FORMAL SPECIFICATION

and

SOFTWARE DEVELOPMENT

DINES BJØRNER
Technical University of Denmark
and Danish Datamatics Centre

and

CLIFF B. JONES
The University, Manchester, England

in collaboration with:

Derek Andrews
Elizabeth Fielding
Wolfgang Henhapl
Peter Lucas
Hans Henrik Lövengreen
and
Joseph E. Stoy

ENGLEWOOD CLIFFS, NEW JERSEY LONDON NEW DELHI
SINGAPORE SYDNEY TOKYO TORONTO WELLINGTON

Library of Congress Cataloging in Publication Data
Bjørner, D. (Dines, 1937—
Formal specification and software development.

Bibliography: p.
Includes index.
1. Electronic digital computers—Programming.
2. Programming languages (Electronic computers) I. Jones,
C. B. (Cliff B.), 1944— II. Title.
QA76.6.B575 1982 001.64'2 82-7656
ISBN 0-13-329003-4 AACR2

British Library Cataloguing in Publication Data

Bjørner, D.
Formal specification and software development.
1. Computer programs 2. Software compatibility
I. Title II. Jones, C.
001.64'25 QA76.6

ISBN 0-13-329003-4

ISBN 0-13-329003-4

PRENTICE-HALL INTERNATIONAL, INC., *London*
PRENTICE-HALL OF AUSTRALIA PTY. LTD., *Sydney*
PRENTICE-HALL CANADA, INC., *Toronto*
PRENTICE-HALL OF INDIA PRIVATE LIMITED, *New Delhi*
PRENTICE-HALL OF JAPAN, INC., *Tokyo*
PRENTICE-HALL OF SOUTEAST ASIA PTE., LTD., *Singapore*
PRENTICE-HALL, INC., *Englewood Cliffs, New Jersey*
WHITEHALL BOOKS LIMITED, *Wellington, New Zealand*

1 0 9 8 7 6 5 4 3 2 1

Printed in the United States of America

FORMAL SPECIFICATION

and

SOFTWARE DEVELOPMENT

Prentice-Hall International
Series in Computer Science

C.A. R. Hoare, Series Editor

Published

BACKHOUSE, R.C., *Syntax of Programming Languages: Theory and Practice*
de BAKKER, J.W., *Mathematical Theory of Program Correctness*
BJØRNER, D. and JONES, C.B., *Formal Specification and Software Development*
DROMEY, R.G., *How to Solve it by Computer*
DUNCAN, F., *Microprocessor Programming and Software Development*
GOLDSCHLAGER, L and LISTER, A., *Computer Science: A Modern Introduction*
HENDERSON, P., *Functional Programming: Application and Implementation*
JONES C.B., *Software Development: A Rigorous Approach*
REYNOLDS, J.C., *The Craft of Programming*
TENNENT, R.D., *Principles of Programming Languages*
WELSH, J. and ELDER, J., *Introduction to Pascal,* 2nd Edition
WELSH, J. and McKEAG, M., *Structured System Programming*

To

TONY HOARE

on the occasion of his being elected

a Fellow of the Royal Society

CONTENTS

PREFACE

People today use an enormous number of 'systems' ranging in complexity from washing machines to international airline reservation systems. Computers are used in nearly all such systems: accuracy and security are becoming increasingly essential. The design of such computer systems should employ development methods as systematic as those used in other engineering disciplines. A systematic development method must provide a way of writing specifications which is both precise and concise; it must also include a way of relating design to specification.

A concise specification can be achieved by restricting attention to what a system is to do: all consideration of implementation details is postponed. With computer systems this is done by: a) building an abstract model of the system — operations being specified by pre- and post-conditions; b) defining languages by mapping program texts onto some collection of objects whose meaning is understood; c) defining complex data objects in terms of abstractions known from mathematics. Ths last topic, the use of abstract data types, pervades all work on specifications and is necessary in order to apply the ideas to systems of significant complexity. The use of mathematically based notation is the best way to achieve precision.

A design generates a number of sub-components and a way of combining them. These sub-components must be specified. Ultimately sub-components satisfying these separate specifications will be combined to form a system which should satisfy the overall specification. If all the specifications are precise enough, it is possible to prove that a design step is correct: that is, it fulfils the original specification. This is done before the sub-components have been implemented. Such proofs of correctness are of particular importance in the early stages of a design because any mistakes made then are likely to be particularly expensive to detect and correct later.

The early stages of design frequently involve choosing machine representations for abstract data objects. For this reason, special emphasis is given to proofs of data refinement (also called object transformation).

The lowest level of design for computer systems is often called 'coding', this is distinguished from the earlier design stages only by the fact that the sub-components required are all available in the language or support software being used, and techniques are available to prove that the code meets the module specifications. This description is somewhat oversimplified: design is by no means a strictly 'top-down' activity, but in order to be understandable, the eventual documentation must be presented in a top-down structure.

The work on specifications of large systems was at the outset prompted by the need for formal definitions of programming languages. John McCarthy argued for the provision of such definitions (a more complete historical background, with references, is given in chapter 1 of this book). The size of the PL/I language prompted the attempt to apply to it some of the ideas on formal

language definition, and in the mid 1960's a definition of the PL/I language was developed in the IBM Laboratory at Vienna. This definition used 'operational semantics' and the overall approach became known as the 'Vienna Definition Language (VDL).

Christopher Strachey's group in Oxford University developed the concept of 'denotational' or 'mathematical semantics'. In the early 1970's, prompted by Hans Bekic, this new approach was adopted by the Vienna group: the more recent work of the group is thus based on the denotational approach. Some confusion has, perhaps, been caused by the decision to refer to the new work as the '*Vienna Development Method*' (whose initials, *VDM*, are too like those of the other work). The meta-language used in VDM was known internally as '*META-IV*': it is denotational in approach. VDM is, however, more than just a formal definition language: as the name implies it is a complete systematic development method. The other part of the background to the VDM work is provided by the work on program development methods of people such as Bob Floyd, Peter Naur, Tony Hoare, Robin Milner, Niklaus Wirth and Edsger Dijkstra. In particular the idea of 'data refinement' (or object transformation) is a key component of VDM.

There is emerging an increasing acceptance of the need for formal specification and design techniques. VDM is a systematic development method which has a wide variety of applications; it has been, and is being used in major companies and courses on it have been given throughout Europe. Even within the confines of this book, actual programming languages and database systems are discussed.

The aim of this book is to provide a source document for both industrial application of, and for post-graduate courses on, VDM. The only knowledge assumed of the reader is that of set and logic notation. The book is divided into three major parts. General ideas, and in particular the meta-language, are covered in the three chapters of part I. Part II is concerned with use of the VDM on programming languages; other applications are considered in part III. (Recent work on parallelism is mentioned only via references in this book.)

The parts and chapters are connected by 'link material' which provides the context for the individual contributions, gives further reference and provides hints on alternative ways to read the book.

We should like to acknowledge help from the following people. Gordon Plotkin reviewed two versions of the whole book; Tony Addyman, Stephen Bear, Ian Cottam, Chris Kirkham and Ann Welsh each reviewed various chapters. The typists, Annie Rasmussen and Birte Skovlund did a superb job in entering extremely difficult manuscripts. Finn Hansen prepared the artwork. The Danish Datamatics Centre provided the word processing system. Jørgen Fischer Nilsson, Bo Stig Hansen, Jan Storbank Pedersen, and Lennart Schultz worked on early versions of chapters 12 and 13. We should also like to thank the contributors without whom this volume would have been impossible. Finally Ron Decent and many others at Prentice-Hall International were most helpful in the final stages of the preparation of this book.

D.B.

C.B.J.