Lecture Notes in Computer Science 544
Edited by G. Goos and J. Hartmanis

M. Broy M. Wirsing (Eds.)

Methods of Programming

Selected Papers on the CIP-Project

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

Volume Editors

Manfred Broy
Institut für Informatik, Technische Universität München
Postfach 20 24 20, W-8000 München 2, FRG

Martin Wirsing
Universität Passau, Fakultät für Mathematik und Informatik
Postfach 25 40, Innstraße 33, W-8390 Passau, FRG

CR Subject Classification (1991): D.1-2

ISBN 3-540-54576-X Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-54576-X Springer-Verlag New York Berlin Heidelberg

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

Dedicated to Prof. Dr. Dr. h.c. mult. F.L. Bauer

Preface

The systematic development of software systems is a central task of computing science. A software system is the result of putting together knowledge about the application, the requirements and the structures of computing science. The latter aspect comprises knowledge about data structures, algorithms and notions of efficiency of target computing systems. Moreover, partially conflicting goals of optimization of programming systems such as simplicity of implementation versus user friendliness or run-time efficiency versus space efficiency have to be kept in mind by a software engineer during the development of a program. This leads into intricate problems. Help comes from an adequate structuring of the program design process. A systematic procedure and the division of the programming task into a number of goal-directed substeps contribute to make this process manageable. If the design process is to be supported by programming systems itself, it is necessary to formalize this particular area of application, namely the area of programming.

Under the heading CIP (Computer-aided Intuition-guided Programming), a group of researchers led by Prof. F. L. Bauer and the late Prof. K. Samelson started work in the year 1975 in the direction of formal program specification, transformational programming and tool support for program development. Initially, the transformation of recursion to iteration was one of their major interests, but soon it became apparent that the problem of structured specification and the transition from purely descriptive specifications to algorithms are major milestones in programming. A rule-based approach to the design of programs was provided by algebraic specification techniques for data structures. Consequently, theory, methodology, language and applications as well as aspects of compilers and interpreters were investigated within the project under the leadership of F. L. Bauer and K. Samelson. The CIP group developed an integrated approach to the design of programming systems. The concept of a wide spectrum language was just one aspect among many others.

It was central to these investigations to look at questions of methodology. In their book, F.L. Bauer and H. Wössner[1] described a formal approach to programming language concepts and program development based on algebraic specifications and program transformations. The collection of papers in this volume presents not only further examples of this approach but also evolutions and modifications of the original ideas of the CIP project. The topics range from descriptions of the program development process to derivations of algorithms from specifications.

The first part of the volume considers aspects of the development process and reusability. It begins with two papers on the modelling of the programming process. W. Hesse presents a metamodel for the management of the software development process which is based on his experience in office automation systems. B. Krieg-Brückner describes how in the ESPRIT project PROSPECTRA algebraic specifications and transformational program developments can be used for metaprogram development in order to formalize the program development process itself. Main aspects are higher-order functions and homomorphic extension functionals which act as development strategy and as induction schema for proofs. The other two papers of this chapter deal with aspects of reusability. H. Partsch and N. Völker are concerned with the reusability of transformational developments. They study the role of reuse for deriving different algorithms from the same specification based on the derivation of one particular algorithm. The suggested strategy is exemplified by the development of two pattern-matching algorithms. The paper of M. Wirsing and R. Hennicker concentrates on specifications rather than on transformations and is based on the notion of reusable components. A method for the systematic development and reuse of specifications is presented. It is illustrated by the construction of a reusable component for a simple word processing system.

The second part of the volume contains three papers on deductive program development. M. Broy presents a style of program construction which consists of alternating successive steps of specification, derivation and verification within a purely axiomatic framework. The technique is demonstrated by the development of an algorithm for the evaluation of expressions in reverse Polish notation. P. Pepper proposes "Literate Program Derivation": he combines the ideas of Don Knuth's literate programming with the concepts of deductive program development and shows his ideas using the string searching algorithm of Boyer and Moore as an example. Finally, R. Steinbrüggen demonstrates by the example of the line-drawing problem how existential propositions can be transformed into pre-algorithmic descriptions using Skolem's idea of quantifier elimination.

---

[1]F.L. Bauer, H. Wössner: Algorithmic language and program development. Berlin: Springer 1982

In the third part, case studies are presented which emphasize the use of algebraic and functional methods for program development. In the paper from U. Berger, W. Meixner and B. Möller, a formal derivation of a garbage collection algorithm is given which starts from an abstract specification at the level of graphs and derives an algorithm to a level which can be transcribed directly into machine code. The basic idea of the derivation is an "algebra of finite maps" which allows one to describe and transform pointer structures in a general way. R. Berghammer and H. Ehler prove and use elements of functional programming such as currying, tupling and fusion in the context of transformational program development. As their main example they study an algorithm for detecting cycles in directed graphs. The case study by C. Delgado Kloos and W. Dosch aims at showing how hardware descriptions can be derived from a specification within the framework of transformational programming. In their paper they develop circuit descriptions for a family of binary adders from a common specification of the functional behaviour.

The work of the CIP group relies on ideas and work of F. L. Bauer and K. Samelson. Their continual advice and constructive criticism and their insistence on clear concepts and well-structured solutions were of fundamental importance for this research. F. L. Bauer has not only given his research group suggestions and perspectives but also created a fruitful climate for scientific team work which, in spite of clearly formulated goals, always provided room for new ideas and theoretical studies. The ten contributions in this volume are based on formal work directed by F. L. Bauer and K. Samelson. We, the former members of the CIP-project, would like to express with this contribution our gratitude and thanks to our teachers and project leaders F. L. Bauer and K. Samelson.

Finally as the editors of this volume we would like to thank Ruth Breu for her help in preparing the volume, B. Möller for preparing the CIP bibliography and Springer-Verlag for their excellent cooperation concerning the publication of this volume.

Munich and Passau, July 1991 Manfred Broy and Martin Wirsing

Table of Contents

Part I. Development Models and Reusability

Part II. Deductive Program Development

Part III. Case Studies in Development