Value-Based Software Engineering

Stefan Biffl · Aybüke Aurum · Barry Boehm ·
Hakan Erdogmus · Paul Grünbacher (Eds.)

Value-Based Software Engineering

With 69 Figures and 41 Tables

Springer

Editors
Stefan Biffl
Institute for Software Technology
Vienna University of Technology
Karlsplatz 13
1040 Wien, Austria
stefan.biffl@tuwien.ac.at

Aybüke Aurum
School of Information Systems,
Technology and Management
University of New South Wales
Sydney, NSW 2052, Australia
aybuke@unsw.edu.au

Barry Boehm
Center for Software Engineering
University of Southern California
941 W 37th Place,
Los Angeles, CA 90089-0781, USA
boehm@sunset.usc.edu

Hakan Erdogmus
Software Engineering
NRC Institute for Information
Technology
National Research Council Canada
Building M50, 1200 Montreal Rd.
Ottawa, ON, Canada K1A 0R6
Hakan.Erdogmus@nrc-cnrc.gc.ca

Paul Grünbacher
Systems Engineering & Automation
Johannes Kepler University Linz
Altenbergerstr. 69
4040 Linz, Austria
paul.gruenbacher@jku.at

Library of Congress Control Number: 2005930639

ACM Computing Classification (1998): D.2.1, D.2.8, D.2.9, D.2.10, K.6.1, K.6.3

ISBN 978-3-540-25993-0 ISBN 978-3-540-29263-0 (eBook)
DOI 10.1007/978-3-540-29263-0

Springer is a part of Springer Science+Business Media
springeronline.com

Cover design: KünkelLopka, Heidelberg
Typesetting: Camera ready by the editors
Production: LE-TeX Jelonek, Schmidt & Vöckler GbR, Leipzig
Printed on acid-free paper 45/3142/YL - 5 4 3 2 1 0

Foreword

Ross Jeffery

When, as a result of pressure from the CEO, the Chief Information Officer poses the question "Just what is this information system worth to the organization?" the IT staff members are typically at a loss. "That's a difficult question," they might say; or "well it really depends" is another answer. Clearly, neither of these is very satisfactory and yet both are correct. The IT community has struggled with questions concerning the value of an organization's investment in software and hardware ever since it became a significant item in organizational budgets. And like all questions concerning value, the first step is the precise determination of the object being assessed and the second step is the identification of the entity to which the value is beneficial. In software engineering both of these can be difficult. The precise determination of the object can be complex. If it is an entire information system in an organizational context that is the object of interest, then boundary definition becomes an issue. Is the hardware and middleware to be included? Can the application exist without any other applications? If however the object of interest is, say, a software engineering activity such as testing within a particular project, then the boundary definition becomes a little easier. But the measure of benefit may become a little harder.

In this book the issues related to the value of different software engineering activities are addressed along with the benefits and opportunities in decision making under conditions of conflict of decision criteria in uncertain contexts.

Because software has many stakeholders including developers, users, and managers, it is essential that a comparative measure of the software be devised to support software decisions. This is the aim of value-based software engineering. If we can develop models and measures of value which are of use to the manager, the developer, and the user, then trade-off decisions can become possible, for example between quality and cost or between functionality and schedule. Without the comparative measures, the comparisons are impossible and the decisions regarding development alternatives can only address one criterion, such as defects or functionality, at any point in time, since we need to measure defects or functionality using the same yardstick. Value can be that yardstick.

If we were to divide the software engineering domain simplistically into the production of shrink-wrapped and other products, we could start to divide the problem. In the case of shrink-wrapped, the definition of the object of interest becomes quite clear. It is a product that is sold. The valuation of interventions in the software engineering activities in this domain appears easier than in many other domains. In this case the quality model work that has been carried out in software engineering can provide some insights into the relative value of product characteristics. It would then be possible to investigate the software engineering interventions that give rise to changes in the quality characteristics that are valued by the consumer of the software product. In this manner the link between software engi-

neering process interventions and product characteristics allows for a value-based measure for those interventions.

Another way of looking at value in this context might be the work that has been carried out on product performance. It has been shown in many countries, for example, that outstanding product success derives from product advantage (defined as superior price/performance, customer benefits, and relative product quality), pre-development assessments, cross-functional teams, focus on markets where influence exists, and other factors. Perhaps value-based software engineering needs to understand some of these factors and then link them to substantive software quality models if value-based decisions are to be made in the software engineering context.

But how might we assess interventions in software engineering? Since software engineering is a human-intensive activity that results in a logical product that is often used as a part of a business process, the determination of value can draw from many disciplines. Perhaps one issue of interest is the assessment of the value of training of software engineers. In this case the value of human resource intervention programs may be a part of the area of interest. Can we make use of work, as given by the Brogden utility equation, for measuring the change in utility in dollars after a training program when looking at the value of project training interventions in software engineering?

Another factor that seems clearly of concern in this area is the methods we use to value information when we are making decisions under conditions of uncertainty. Methods such as the use of the expected value of perfect information (EVPI) can set the upper value bound in these conditions. The minimum can also be determined using these techniques. In this way it might be possible to consider the payoff maximization for software engineering interventions as well as the minimization of regret or loss.

Clearly these are complex, multidisciplinary opportunities for the research community, with significant potential economic impact across economies. In this book the editors have collected the current state of the art in the application of value-based approaches to software engineering activities and decisions. The book sets a framework for value, the theoretical foundations, the practices, and the application. The authors are drawn largely from the software engineering research community that is involved in the areas of software engineering decision making, measurement, and investment. This book presents an exciting collection of chapters in an area of research that will develop over the ensuing years as the importance of this work gains recognition in the wider community.

Author Biography

Ross Jeffery is Professor of Software Engineering in the School of Computer Science and Engineering at UNSW and Program Leader in Empirical Software Engineering in National ICT Australia Ltd. (NICTA). Previously he was Director of the Centre for Advanced Software Engineering Research (CAESER) at the Uni-

versity of New South Wales. Professor Jeffery was the founding Head of the School of Information Systems at UNSW from 1989 to 1995 and Associate Dean (Technology) for the Faculty of Commerce and Economics from 1996 to 1999. He was the founding Chairman the Australian Software Metrics Association (ASMA) where he served as Chairman from its inception for a number of years. He is Chairman of the IEAust/ACS Joint Board on Software Engineering. He has served on the editorial board of the IEEE Transactions on Software Engineering, and the Wiley International Series in Information Systems and he is Associate Editor of the Journal of Empirical Software Engineering. He has also been on the steering committee of the IEEE and ACM International Conference on Software Engineering and served as Program Co-Chair for the 1995 conference in Seattle. He is a founding member of the International Software Engineering Research Network (ISERN). He was elected Fellow of the Australian Computer Society for his contribution to software engineering research. His current research interests are in software engineering process and product modeling and improvement, electronic process guides and software knowledge management, software quality, software metrics, software technical and management reviews, and software resource modeling and estimation. His research has involved over fifty government and industry organizations over a period of 15 years and has been funded by industry, government, and universities. He has co-authored four books and over one hundred and twenty research papers.

Preface

Stefan Biffl, Aybüke Aurum, Barry Boehm, Hakan Erdogmus, Paul Grünbacher

This book tackles software engineering decisions and their consequences from a value-based perspective. The chapters of the book exploit this perspective to foster

- better evaluation of software products, services, processes, and projects from an economic point of view;
- better identification of risks for software development projects and effective decision support for them in a multicriteria and uncertain environment;
- better project management through a better understanding of the contribution of the activities and practices involved, the techniques, artifacts, and methods used, as well as the functionality, products, and systems delivered.

What Do We Mean by "Value"?

The goal of software engineering is to create products, services, and processes that add value. People who contribute to the creation of these artifacts – analysts, process engineers, software engineers, testers, managers, executives – strive in their decisions and actions to maximize some simple or complex notion of value, whether consciously or unconsciously, and whether with respect to shared goals or to satisfy personal objectives. Alas, when value considerations remain implicit, the overall effect may very well be negative. Examples of undesirable consequences of implicit and clashing value perspectives abound. A good case in point is when developers value superior design, the marketing of new, nifty functionality, quality assurance "zero defects" and the management of short time-to-market. Another example is when product quality is pursued for quality's sake with little regard to shareholder value (Favaro, 1996). Yet another is when management tries to drive development costs down by treating developers as a replaceable commodity or by evaluating them using one-dimensional performance metrics, and the development team reacts by creating knowledge silos or by "coding to rule" to protect its own interests. If value perspectives are not explicated and reconciled, everybody loses in the end.

Value-based software engineering (VBSE) brings such value considerations to the foreground so that software engineering decisions at all levels can be optimized to meet or reconcile explicit objectives of the involved stakeholders, from marketing staff and business analysts to developers, architects, and quality experts, and from process and measurement experts to project managers and executives. In VBSE, decisions are not made in a setting blind to value perspectives, whether common or differing, of these project participants.

Driven by both individual and collective goals, these stakeholders all hope to derive some benefit, whether tangible or intangible, economic or social, monetary or utilitarian, or even aesthetic or ethical. By the term value, we refer to this ulti-

mate benefit, which is often in the eye of the beholder and admits multiple characterizations.

A *Dictionary of Canadian Economics* defines value as*: "The quantity of one product or service that will be given or accepted in exchange for another. It is therefore a measure of the economic significance of a particular good or service. This value in exchange depends on the scarcity of the good or service and the extent to which it is desired."*

While this certainly is a common definition of value and is addressed prominently in the book, it represents only one dimension. A *Modern Dictionary of Sociology* defines value more abstractly as a *"...generalized principle of behavior to which the members of a group feel a strong commitment and which provides a standard for judging specific acts and goals."*

In the same spirit, the *Oxford Companion to Law (1980)* points out that *"...value may consist of spiritual or aesthetic qualities, or in utility in use, or in the amount of money or other goods which could be obtained in exchange for the thing in question…"* although the latter, monetary sense, by virtue of being the most tangible, is the most relevant in legal contexts.

In this book, you will find many contributions that stress the more general, group-oriented, and utilitarian aspect of value alongside those that focus on the more traditional, economic and monetary aspect. Neither aspect takes precedence over the other; both aspects are relevant to tackling the wide spectrum of software engineering issues covered in this book.

A Historical Perspective

To our knowledge, the first significant text to address value considerations beyond cost models in the software development context was Boehm's Software Engineering Economics (Boehm, 1981). Boehm later focused on the relationship between value and software process. The result was the spiral model of software development, which brought to the foreground risk management as an integral component in software process (Boehm, 1986).

The value-based management movement of the early 1990s (McTaggart, 1994) inspired an *IEEE Software* essay entitled "When the Pursuit of Quality Destroys Value" (Favaro, 1996). This essay made the controversial argument that superior quality should not be a goal in itself in the absence of favorable economics. Favaro et al. used the adjective "value-based" in the software development context in a later article addressing the economics of software reuse (Favaro et al., 1998). The same year the Economics-Driven Software Engineering Research (EDSER) workshops debuted at the International Conference on Software Engineering (ICSE) as a forum to share experiences and promote VBSE-related issues among the research community. The EDSER workshops have since been collocated with this annual conference with increasing popularity, and continue to be an important source of information. Two years after EDSER's debut, Boehm and Sullivan proposed the first agenda for VBSE research at ICSE 2000.

Over time, the scope of VBSE research expanded to include aspects of value other than economic and monetary. Of particular historical interest is the WinWin model of requirements negotiation, introduced by Boehm and others in the mid-1990s“ (Boehm et al., 1998). The WinWin model stressed the multi-stakeholder perspective by incorporating into the spiral model an approach for reconciling differing value propositions of project stakeholders. During the late 1990s and early 2000s, the advent of empirical and evidence-based software engineering, value-based management approaches, preference-based decision making, as well agile software development and other risk-driven methods continued to push the VBSE agenda forward and enlarge its scope. In 2003, Boehm proposed a formal VBSE agenda that captures the expanding scope of this burgeoning field (Boehm, 2003). The book both revisits and builds on this agenda.

Why Should You Care About Value-Based Software Engineering?

It is impossible to effectively address value considerations when software development is treated as an ad hoc endeavor. Much like in conventional engineering, the incorporation of value considerations requires treating software development as a purposeful endeavor, which aims at the cost-effective and reliable construction and maintenance of products that meet specific, if not always static, goals. Hence the title of the book: *Value-Based Software Engineering.*

Software admittedly has unique internal and external characteristics, in particular its highly flexible and volatile nature and its heavy dependence on collaboration among creative and skilled people, that in many instances necessitate a construction and management approach radically different from that of building a bridge or a ship, and more akin to new product development. However, basic engineering principles of discipline, economy, rigor, quality, and utility, and, to a certain extent, repeatability and predictability, still very much apply. As in conventional engineering, value considerations affect the trade-offs among these principles, but probably with much more subtlety, severity, and variety than they do in the engineering of hard products.

But why are these trade-offs so important? For no other reason than that they ultimately determine the outcome of a software project. The message of those who studied the characteristics of successful software organizations and projects is pretty strong. Both prominent business school researchers, such as Alan McCormack of the Harvard University and Michael Cusumano of the Massachusetts Institute of Technology, and software engineering thought leaders, such as Tom DeMarco, Larry Constantine, and Tim Lister, have repeatedly pointed out to the importance of value factors and the underlying trade-offs in their writings. Since the mid-1980s, the frequently cited CHAOS reports from the Standish Group have consistently identified closely related issues, such as the misalignment of IT spending with organizational objectives and user needs, as sources of failure in software projects. Our main purpose in the production of this book was to draw attention to these issues, which are impossible to reason about in a value-neutral and ad hoc setting.

The Scope of the Book

The International Organization for Standardization (ISO) defines software engineering as "the systematic application of scientific and technological knowledge, methods, and experience to the design, implementation, testing, and documentation of software to optimize its production, support, and quality" (*Information Technology: Vocabulary, Part 1, Fundamental Terms*). While the ISO definition might suffice in a value-neutral setting, we must extend the scope considerably to address value considerations effectively. Three shortcomings of this definition are remarkable from a value-oriented perspective.

First is its exclusion of economics, management science, cognitive sciences, and humanities from the body of knowledge required to create successful software systems. Value-based software engineering however cannot ignore this body of knowledge because it considers software development as a purposeful activity carried out by people for people.

The second shortcoming of the ISO definition is its delimitation of software development by technical activities such as design, implementation, and testing. VBSE in contrast must also consider, as part of the software engineering lifecycle, management-oriented activities – such as business case development, project evaluation, project planning, process selection, project management, risk management, process measurement, and monitoring – that have often been considered peripheral. VBSE as such is a multifaceted, multidisciplinary approach that covers all practices, activities, and phases involved in software development, addressing a wide variety of decisions about technical issues, business models, software development processes, software products and services, and related management practices.

The third shortcoming of the ISO definition is its failure to explicitly recognize the ultimate goal: ensuring that software systems continue to meet and adapt to evolving human and organizational needs to create value. VBSE must put these needs foremost. According to VBSE, it is not enough, or at times not even critical, for software projects to merely meet unilaterally preset schedule, budget, process, and quality objectives. Rather, it is necessary that the resulting products and services persist to increase the wealth of the stakeholders and optimize other relevant value objectives of these projects.

Who Should Read This Book?

This book is intended for those who care about the impact of value considerations in software development activities and decisions. And who should care about such considerations? Well, just about everyone: academics, managers, practitioners, and students of software engineering who recognize that software is not created in a void, that software development involves many participants – executives, project managers, business analysts, developers, quality assurance experts, users, the general public, and so on – with varying roles and stakes in both the final products and the processes used the create those products.

The book appeals particularly to readers who are interested in high-level aspects of software engineering decision making because of its focus on organizational, project-, process-, and product-level issues rather than on low-level, purely technical decisions. The target audience includes, but is not limited to:

- product managers, project managers, chief information officers who make high-level decisions;
- process experts, measurement experts, requirements engineers, business analysts, quality assurance experts, usability experts, and technical leads who participate in various lifecycle activities at key interface points and whose influence span multiple levels and phases;
- software engineering researchers, educators, and graduate students who teach or study software process, evaluate existing and new practices, technologies, methods, or products, or teach or investigate managerial, social, and economic aspects of software development.

To benefit from this book, the reader should have at least taken advanced courses or studied advanced texts on software engineering or software process, or worked in the software industry long enough to acquire an appreciation of the many trade-offs involved from beyond a purely technical perspective.

How Is the Book Organized?

We organized the book in three parts. Part 1 focuses on the foundations of VBSE and provides examples of frameworks for reasoning about value considerations in software development activities. Part 2 provides methods and techniques for VBSE that build upon the foundations and frameworks presented in Part 1. Finally, Part 3 demonstrates the benefits of VBSE through concrete examples and case studies.

While we believe that all chapters contain ideas applicable in a variety of situations, because the book addresses a wide spectrum of issues and activities, certain chapters will inevitably be more relevant to some readers than others, depending on the reader's orientation. We recommend that all readers familiarize themselves with Chapter 1 regardless of their interests, as this chapter sets the tone for the rest of the book. There are many ways to dissect the content according to particular interest areas. We hope that the following road map will help orient the reader who wishes to quickly zoom in on a specific topic.

If you are interested in project-level decisions, economic valuation of software projects and assets, and reasoning under uncertainty, make sure to read Chapters 3, 5, and 17. Readers interested in VBSE-related concepts and theories applicable to a range of software engineering lifecycle activities should start with Chapters 2, 4, 6, and 8. Chapters 7, 9, and 12 are recommended reading for those with an interest in product planning, and Chapters 6, 7, and 9 for those focusing on requirements gathering and negotiation. If the focus is on software process issues and tool adoption, Chapters 6, 8, 13, 15, and 16 discuss approaches that aid in process improvement and measurement as well as impact evaluation. Chapters 4, 10, 11,

and 14 will appeal to the reader interested in product evaluation and testing-related issues. Chapters 8, 14, and 15 will appeal to those who tackle knowledge management problems. Finally, Chapters 3, 5, 6, and 13 are relevant to readers who are interested in risk management.

Whatever your orientation and interests, we hope that the book will inspire you to incorporate value considerations to your own work, or, if you have already been operating in a value-conscious setting, that you will find new insights and resources to draw upon. Good reading!

Acknowledgements

This book would not have been possible without the efforts of many. We are thankful to the authors who contributed the individual chapters and worked diligently with the editors and external reviewers to enhance the quality of the book. At least three reviewers evaluated each chapter and provided extensive feedback to improve the clarity of presentation and ensure technical coherence. Their efforts are much appreciated. We also thank Matthias Heindl, Stefan Kresnicka, Martina Lettner, Muhammad Asim Noor, Barbara Schuhmacher, Norbert Seyff, Rick Rabiser, and Markus Zeilinger for their help during this project. Finally, we thank Springer, our publisher, for trusting our vision, and in particular Ralf Gerstner for his support.

References

(Boehm, 1981) Boehm, B. W.: Software Engineering Economics (Prentice-Hall, 1981)

(Boehm, 1986) Boehm, B. W.: A Spiral Model of Software Development and Enhancement. Software Engineering Notes, **11**(4)

(Boehm et al., 1998) Boehm, B. W., Egyed, A., Kwan, J., Port, D., Shaw, A., Madachy, R.: Using the WinWin Spiral Model: A Case Study. IEEE Computer, (July 1998)

(Boehm, 2003) Boehm, B. W.: Value-Based Software Engineering. Software Engineering Notes, **28**(2):2003

(Favaro, 1996) Favaro, J.: When the Pursuit of Quality Destroys Value. IEEE Software (May 1996)

(Favaro et al., 1998) Favaro, J., Favaro, K. R., Favaro, P. F.: Value-based Reuse Investment, Annals of Software Engineering, **5** (1998)

(McTaggart, 1994) McTaggart, J.: The Value Imperative (The Free Press, 1994)

Table of Contents

List of Contributors

David L. Atkins
Department of Computer Science
American University in Cairo
Cairo 11511, Egypt
Email: datkins@aucegypt.edu

Aybüke Aurum
School of Information Systems, Technology and Management
University of New South Wales
Sydney NSW 2052, Australia
Email: aybuke@unsw.edu.au

Michael Berry
University of New South Wales
School of Computer Science and Engineering
Sydney NSW 2052, Australia
Email: Michael.Berry@student.unsw.edu.au

Stefan Biffl
Institute of Software Technology and Interactive Systems
Technische Universität Wien
Karlsplatz 13, A-1040, Vienna, Austria
Email: Stefan.Biffl@tuwien.ac.at

Barry W. Boehm
University of Southern California
Center for Software Engineering
941 W. 37th Place, SAL Room 328
Los Angeles, CA 90089-0781, USA
Email: boehm@cse.usc.edu

Torgeir Dingsøyr
SINTEF Information and communication technology
Department of Software Engineering
NO-7465 Trondheim, Norway
Email: Torgeir.dingsoyr@sintef.no

Alexander Egyed
Teknowledge Corporation
4640 Admiralty Way, Suite 1010
Marina Del Rey, CA 90292, USA
Email: aegyed@ieee.org

Hakan Erdogmus
Institute for Information Technology,
National Research Council Canada
M50, 1200 Montreal Rd., Ottawa, ON, Canada K1A 0R6
Email: Hakan.Erdogmus@nrc-cnrc.gc.ca

John Favaro
Consulenza Informatica
Via Gamerra 21
56123 Pisa, Italy
Email: john@favaro.net

Ann Fruhling
Department of Computer Science
College of Information Science & Technology
University of Nebraska at Omaha, USA
Email: afruhling@mail.unomaha.edu

Paul Grünbacher
Systems Engineering and Automation
Johannes Kepler University Linz
Altenbergerstr. 69, 4040 Linz, Austria
Email: paul.gruenbacher@jku.at

Michael Halling
Department of Finance
University of Vienna
Brünnerstr. 72, 1210 Vienna, Austria
Email: michael.halling@univie.ac.at

Warren Harrison
Portland State University
1825 SW Broadway
97207 Portland, OR, USA
Email: warren@cs.pdx.edu

Apurva Jain
University of Southern California
Center for Software Engineering
941 W. 37th Place, SAL Room 328
Los Angeles, CA 90089-0781, USA
Email: apurvaja@usc.edu

Sabine Köszegi
Department of Finance
University of Vienna
Brünnerstr. 72, 1210 Vienna, Austria
Email: Sabine.Koeszegi@univie.ac.at

Sebastian Maurice
Software Engineering Decision Support Lab
2500 University Drive NW
Calgary, Alberta, Canada T2N 1N4
Email: smaurice@ucalgary.ca

Audris Mockus
Avaya Corporation
Email: audris@avaya.com

An Ngo-The
Software Engineering Decision Support Lab
2500 University Drive NW
Calgary, Alberta, Canada T2N 1N4
Email: ango@cpsc.ucalgary.ca

Dietmar Pfahl
University of Calgary
Schulich School of Engineering
2500 University Drive NW
Calgary, Alberta Canada T2N 1N4
ICT Building
Email: dpfahl@ucalgary.ca

Rudolf Ramler
Software Competence Center Hagenberg GmbH
Hauptstrasse 99
4232 Hagenberg, Austria
Email: rudolf.ramler@scch.at

Donald J. Reifer
Reifer Consultants, Inc.
P.O. Box 4046
Torrance, CA 90510-4046
Email: d.reifer@ieee.org

Günther Ruhe
iCORE Professor and Industrial Research Chair Software Engineering
2500 University Drive NW
Calgary, Alberta, Canada T2N 1N4
Email: ruhe@ucalgary.ca

Omolade Saliu
Software Engineering Decision Support Lab
2500 University Drive NW
Calgary, Alberta, Canada T2N 1N4
Email: saliu@cpsc.ucalgary.ca

Harvey Siy
Lucent Technologies
Email: hpsiy@lucent.com

Rudolf Vetschera
Department of Business Studies
University of Vienna
Brünnerstr. 72, 1210 Vienna, Austria
Email: rudolf.vetschera@univie.ac.at

Gert-Jan de Vreede
Department of Information Systems & Quantitative Analysis
College of Information Science & Technology
University of Nebraska at Omaha, USA
Email: gdevreede@mail.unomaha.edu

Claes Wohlin
Department of Systems and Software Engineering
Blekinge Institute of Technology Box 520
SE-372 25 Ronneby, Sweden
Email:Claes.Wohlin@bth.se