The Software Factory

Managing Software Development and Maintenance

James R. Johnson

The Software Factory

Managing Software Development and Maintenance

James R. Johnson

QED Information Sciences, Inc.
Wellesley, Massachusetts

QED Plaza
P.O. Box 181
Wellesley, MA 02181

Library of Congress Catalog Number: 88-18580
International Standard Book Number: 0-89435-260-1

Printed in the United States of America
89 90 91 10 9 8 7 6 5 4 3 2 1

Library of Congress Cataloging-in-Publication Data

Johnson, James R.
The software factory.

1. Computer software—Development—Management.
I. Title.
QA76.76.D47J65 1988 005.1 88-18580
ISBN 0-89435-260-1

Foreword

Over the past 30 years a major new set of managerial challenges has been created by the rapid evolution and spread of information system technology. This evolution has provided a number of firms with the means to become more effective competitors by creatively using this technology. As a result, a significant focus on strategic and competitive uses of Information Technology (IT) dominates the literature. However, the effort required to actually implement these systems was vastly underestimated by many professionals.

The challenge of implementation has shown that many managers moved so stolidly toward engaging senior management and pursuing strategic/competitive use of technology, they moved away from some of the fundamentals of managing software development. Many IT managers face significant problems since their first hand technical experience was with technology so different from that of the 1980s. Further, the understanding of what makes acceptable management practice in the IT field has changed dramatically. Virtually all current conventional wisdom on management of this activity has been introduced in the last ten years. This places a special burden on IT management to not just address the day-to-day operating problems, but to also identify, assimilate

and implement very different methods and approaches. If managers are not committed to a process of self-revewal they will become obsolete.

This book concentrates on effective management of IT resources with heavy emphasis on practical insights for delivering and maintaining high quality software systems. It provides a unique opportunity to examine, in depth, contemporary management of an IT activity.

The Software Factory is for professionals written by an outstanding professional. Jim's years of experience clearly come through in the hard-hitting, insightful chapters. He has developed an integrated view of IT management from a perspective of the late 1980s. Given the depth of factual and practical information on the software development and maintenance activity, I think you will find this an unusually helpful book.

James I. Cash, Jr.
Professor of Business Administration
Harvard Graduate School of Business Administration

Preface

Having worked in a Software Factory for 16 years and having managed the area for seven, it is satisfying to document our software development approach. In 1980, QED published my previous book titled *Managing For Productivity In Data Processing*. When requested to update the material, I realized that so much had changed over the years, a revision was impractical.

Thus, a new book resulted with a focused objective, managing a "factory," a factory producing business application software. It is not an introductory text; the book is intended for professionals working in the MIS field who have practical experience in software development.

In February of 1988, I assumed responsibility for our Data Center operations. Finalizing the book after leaving the Software Factory was a nostalgic experience.

James R. Johnson
Director of Data Center
Hallmark Cards, Inc.

Acknowledgements

This book resulted from the efforts of eight Systems Development Managers at Hallmark Cards, Inc., who were dedicated to improving the software development process. In fact, the same team of managers were together for seven years. Thus, credit for the organization's productivity and the concepts documented in this book goes to the team: Lavon Faught, Bob Fisher, Lynne Haler, Roy Henrich, Harold Lundy, Jim McDonald, Ron Smith, and Pete Thorsell. Each manager was involved in the implementation of one or more of the productivity concepts in this organization.

A Software Factory doesn't function in a vacuum. Our vice president, John Collins, provided support and consistent direction over the years. Also, the supporting technical departments helped us realize productivity with subsecond response time and software tools.

I would like to thank my secretary, Charlene Gill, for tolerating my constant revisions. Her talent for reading, correcting, and typing the material was invaluable.

The artwork was created by Susan Johnson.

Introduction

PERSPECTIVE

Over the past ten years, our organization has pursued productivity techniques for the Systems Development function. This book highlights what was learned, from counting Lines of Code (LOC) to Business System Planning (BSP). Our Systems Development department is a Software Factory, over two hundred professionals generating between 40 and 60 products per year, each with the following average characteristics:

Man Days	400–450
LOC	35,000
Function Points	300–400

The emphasis, varying from year to year, depended on new technology or areas of opportunity. For example, consider the major issues explored in the past six years: 1988—Expert Systems; 1987—Code Generators; 1986—Performance Measurement and Third-Generation Language (3GL) Productivity; 1985—Technology Scan; 1984—Fourth-Generation Languages (4GL); 1983—Prototyping, Response Time Study; 1982—Function Points, Information Center.

Our conclusions have been validated by publishing articles in popular periodicals or speaking at MIS conferences. Comparing internal strategies to external sources, obtained passively through reading periodicals and books and attending conferences, or obtained actively by writing articles or presentations, assured a rational approach.

A FACTORY

Typically, Computer Operations is visualized as a factory, the computer processing thousands of jobs per day. However, the term *Software Factory*, in the context of this book, applies to the manufacture of programs, more specifically, programming system products or documented, integated business applications (as originally defined by Fred Brooks). Our term for this product is a *production system*, which requires "creator independence," allowing end-user modification for routine operation via tables or parameters.

Appendix A contains six years of statistics on the factory's output. The base information includes Lines of Code, programs, program bugs (Unusual Conditions Reports, UCRs), projects' schedule and budget performance, Function Points, and staffing levels.

IMPORTANCE

Recent renewed interest in the Software Factory is based on two factors—one positive, one negative. Using technology as a competitive weapon generally requires strategic on-line systems; thus, management turns to the factory for implementation. On the negative side, visibility of high maintenance costs has top management concerned.

Also, as new facets of MIS (electronic mail, local area networks, personal computers, departmental computers, teleconferencing, etc.) proliferate, the implementation of systems has become more complex based on additional integration. Thus, managing a corporation's Software Factory will always be a priority subject.

ENVIRONMENT

One not necessarily intuitive observation learned from interfacing with other companies is that organizational environment

and cultural differences are as important as technological considerations when selecting productivity techniques. The management priorities for a factory depend on both MIS and corporate characteristics, such factors as the stage of automation and end-user data processing activity.

Although software factories produce similar products, factory profiles vary considerably, directly reflecting the corporate environment/culture. The description of our organization in Appendix B provides the reader perspective and allows comparisons of organizational and environmental factors. The comments are not extensive; the intent is to convey a "flavor" of the environment.

ORGANIZATION

The book is organized in seven parts, each with a central theme. The organization is an arbitrary choice; Business System Planning (BSP) might have been first rather than last. My logic was to start with productivity tools, cover measurement, motivation, and project control second, ending with BSP. Although the book is focused, some topics have been slighted, such as recruiting and organization. These topics are important, but are unique to a corporate environment. Stating the questions answered in each part clarifies their content.

Part I The Weakest Link

Is subsecond response time cost-effective?
Does programmer productivity improve if response time is reduced from .5 seconds to .3, or from .3 to .2?
How should service-level agreements be monitored?

Part II Directional Tools

Why the paradox of different 4GL strategies?
Is CPU consumption the critical factor?
Does complexity influence language strategy?
Are code generators the future?
Will the factory adopt Expert Systems?

Part III Productivity Tools

Why does prototyping apply?
Does cloning code improve productivity?
Do bullpens improve productivity?
Will audits improve quality?
Can testing be planned?

Part IV Measurement

What are productivity measures?
Do the same measures apply, at different organizational levels, to the programmer analyst?
Is it possible to quantify total performance?
How much detail is enough?

Part V Motivation

Is burnout real?
Are performance reviews a positive experience?

Part VI Projects

What basic steps apply to all projects?
Does an advanced procedure exist?
Are function points superior to lines of code?

Part VII Business System Planning (BSP)

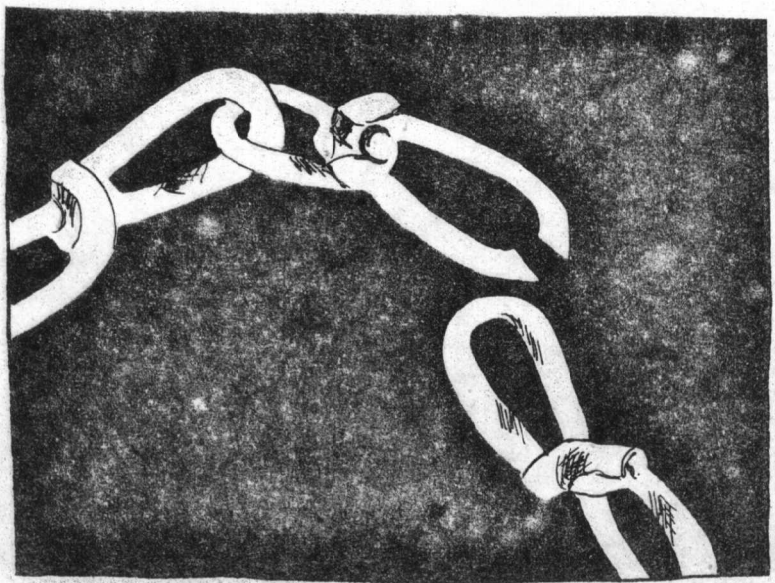
Are corporate priorities reflected in the MIS portfolio?
Is top management involved in MIS planning?
What are the alternative methodologies?

The writing approach varies considerably among chapters: Chapter 2 starts with a dialog, chapters 4, 5, and 10, a quiz, and the bulk of chapters 7 and 8 are examples. This variety should provide a change of pace while maintaining readability. Major references are included after selected chapters. Appendix C relates our response time data to the Thadhani curve as referenced in the first chapter.

Contents

PART I

The Weakest Link



Consistent subsecond response time and computer availability are critical to productivity. Our concentrated research into the subject started in 1983, when subsecond response time was first realized. After comprehending the productivity gains, I was amazed. Why didn't the industry believe the results? Publicity was minimal.

In reviewing the conclusions with Walter Doherty and Bucky Pope from IBM's Thomas Watson Research Center, I learned they had similar thoughts. Thus, the goal of Chapter 1 is to "prove" the value of response time. A simple technique for any organization to validate the impact of subsecond response time is provided. Appendix C, the Thadhani curve, supplements Chapter 1. It is recommended to those interested in more supporting detail on measuring sub-second response time.

Recording computer availability as viewed from the end user's terminal is a difficult task. Commonly quoted availability statistics seldom reflect the actual time a user can execute all transactions. A complete chain, composed of both hardware and software links (components), must exist for end-user availability. As hardware reliability improves, the weakest links become the software components. Topics included: calculating availability with independent components, external monitoring of availability, and alternative hardware/software options.