The 1979 Copenhagen Winter School on Abstract Software Specifications had a final panel session of some 60 minutes duration. The editor regrets being unable, at this time, to include an edited transcript of that most clarifying and concluding event. He does hope, however, some day, to be able to furnish such a written record; and invites readers to inquire.

At the banquet, where Professors Dahl and Naur entertained the more than 110 diners with several most enjoyable pieces of Bach (Naur: flute, Dahl: a somewhat out of tune piano), participants had contributed to a Winter School Song book. The editor also regrets to be likewise unable to provide this as an appendix to this volume. Since he particularly enjoyed the personal creations of Richard L. Wexelblat (of UNIVAC, Penn., USA), readers might likewise persuade the editor to provide a copy of that songbook.



Dines Bjørner

Holte, Easter 1980

-
- Vol. 49: Interactive Systems. Proceedings 1976. Edited by A. Blaser and C. Hackl. VI, 380 pages. 1976.
- Vol. 50: A. C. Hartmann, A Concurrent Pascal Compiler for Mini-computers. VI, 119 pages. 1977.
- Vol. 51: B. S. Garbow, Matrix Eigensystem Routines - Eispack Guide Extension. VIII, 343 pages. 1977.
- Vol. 52: Automata, Languages and Programming. Fourth Colloquium, University of Turku, July 1977. Edited by A. Salomaa and M. Steinby. X, 569 pages. 1977.
- Vol. 53: Mathematical Foundations of Computer Science. Proceedings 1977. Edited by J. Gruska. XII, 608 pages. 1977.
- Vol. 54: Design and Implementation of Programming Languages. Proceedings 1976. Edited by J. H. Williams and D. A. Fisher. X, 496 pages. 1977.
- Vol. 55: A. Gerbier, Mes premières constructions de programmes. XII, 256 pages. 1977.
- Vol. 56: Fundamentals of Computation Theory. Proceedings 1977. Edited by M. Karpiński. XII, 542 pages. 1977.
- Vol. 57: Portability of Numerical Software. Proceedings 1976. Edited by W. Cowell. VIII, 539 pages. 1977.
- Vol. 58: M. J. O'Donnell, Computing in Systems Described by Equations. XIV, 111 pages. 1977.
- Vol. 59: E. Hill, Jr., A Comparative Study of Very Large Data Bases. X, 140 pages. 1978.
- Vol. 60: Operating Systems, An Advanced Course. Edited by R. Bayer, R. M. Graham, and G. Seegmüller. X, 593 pages. 1978.
- Vol. 61: The Vienna Development Method: The Meta-Language. Edited by D. Bjerner and C. B. Jones. XVIII, 382 pages. 1978.
- Vol. 62: Automata, Languages and Programming. Proceedings 1978. Edited by G. Ausiello and C. Böhm. VIII, 508 pages. 1978.
- Vol. 63: Natural Language Communication with Computers. Edited by Leonard Bolc. VI, 292 pages. 1978.
- Vol. 64: Mathematical Foundations of Computer Science. Proceedings 1978. Edited by J. Winkowski. X, 551 pages. 1978.
- Vol. 65: Information Systems Methodology. Proceedings, 1978. Edited by G. Bracchi and P. C. Lockemann. XII, 696 pages. 1978.
- Vol. 66: N. D. Jones and S. S. Muchnick, TEMPO: A Unified Treatment of Binding Time and Parameter Passing Concepts in Programming Languages. IX, 118 pages. 1978.
- Vol. 67: Theoretical Computer Science, 4th GI Conference, Aachen, March 1979. Edited by K. Weihrauch. VII, 324 pages. 1979.
- Vol. 68: D. Harel, First-Order Dynamic Logic. X, 133 pages. 1979.
- Vol. 69: Program Construction. International Summer School. Edited by F. L. Bauer and M. Broy. VII, 651 pages. 1979.
- Vol. 70: Semantics of Concurrent Computation. Proceedings 1979. Edited by G. Kahn. VI, 368 pages. 1979.
- Vol. 71: Automata, Languages and Programming. Proceedings 1979. Edited by H. A. Maurer. IX, 684 pages. 1979.
- Vol. 72: Symbolic and Algebraic Computation. Proceedings 1979. Edited by E. W. Ng. XV, 557 pages. 1979.
- Vol. 73: Graph-Grammars and Their Application to Computer Science and Biology. Proceedings 1978. Edited by V. Claus, H. Ehrig and G. Rozenberg. VII, 477 pages. 1979.
- Vol. 74: Mathematical Foundations of Computer Science. Proceedings 1979. Edited by J. Bečvář. IX, 580 pages. 1979.
- Vol. 75: Mathematical Studies of Information Processing. Proceedings 1978. Edited by E. K. Blum, M. Paul and S. Takasu. VIII, 629 pages. 1979.
- Vol. 76: Codes for Boundary-Value Problems in Ordinary Differential Equations. Proceedings 1978. Edited by B. Childs et al. VIII, 388 pages. 1979.
- Vol. 77: G. V. Bochmann, Architecture of Distributed Computer Systems. VIII, 238 pages. 1979.
- Vol. 78: M. Gordon, R. Milner and C. Wadsworth, Edinburgh LCF. VIII, 159 pages. 1979.
- Vol. 79: Language Design and Programming Methodology. Proceedings, 1979. Edited by J. Tobias. IX, 255 pages. 1980.
- Vol. 80: Pictorial Information Systems. Edited by S. K. Chang and K. S. Fu. IX, 445 pages. 1980.
- Vol. 81: Data Base Techniques for Pictorial Applications. Proceedings, 1979. Edited by A. Blaser. XI, 599 pages. 1980.
- Vol. 82: J. G. Sanderson, A Relational Theory of Computing. VI, 147 pages. 1980.
- Vol. 83: International Symposium Programming. Proceedings, 1980. Edited by B. Robinet. VII, 341 pages. 1980.
- Vol. 84: Net Theory and Applications. Proceedings, 1979. Edited by W. Brauer. XIII, 537 Seiten. 1980.
- Vol. 85: Automata, Languages and Programming. Proceedings, 1980. Edited by J. de Bakker and J. van Leeuwen. VIII, 671 pages. 1980.
- Vol. 86: Abstract Software Specifications. Proceedings, 1979. Edited by D. Bjerner. XIII, 567 pages. 1980
-

CONTENTS

- . Dedication
- . Acknowledgement
- . Foreword
- . Contents

PRELUDE:

- 1. H.Zemanek 1
Abstract Architecture

CONSTRUCTIVE DEFINITIONS

- 2. J.Stoy 43
Foundations of Denotational Semantics
- 3. C.B.Jones 100
Models of Programming Language Concepts
- 4. D.Bjørner 144
Formalization of Data Base Models
- 5. D.Bjørner 216
... Block-structured GOTO Modelling:exits vs. Continuations

ALGEBRAIC SEMANTICS

- 6. S.N.Zilles 248
Introduction to Data Algebra
- 7. O.-J.Dahl 273
Time Sequences as a Tool for Describing Program Behaviour
- 8. R.M. Burstall & J.A. Goguen 292
The Semantics of CLEAR, A Specification Language
- 9. B.Dömölki 333
An Example of Hierarchical Program Specification

PROGRAM SPECIFICATIONS

- 10. B.Liskov 354
Modular Program Construction Using Abstractions
- 11. P.Lucas 390
On the Structure of Application Programs

AN INTERLUDE

- 12. H.Gerstmann & A.Ollongren 439
Abstract Objects as Abstract Data Types

PARALLELISM & NON-DETERMINISM

- 13. P.E. Lauer, M.W. Shields & E. Best 451
Design & Analysis of Highly Parallel & Distributed Systems
- 14. D. Park 504
On the Semantics of Fair Parallelism
- 15. G.D. Plotkin 527
Dijkstras Predicate Transformers & Smyth's Power Domains

POSTLUDE

- 16. H.Zemanek 554
Banquet Talk

- LIST OF PARTICIPANTS 564

ABSTRACT ARCHITECTURE

General Concepts for Systems Design

by

Heinz Zemanek

IBM Fellow

Professor at the University of Technology, Vienna

Paper for the Winterschool on Abstract Software Specification

at the

Danish University of Technology

Copenhagen, 2 February 1979

CONTENTS

	Introduction	2
1.	The Origin of Computer Architecture	4
2.	The Nature of Design	6
3.	The System	10
4.	The Transition from Informal to Formal	14
5.	System Information	17
6.	Four Phases of Technical Design	20
7.	Three Levels and Three Objects of Design	23
8.	Five Types of Information Treatment	28
9.	Some General Concepts	32
10.	Errors, Failures and Perturbations	37
	Conclusion	40

INTRODUCTION

This is not a paper like the other papers of this winterschool. Its intent is neither to teach a formal structure nor to give any lemmas or proofs. Its aim is to make you, after all you have heard during these two weeks, think of the purpose of abstract specification, to reconsider, to contemplate the wonderful tools you have been confronted with - and their use in the world. Which is a world of users, a world of people who are very far from the abstraction we have cultivated.

Our computer is a great thing, as a device and as a mental concept. It is so incredibly flexible, it can be made to do everything we want. But actually nothing in this technology is there for itself. It all has a practical purpose, it all is here for service. It seems appropriate to call this device computer. Because it shares with mathematics the property of being at the same time the queen of science and technology and the most humble servant.

We have developed our creation for more than 25 years - and what a world of possibilities, of mechanisms - concrete and abstract - and of applications has been added! We could be very proud of our achievements, did not there arise the disquieting question: do we indeed master what we have got? And this question has very many meanings - of which I will select only one: do we master the design of our structures, hardware, software and applications? And I will ask it in a slightly different form: what makes a design a good design?

The answer cannot be an algorithm and there is no intention to develop a measure for quality or beauty. Design, in contrast to mathematical theories and defined measurement, happens in a world of unremovable contradictions. The engineer is supposed to make use of applicable theories and available measurements - but his strength is where the parts covered by theories and measurements have their rough edges against each other, where satisfying the one means offending the other. In this situation, the engineer applies his ability to find the compromise, to bring an entity to work in spite of

unresolved theoretical contradictions, to get the thing produced at acceptable costs and delivered at the promised date. If I have here properly described the task of an engineer: is it not completely clear that software design is an engineering activity with mathematics and measurement as auxiliary tools? But then it is evident that there has not been done enough to cultivate the engineering character of software technology.

The keyword which has triggered my thoughts and my research work is *computer architecture*, a term which is now used very frequently. The goal of my research and the intention of this paper are to clarify the meaning of this term.

1. THE ORIGIN OF COMPUTER ARCHITECTURE

Both the term *computer architecture* and the idea of architectural design were used, as far as I have found out, for the first time in 1962 by Fred P. Brooks Jr. for his contribution to the book describing the development of the IBM-Computer STRETCH, which contribution has the title *Architectural Philosophy* [1]. This paper contains a definition which should have been generally accepted:

Computer architecture, like any other architecture, is the art of determining the needs of the user of a structure and then designing is to meet those needs as effectively as possible within the economic and technological constraints.

The spirit of this paper, the whole book and the development of STRETCH were leading up to a revolution in computer design - to the development of the IBM System/360. Its three architects, Fred Brooks, Gerrit Blaauw and Gene Amdahl, not only for the first time conceived a full spectrum of computers - from 360/20 to 360/95 - but, moreover, the spectrum was a family of models derived from a common concept, so that the design achieved what architecture should achieve: a style.

In their description of the architecture, however, the definition of architecture is worded already a bit differently [2]:

The term 'architecture' is used here to describe the attributes of a system as seen by the programmer, i.e. the conceptual structure and functional behaviour, as distinct from the organization of the data flow and controls, the logical design, and the physical implementation.

This sounds as if the authors had anticipated what would happen and tried to inhibit it: the term *architecture* has, since then, lost its precise and obliging meaning, and today it is almost as broad in its application as the term *structure*. Authors do use the term *architecture* when they mean