Software Project Effort Estimation

Adam Trendowicz • Ross Jeffery

Software Project Effort Estimation

Foundations and Best Practice Guidelines for Success

Springer

Adam Trendowicz
Fraunhofer Institute for
Experimental Software Engineering
Kaiserslautern
Germany

Ross Jeffery
The University of New South Wales
Sydney
New South Wales
Australia

ISBN 978-3-319-03628-1 ISBN 978-3-319-03629-8 (eBook)
DOI 10.1007/978-3-319-03629-8
Springer Cham Heidelberg New York Dordrecht London

Library of Congress Control Number: 2014931020

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Foreword

Software effort estimation is one of the oldest and most important problems facing software project management; being able to plan correctly is the basis for all project management activities. One cannot manage a project without the knowledge of what resources are needed to achieve the project goals. It is an area where there has been a great deal of research in the development and fine-tuning of new models and encoding of experience in applying these models.

Today, there are a large number of models, each having different strengths and weaknesses in general and, more importantly, different strengths and weaknesses relative to the environment and context in which they are to be applied, for example, the historical data available and the kinds of factors that are relevant. At the start of a project, it is difficult to understand all the influencing factors and risks; there is a minimal amount of information available. Effort needs to be reestimated at various points in time as the project progresses. And how do you balance early effort commitment against new estimates? What trade-offs are possible?

Which models to apply under what conditions is difficult and requires a great deal of insight into the environment. As with all software engineering approaches and models, it is critical to understand the context in which the approach is to be applied, the model assumptions and context for which the model was developed (not always made clear by the model developer), and how to apply and tailor the model to your context.

This book addresses all these points and provides a large set of model types and classes, focusing on what you need to understand about your environment, what information you need to be able to apply the model, what models are most effective for a particular environment, and how you can learn from the model's application so you can evolve and improve your model over time.

The book is full of insights and useful advice on what to do and how to do it, what to be wary of, and the limitations of effort estimation. Just reading the tips contained in each chapter is a valuable experience.

The book goes beyond effort estimation and provides enormous insights into project management, in general, discussing such issues as project trade-offs, risk assessment, and organizational learning.

This is the most complete work on all aspects of software effort estimation that I have seen and provides an excellent reference for the field. It belongs on the bookshelf of every organization that needs to manage a software project. At the same time, it is an excellent text for a university course on software effort estimation, a topic that is typically insufficiently treated in most curricula.

December 2013

Victor R. Basili
University of Maryland
College Park, MD, USA

On True Success

Past successes, no matter how numerous and universal, are no guarantee of future performance in a new context.

– Henry Petroski

Failure is success if we learn from it.

– Malcolm Forbes

Success consists of going from failure to failure without loss of enthusiasm.

– Winston Churchill

To be defeated and not submit, is victory; to be victorious and rest on one's laurels, is defeat.

– Józef Piłsudski (First Marshal of Poland)

Preface

The time for action is now. It's never too late to do something.

—Antoine de Saint-Exupery.

What Is This Book About?

In this book, we focus on the estimation of software development effort. Three aspects are considered important for the proper handling of effort estimation: (1) *foundations* of software effort estimation, (2) selecting the most suitable estimation *approach*, and (3) successfully using effort estimation in specific *contexts*.

What Is This Book NOT About?

This book does not include project planning activities that typically follow effort estimation. We do not discuss such aspects as how to allocate project resources to work tasks, how to sequence work activities, how to determine critical paths, and how to resolve resource conflicts. Finally, we are not addressing project scheduling or budgeting. We refer readers interested in these subjects to books that address project management topics, for example, the PMI's (2013) Project Management Body of Knowledge (PMBOK Guide) or OGC's (2009) PRINCE2, which offer very useful overviews of common project management practices.

To Whom Is This Book Addressed?

In its very early stage, this book was intended as a collection of notes, where the most relevant estimation principles, definitions, and empirical observations, found in the literature and from experience, were gathered. In the course of time, this was shared with others. This book aims to inherit the intention of these initial notes and the needs of people they were shared with. It is addressed to those who want to take

actions in order to improve their estimation practices, yet are missing (1) the necessary knowledge and understanding of estimation principles and (2) a concise reference of best practices and most common estimation approaches they can start with and adapt to their particular needs. This book assumes one prerequisite about its intended audience: it assumes that readers believe that it is never too late to do something about your estimation practices, irrespective of whatever shape they are now in.

Software Practitioners

This book is intended for all software practitioners responsible for software effort estimation and planning in their daily work. This includes primarily, but is not limited to, those who are responsible for introducing and maintaining estimation practices in a software development organization.

Students

In this book, we also appreciate the value of the old saying "as the twig is bent, so grows the tree" and address the content to students of software engineering programs, particularity project and process management courses.

How to Read This Book

We anticipated this book to be a reference guidebook you can grab whenever you need to learn or recall specific aspects of effort estimation. The way you read the book depends on your particular needs at a given moment. So before you start, think for a moment—what do you want to achieve?

- *If you want to understand the basic challenges and principles of software effort estimation*, read Chaps. 1 and 2.
- *If you want to master* the *principal concepts and techniques of existing estimation methods*, read Chaps. 3–5 and the Appendix.
- *If you want to select the most suitable estimation method for estimating software development effort in your specific context*, read Chaps. 6 and 7.
- *If additionally you want to get a quick insight into the most common estimation methods*, *including their prominent strengths and weaknesses*, read Chaps. 8–15, or only some of them if you are interested in any specific method we present there.
- *If you want to introduce a new estimation approach or improve the one you have been using*, read Chap. 16.
- *In any case, read the best-practice guidelines* we present in Chap. 17.

Moreover, each part of the book begins with a brief summary of the chapters it encompasses. Refer to these summaries to quickly decide which chapter to read.

Key Terminology Used in This Book

In this book, we use several basic terms, which in other literature and in practice are often used interchangeably. In order not to confuse the reader, we would like to start by clarifying the most important terms we will use throughout the text.

Cost Versus Effort

Although principally and intuitively different, the terms "cost" and "effort" are often used as synonyms in the software project management area. The Webster dictionary defines cost as "the amount or equivalent paid or charged for something" and effort as "conscious exertion of power" or "the total work done to achieve a particular end". In the software engineering domain, cost is defined in a monetary sense, and with respect to software development projects, it refers to partial or total monetary cost of providing (creating) certain products or services. Effort, on the other hand, refers to staff time spent on performing activities aimed at providing these products or services. In consequence, project cost includes, but is not limited to, project effort. In practice, cost includes such elements as fixed infrastructure and administrative costs for example. Moreover, dependent on the project context (e.g., currency or cost of staff unit) despite the same project effort, project cost may differ.

In the software engineering literature and practice, "cost" is often used as a synonym for "effort." One of the ways to notice the difference is to look at units used. Cost in a monetary sense is typically measured in terms of a certain currency (e.g., $, €, ¥, etc.), whereas cost in an effort sense is typically measured as staff time (e.g., person-hours, person-days, person-months, etc.).

In this book, we focus on estimating software development effort, and we consistently differentiate between cost and effort.

Estimation Versus Prediction Versus Planning

In software engineering, effort estimation, prediction, and planning are related to each other; yet, they have different meanings, that is, they refer to different project management activities. Actually, the dictionary definitions perfectly reflect the differences between these three processes:

- *Estimation*: "the act of judging tentatively or approximately the value, worth, or significance of something"
- *Prediction*: "the act of declaring or indicate in advance; especially: foretelling on the basis of observation, experience, or scientific reason"
- *Planning*: "the act or process of making or carrying out plans; *specifically*: the establishment of goals, policies, and procedures for a social or economic unit"

Estimation Versus Prediction

Both estimation and prediction contain an element of uncertainty; the first refers to approximating an actual state, whereas the latter refers to a future state. Simplifying, we may define prediction as estimating in advance. Since in software engineering, effort estimation refers to approximating development effort in advance, before development is completed, it should actually be called effort prediction. Yet, in practice, both terms are used interchangeably. In this book, we will follow this practice and use estimation and prediction as synonyms for foretelling the effort required for completing software development projects.

Prediction Versus Planning

There is, however, a significant difference between prediction and planning. Prediction refers to an unbiased, analytical process of approximating a future state. Planning, on the other hand, refers to a biased process of establishing goals with respect to the future state. Although predictions form a foundation for planning, plans do not have to be (and typically are not) the same as predictions. In the case of software development, the goal of prediction is to accurately foretell resources (such as effort) required to provide project outcomes. The goal of effort planning is, on the other hand, is to plan the project in such a way that the project goals are achieved. In other words, we plan means within a project to achieve a specific project's end.

Kaiserslautern, Germany Adam Trendowicz
Sydney, NSW, Australia Ross Jeffery

Acknowledgments

A number of great people and organizations have made their explicit or implicit contribution to this book by inspiring us, contributing to our knowledge, or helping us in the creation of this book. Hereby, we would like to express our gratitude to these people and organizations.

We would like to thank the reviewers of the book manuscript, Yasushi Ishigai and Mirosław Ochodek, for their valuable remarks.

Adam Trendowicz would like to express special thanks to Yasushi Ishigai for recent years of collaborative work in industrial contexts and great discussions on the industrial challenges regarding effort estimation and potential solutions to these challenges. Moreover, he would like to thank Fraunhofer Institute for Experimental Software Engineering (IESE), Kaiserslautern, Germany, for giving him an opportunity to develop their professional expertise, including software effort estimation.

Ross Jeffery would like to thank National ICT Australia (NICTA) and the University of New South Wales, who have supported his research for many years. He would also like to thank the many academic and industry colleagues who have assisted with the research in effort estimation.

We would like to convey special thanks to Mr. Ralf Gerstner and Ms. Victoria Meyer from Springer for their great support in copyediting this book.

Last but not least, we seek forgiveness from all those whose names we have failed to mention.

Disclaimer

Any of the trademarks, service marks, collective marks, registered names, or similar rights that are used or cited in this book are the property of their respective owners. Their use here does not imply that they can be used for any purpose other than for the informational use as contemplated in this book. Rather than indicating every occurrence of a trademarked name as such, this report uses the names only with no intention of infringement of the trademark. The following table lists trademark names used in this book.

Trademark	Subject of trademark	Trademark owner
CoBRA®	Cost Estimation, Benchmarking, and Risk Assessment	Fraunhofer Institute for Experimental Software Engineering (IESE)
GQM+Strategies®		Fraunhofer Institute for Experimental Software Engineering (IESE)
CMMI®	Capability Maturity Model Integrated	Software Engineering Institute (SEI)
MS Office®	MS Word®, MS Excel®, and MS PowerPoint®	Microsoft® Corporation
PMBOK®	Project Management Body of Knowledge Guide	Project Management Institute (PMI)
PRINCE2™	Projects in Controlled Environments 2	Office of Government Commerce (OGC)

Acronyms

AC	Actual cost
ACWP	Actual cost of work performed
AHP	Analytic hierarchy process
ANGEL	Analogy estimation tool
ANN	Artificial neural networks
AVN	Analogy with virtual neighbor
BBN	Bayesian belief network
BCWP	Budgeted cost of work performed
BCWS	Budgeted cost of work scheduled
BRACE	Bootstrap-based analogy cost estimation
BRE	Balanced relative error
CART	Classification and regression trees
CASE	Computer-aided software engineering
CI	Confidence interval
CMMI	Capability maturity model integrated
CoBRA	Cost estimation, benchmarking, and risk assessment
COCOMO	Constructive cost model
COTS	Commercial off-the-shelf
DAG	Directed acyclic graph
DBMS	Database management system
EF	Experience factory
EQF	Estimating quality factor
EO	Effort overhead
ESA	European Space Agency
EV	Earned value
EVM	Earned value management
FP	Function points
FPA	Function points analysis
GAO	US Government Accountability Office
GP	Genetic programming
GQM	Goal-question-metric
IEEE	Institute of Electrical and Electronics Engineers
IFPUG	International Function Point Users Group

IRQ	Interquartile range
ISBSG	International Software Benchmarking Standards Group
JPD	Join probability distribution
KPA	Key process area
LAD	Least absolute deviation
LMS	Least median of squares
LOC	Lines of code
MCDA	Multi criteria decision analysis
MIS	Management information systems
MMRE	Mean magnitude of relative error
MRE	Magnitude of relative effort
MSE	Mean squared error
MSWR	Manual stepwise regression
NPT	Node probability table
OEM	Original equipment manufacturer
OLS	Ordinary least squares
OS	Operating system
PDCA	Plan-do-check-act
PDR	Product design review
PERT	Program evaluation and review technique
PI	Prediction interval
PMI	Project Management Institute
PMBOK	Project Management Body of Knowledge
POP	Predictive object points
PRINCE	Projects in controlled environments
PROBE	Proxy-based estimation
PV	Planned value
QA	Quality assurance
QIP	Quality improvement paradigm
QSM	Quantitative software management
RE	Relative estimation error
ROC	Rank order centroid
RR	Robust regression
SEER-SEM	Software Evaluation and Estimation of Resources-Software Estimating Model
SEI	Software Engineering Institute
SLIM	Software lifecycle management
SLOC	Source lines of code
SMART	Specific, measurable, attainable, relevant, timely
SPI	Software process improvement
SPR	Software productivity research
UCP	Use-case points
WBS	Work breakdown structure

Contents

About the Authors

Adam Trendowicz is a senior consultant at the Fraunhofer Institute for Experimental Software Engineering (IESE) in Kaiserslautern, Germany, where he leads the team of "Measurement and Prediction." He received his Ph.D. in Computer Science from the University of Kaiserslautern (Germany). Dr. Trendowicz has led software cost estimation and software measurement improvement activities in software companies of different sizes and from various domains (e.g., in Germany, Japan, and India). He has been involved in functional software size estimation (Function Point Analysis) and productivity benchmarking in organizations from both industry and the public sector. Dr. Trendowicz has taught several tutorials on software cost estimation and supervised the "Software Economics and Risk Management" module within the distance master studies program "Software Engineering for Embedded Systems"—a program developed jointly by the University of Kaiserslautern and Fraunhofer IESE. Finally, Dr. Trendowicz has authored the book titled *Software Cost Estimation, Benchmarking, and Risk Assessment. The Software Decision-Makers' Guide to Predictable Software Development*. Moreover, he has coauthored more than 20 international journal and conference publications. Dr. Trendowicz's other software engineering interests include (1) project management, (2) software product quality modeling and evaluation, and (3) technology validation by means of empirical methods.

Ross Jeffery is Emeritus Professor of Software Engineering in the School of Computer Science and Engineering at the University of New South Wales and research consultant in the Systems Software Research Group in National ICT Australia (NICTA). His research interests are in the 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 50 government and industry organizations over a period of 20 years and has been funded by industry, government, and universities. He has coauthored 4 books and over 190 research papers. He has served on the editorial

board of the *IEEE Transactions on Software Engineering*, the *Journal of Empirical Software Engineering*, and the Wiley International Series in Information Systems. He was 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.