PROGRAMMING
for
SOFTWARE
SHARING

ON INFORMATION SCIENCE

A series devoted to the publication of courses and educational seminars given at the Joint Research Centre, Ispra Establishment, as part of its education and training program. Published for the Commission of the European Communities, Directorate-General Information Market and Innovation.

Additional volumes in preparation.

The publisher will accept continuation orders for this series which may be cancelled at any time and which provide for automatic billing and shipping of each title in the series upon publication. Please write for details.

PROGRAMMING for SOFTWARE SHARING

Edited by

D.T. Muxworthy

Program Library Unit, Edinburgh, U.K.

D. REIDEL PUBLISHING COMPANY

A MEMBER OF THE KLUWER

ACADEMIC PUBLISHERS GROUP

DORDRECHT / BOSTON / LANCASTER

Library of Congress Cataloging in Publication Data

Main entry under title:

Programming for software sharing.

CIP

(Ispra courses on information science)

Includes index.

1. Software compatibility. 2. Electronic digital computers—
Programming. I. Muxworthy, D. T. II. Series.

QA76.6.P7514 1983 001.64'2 83-8685

ISBN-13: 978-94-009-7147-9

e-ISBN-13: 978-94-009-7145-5

DOI: 10.1007/978-94-009-7145-5

Commission of the European Communities

Joint Research Centre Ispra (Varese), Italy

Publication arrangements by

Commission of the European Communities

Directorate-General Information Market and Innovation, Luxembourg

EUR 8516 EN

Copyright © 1983, ECSC, EEC, EAEC, Brussels and Luxembourg

Softcover reprint of the hardcover 1st edition 1983

LEGAL NOTICE

Neither the Commission of the European Communities nor any person acting on behalf of the Commission is responsible for the use which might be made of the following information.

Published by D. Reidel Publishing Company

P.O. Box 17, 3300 AA Dordrecht, Holland

Sold and distributed in the U.S.A. and Canada

by Kluwer Boston Inc.,

190 Old Derby Street, Hingham, MA 02043, U.S.A.

In all other countries, sold and distributed

by Kluwer Academic Publishers Group,

P.O. Box 322, 3300 AH Dordrecht, Holland

All Rights Reserved

No part of the material protected by this copyright notice may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying, recording or by any informational storage and retrieval system, without written permission from the copyright owner.

CONTENTS

Preface / H. J. Helms	vii
Introduction / D. Muxworthy	ix
Software Development Methodology	
The Software Life Cycle / G. Goos	3
Design of Programs / G. Goos	15
Program Structure / G. Goos	35
Programming Languages and Style / P. Poole	49
Program Validation / P. Poole	69
A Programming Example / G. Goos	87
Managing Software Development / M. Dowell	97
Flexibility	
Flexibility / W. Waite	111
Operating System Interfaces / W. Waite	127
Language Standards / D. Muxworthy	139
Software Tools for Software Sharing / D. Muxworthy	149
Program Documentation / G. Gaggero	175
Transfer and Sharing	
Software Transfer and Sharing – An Overview / B. Ford	191
The Legal Protection of Computer Software / B. Harris	203
Experience in Multi-Machine Software Development: the NAG Library – A Case Study / B. Ford	211
Transportable User Documentation for Numerical Software / B. Ford	225
Data Sharing in Distributed Computing Systems / F. Schreiber	235
FORTTRAN IV Dialect Conversion – A Case Study: CDC to IBM / A. Policini	247
Users’ Experience: Introduction / M. Dowell	263
Experience with the RELAP Code / W. Kolar, S. Duffield, R. Nijsing and G. de Vries	265
Experience with the SIMMER II Code / A. Jones	275
Subject Index	279

PREFACE

Most computer users are familiar with the problems of sharing software with others, and the transfer of programs from one computing environment to another. Software represents an ever-increasing proportion of the cost of computing and these costs tend to nullify all the economic advantages flowing from the wider availability of cheap hardware. Years ago it was hoped that the widespread use of high-level programming languages would help in alleviating the problems of software production, by increasing productivity and by making it simpler for users with similar problems to be able to use the same programs, possibly on different types of machines. It is a common experience that in practice this simple optimism has proved to be unfounded.

It was these considerations which led us in 1979 to organize a two-week course on "Programming for Software Sharing" at the European Community Joint Research Centre, Ispra Establishment (Italy), forming part of the regular series of "Ispra Courses". With prominent invited lecturers, local contributions and through discussion sessions we examined with an audience from many countries the problems involved in the sharing and transfer of software, as well as suggesting ways of overcoming them.

In our local environment we are faced daily with three problems both from engagements in software exchange in the scientific-technical field on a Europe-wide or world-wide basis, and from work with programming techniques and contributions to the international standardization process.

The present volume is however not the proceedings of this course held some years. It is a monograph inspired by that activity. Some contributions are indeed updated versions of material presented during the course, but others are recent papers prepared for the book.

We hope the contributions selected for the book will provide readers with some more insight and understanding of the problems and techniques involved in software sharing. The material should be of value for a wide range of computer users as well as for program developers and implementors.

In presenting this volume our thanks go to lecturers and participants of the Ispra Course, the authors of the contributions to the book and above all to Mr. D. Muxworthy, University of Edinburgh (UK) who undertook the task as editor, and who was instrumental throughout all the plans leading to this monograph.

Many people from the Ispra Establishment have contributed to this publication, which also was inspired by Mr. B. Henry, head of the Ispra Courses. The work and devotion of Messrs. M.D. Dowell and A.A. Pollicini of our Informatics Division is gratefully acknowledged. We also thank Mr. P. De Hoe and his staff from Publications Service of the JCR-Ispra.

Hans Jørgen Helms
Director, Informatics, Mathematics
and Systems Analysis Department

July 1982

INTRODUCTION

David Muxworthy
University of Edinburgh
Edinburgh, UK

A common theme in this volume is that software costs continue to rise whilst hardware costs fall. For many computer installations the true costs of software exceed the true costs of hardware yet while hardware production is based on sound engineering practices, developed over centuries, software production is only very slowly being formalized. The haphazard methods of construction of entities which are to be used in important areas of human activity, from defence systems and macroeconomic systems to areas of more specific potential danger such as the design of motor cars, bridges and chemical plant, would not be tolerated in any other field.

Slowly the methods of producing more reliable software are becoming more widely known. As well as being reliable, good software should be flexible, not only because the problem to be solved commonly changes with time, but also because the facilities available to the software user also change with time and may thus involve several different types of computing systems. Additionally because software production is expensive it is sensible to attempt to achieve economies of scale and to share software with other users with the same or similar problems. Sharing software also implies sharing techniques and a program can contain the distillation of a wealth of experience in a particular problem area. As the users' experience contributors to this volume point out, lack of resources too often means that in practice the choice is between acquiring a shared program and having no program at all.

This book addresses many aspects of sharing software, together with the inextricably linked attributes of good design, reliability and flexibility. Most examples are taken from science and technology but the principles apply to any computer application. More than one author makes comparisons with engineering construction and there are indeed some striking parallels with an industry which developed in the first industrial revolution. It is possible for a railway freight wagon to run throughout much of Europe, or much of North America, and the wheel gauge, loading gauge, couplings and brakes are so standardized that the wagon may be coupled to any appropriate train of almost any railway company. Those who

perceived this requirement from the beginning of railways prevailed over those who advocated locally optimal but non-uniform systems.

When the Stockton and Darlington railway was being built in the early 1820's it was not clear whether the primary haulage would be by horse or by locomotive and the track was designed to accommodate both. Later in the decade the promoters of the Liverpool and Manchester railway organized a public competition for locomotives and one, the Rocket, emerged from its Ordeal, a much more expressive word than benchmark, as the winner on each of the counts of power, reliability and efficiency. The building of the London and Birmingham railway in the 1830's is particularly reminiscent of a large modern software project. The work was so big (112 miles of line plus ancillary buildings) that numerous subcontractors were employed. As unexpected difficulties arose the subcontractors fell behind schedule and finally abandoned work as the money ran out. More money had to be raised and costs rose even higher as extra effort was made to catch up on schedule. One particularly intractable problem, a long tunnel through a hill full of quicksand, required either a complete rerouting or, as happened, the hill to be pumped dry. This alone took 19 months and much extra expense. The whole project was completed many months late and costs were well over twice the original estimates. Plus ça change, plus c'est la même chose.

Thus software writers should not assume that theirs is a completely new field but should draw on what is relevant, project management, codes of practice, concepts of design and testing, use of tools and so on from other areas, particularly engineering. Analogies should not be taken too far however. Most users of a new car can drive away without reading any documentation and the user documentation for a bridge is usually non-existent. For software good documentation is vitally important, and is often lacking, and several authors address this point. Unlike physical products, software may be stolen, leaving the original intact and its owner unaware of the fact. The legal state of software affairs is complex and often indeterminate but considerable efforts are being made to improve matters. Problems of reliable distributed databases affect everyone with a bank account or a credit card and international standards affect all users of computers directly or indirectly. These and other related subjects are covered below.

As editor I have enjoyed reading all the contributions. To guide the reader they have been grouped into three broad categories, namely software development methodology, flexibility, and transfer and sharing. Some authors have made use of the terms "Ada", "UNIX" and "VAX" which are trademarks of the United States Department of Defense, Bell Laboratories and Digital Equipment Corporation respectively. The art of writing shareable software appears to lie in being able to think more abstractly, in so far as is feasible divorcing the design of a program from the language to be used and divorcing the implementation from the actual computer to be used. It might be useful to perceive the computer as malleable in the style of Salvador Dalí, rather than rigid in the style of Canaletto.

I hope readers find the book both interesting and useful.