Jane Cleland-Huang · Orlena Gotel
Andrea Zisman *Editors*

Software and Systems Traceability

Springer

Jane Cleland-Huang · Orlena Gotel ·
Andrea Zisman *Editors*

Software and Systems Traceability

Foreword by Anthony Finkelstein

Springer

Editors
Jane Cleland-Huang
DePaul University
School of Computing
243 S. Wabash Avenue
60604 Chicago
USA
jhuang@cs.depaul.edu

Orlena Gotel
New York
NY 10014
USA
olly@gotel.net

Andrea Zisman
City University
School of Informatics
London
United Kingdom
a.zisman@soi.city.ac.uk

ISBN 978-1-4471-2238-8 e-ISBN 978-1-4471-2239-5
DOI 10.1007/978-1-4471-2239-5
Springer London Dordrecht Heidelberg New York

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

Library of Congress Control Number: 2011941143

Requirements and Relationships: A Foreword

Software engineering is a pessimistic discipline. The glass is always half empty rather than half full. Not surprising really, we are hardened to the grind of improving quality, painstakingly testing and, never quite, eliminating bugs. Critical review is of the essence. We know there is "no silver bullet".

Traceability in software development must however, pessimism set aside, be marked as a success. We have characterised the problem. We have produced industrial strength tools that relieve a substantial part of the practical difficulties of managing traceability relations across different documents. We have arrived at a communal consensus regarding the principal notations to be used in software development, realised in UML, and characterised the relationships amongst these notations. These are all significant practical advances.

Research has gone further. One of the key challenges of traceability has been the return on investment. In essence only a few of the traceability links prove to be of value, that is are subsequently needed in support of a change. It is difficult to predict in advance however, which these might be. Given that establishing, documenting and managing traceability manually is expensive, the balance of costs and benefits is delicate one. It has been shown, convincingly in my view, that off-the-shelf information retrieval techniques will, with some judicious tuning, yield reasonable traceability links. I expect this, once industrially hardened and deployed, to drive cost reduction.

I guess with all this positivity you can sense a "but" coming . . . and you are not wrong. While we have taken steps to advance the state of the art, the nature of the requirements challenge has shifted. The context has altered. Agile development has altered the way that much software is developed (just in case there is any remaining doubt, it is no longer a phenomenon of the programming fringe – it is mainstream software engineering). But agile development is really only a particular manifestation of the underlying trends in which it is becoming clear that it is cheaper to build software quickly, and change it if it fails to satisfy the emerging requirements, than to undertake the discipline of trying to get it exactly right at the outset. This is partly a technical change, the product of improved tools, environments and programming languages, but may also reflect changing business environments, that move at a pace set by a dynamic globalised economy. So we start with more change, indeed with constant change, not simply as an unwanted consequence of the inexorable laws of software evolution but embraced as the essence of software engineering.

More change means a greater need for traceability support. Of course, if you have adopted an agile approach you could argue that there is less to trace to, after all you have in large part eschewed documentation. This, I believe, is an error because it ignores the consequentially altered nature of the requirements task. I will elaborate below.

We have tended to view requirements as a discrete task in which we engage with the customer (a sort of shorthand for stakeholders) on an occasional basis. We are not, any longer, so naive as to believe that requirements elicitation is a one-shot process, but we still understand it to be something that happens from time to time, for clearly specified purposes.

Change changes things. Requirements engineering becomes instead a "relational" process in which the name of the game is continuing customer engagement. In other words, the developer tries to ensure that their application or service grows and adapts in sync with, ideally at the leading edge of, the customer's business. You could say the software is a manifestation of the relationship achieved through continuous interaction and immersion in the business. Managing this ongoing relationship and the associated knowledge of the domain is difficult and demands, I suggest, a different approach on the part of the software developer and a reimagining of requirements elicitation, specification and validation.

So, where does requirements traceability fit into this picture? It provides the information management support for these complex multi-threaded customer relationships and the technical substrate for rapid system evolution. It allows the developer to understand and account for the consequences of ongoing system change in terms of the business. It is the core of a new type of "customer relationship management" system.

I wish I had a better sense of what the new technical demands that follow from the change of view, sketched above, might be. Many of the colleagues, whose work makes up this volume, are better equipped than I am to do this.

Of course, there remains a hard core of large systems development characterised by strong safety and other constraints and bound to the co-development of complex hardware where the agility sketched above has limited impact. Defence and other mission-critical systems exemplify this. There is a continuing need to address traceability in this setting and in particular to support navigation of the complex relationships that arise. Of particular interest, and relevant in the light of the analysis above, are regulatory and compliance processes that engage a demanding framework of requirements and shifting body of stakeholders. This still remains at the edge of what can be practically accomplished and will require further research. This book sheds strong light on the challenges.

I am certain that the technical achievements marked in this volume are the basis for addressing these new frontiers for software and systems engineering and that requirements traceability will be at the forefront of engineering research. Not so pessimistic, really.

London, UK Anthony Finkelstein

Preface

The importance of traceability is well understood in the software engineering community and adopted across numerous software development standards. Industries are often compelled to implement traceability practices by government regulations. For example, the U.S. Food and Drug Administration (FDA) states that traceability analysis must be used to verify that a software design implements all of its specified software requirements, that all aspects of the design are traceable to software requirements, and that all code is linked to established specifications and established test procedures. Other examples are found in the U.S. Federal Aviation Administration (FAA) that states that software developers need to have ways of demonstrating traceability between design and requirements, and in the Capability Maturity Model Integration (CMMI) standard that requires similar traceability practices.

Traceability supports numerous critical activities. For example, pre-requirements traceability is used to demonstrate that a product meets the stakeholders' stated requirements, or that it complies with a set of government regulations. Traceability is also used to establish and understand the relationships between requirements and downstream work products such as design documents, source code, and test cases. In this context, it supports tasks such as impact analysis which helps developers understand how a proposed change impacts the current system, and code verification which identifies superfluous and unwanted features by tracing all elements of the source code back to specific requirements. Traceability can also support reuse of parts of a software system by identifying the parts that match (new) requirements, and the evolution of software systems.

In practice, traceability links are typically created and maintained either through the use of a requirements management tool, or else in a spreadsheet or Word document directly. However, there are numerous issues that make it difficult to achieve successful traceability in practice. These issues include social ones related to communication between project stakeholders, as well as technical issues related to physically creating, maintaining, and using thousands of interrelated and relatively brittle traceability links. As a result, many organisations struggle to implement and maintain traceability links, even though it is broadly recognised as a critical element of the software development life cycle.

In order to overcome the significant challenges in creating, maintaining, and using traceability, over the last 20 years the research community has been actively addressing traceability issues through the exploration of topics related to automating the traceability process, developing strategies for cost-effective traceability, supporting the evolution and maintenance of traceability links, visualising traceability, and developing traceability practices that apply across a wide range of domains such as product lines, multi-agent systems, safety critical applications, aspect-oriented and agile software development, and various regulated industries.

Several workshops and symposia have been organised by the traceability community to bring together researchers and practitioners in order to address the challenges and discuss state-of-the-art work in the area of traceability. These events include the Traceability in Emerging Forms of Software Engineering (TEFSE) workshop series[1]; and the workshops funded by NASA (held at NASA's IV&V facility in 2006) and the NSF (held in Lexington, Kentucky in 2007 in conjunction with TEFSE 2007) that resulted in the creation of a draft Problem Statement and Grand Challenges document.

Another effort of the community was the creation of the International Center of Excellence for Software Traceability (CoEST) in 2005. The main goals of CoEST are to promote international research collaborations; advance education in the traceability area; bring together researchers, practitioners, and experts in the field; create a body of knowledge for traceability; develop a repository of benchmarks for traceability research; and develop new technologies to satisfy traceability needs. More recently, the community has also engaged in the Tracy project, funded by the NSF, with the focus of building research infrastructure, collecting and organising datasets, establishing benchmarks, and developing a tool named TraceLab to provide support for designing and executing a broad range of traceability experiments.

This book complements the current effort of the traceability community by providing a comprehensive reference for traceability theory, research, and practice and by presenting an introduction to the concepts and theoretical foundations of traceability. Several topics in this book represent areas of mature work, which have previously only appeared as research papers in conference proceedings, journals, or individual book chapters. The book therefore serves as a unifying source of information on traceability. As such, we expect the book to serve as a reference for practitioners, researchers, and students. Practitioners reading the book may be especially interested in the mature areas of traceability research, several of which have already been demonstrated to work in industry through various pilot studies, while researchers from all areas of the community may be specifically interested in the cutting edge nature of several topics and the open research challenges that need to be addressed in the future. Students new to the topic should start with a review of the fundamentals in the chapter "Traceability Fundamentals".

[1] TEFSE 2002: Edinburgh, UK; TEFSE 2003: Montreal, Canada; TEFSE 2005: Long Beach, CA; TEFSE 2007: also known as the Grand Challenges of Traceability, Lexington, Kentucky; TEFSE 2009, Vancouver, Canada; TEFSE 2011: Honolulu, Hawaii.

The book contains 16 chapters organised in five Parts. *Part I – Traceability Strategy* describes several traceability terms and concepts, and the activities related to traceability planning and management. *Part II – Traceability Creation* presents a variety of techniques for supporting the creation of trace links. These techniques include the use of Information Retrieval and rule-based methods, an account of the factors that impact traceability creation, methods to create traceability together with the development of software systems, and techniques for traceability creation among heterogeneous artifacts. *Part III – Traceability Maintenance* presents approaches that support traceability in evolving projects in the domains of product line systems and model-driven engineering, as well as the role of the human in the traceability process. *Part IV – Traceability Use* describes the employment of traceability in agile projects, aspect-oriented software development, non-functional requirements, and medical devices. *Part V – Traceability Challenges* presents the outstanding challenges for traceability research and practice, based on a community vision for traceability in 2035, and discusses the open traceability research topics that need to be addressed in the future.

The book also provides a copy of a glossary of traceability terms created by members of the traceability community and used in the material described in the various chapters of the book. The topics presented in these various chapters are illustrated by two case studies in the areas of electronic health care and mobile phone product line systems. The book also provides an overview of the Center of Excellence for Software Traceability and the TraceLab tool. All the above materials are presented in five different appendices in the book.

This book is the product of several years of effort. Andrea Zisman first conceived of the idea in early 2009 and the finished product was brought together in its current form as the result of numerous emails, skype calls, and face-to-face discussions between all three of the editors.

Obviously, any book of this nature demands the contributions and efforts of many different people. This book was no different, and we would like to thank members of the traceability community for their willingness to contribute their time and effort to make this book possible. The process of collecting material for the book was initiated by a call for abstracts in June 2010. At that time, we selectively invited the most promising abstracts for submission as full chapters, and also reached out to request additional chapters for a few missing topics. All submitted chapters went through a rigorous peer-review process and, as a result, we selected the chapters that are presented in this book. We thank the authors of all abstracts and chapters for their contributions to this process.

Chicago, USA	Jane Cleland-Huang
New York, USA	Orlena Gotel
London, UK	Andrea Zisman

Acknowledgments

We would like to thank the following people for their contributions.

Melissa Huang for her illustrations of Traceability Strategy, Traceability Creation, Traceability Maintenance, Traceability Use, Traceability Challenges, and the Appendices, which appear in each of the six sectional headers respectively.

Aleksandra Waliczek for managing the logistics of coordinating the final process of collecting chapters and supporting material from the authors and for integrating all of the material into the draft version of the book for delivery to Springer.

John Van Ort for compiling author information.

The US National Science Foundation (NSF) for partially funding community work on the Grand Challenges of Traceability, CoEST (Center of Excellence for Software Traceability), and TraceLab under grants CNS 0959924 and 0647443.

The US National Aeronautics and Space Administration (NASA) for providing initial seed funding for CoEST and for the first workshop on the Grand Challenges of Traceability under grant NNX06AD02G.

All of the authors who contributed abstracts and/or full chapters as a result of the initial call for chapters.

The anonymous reviewers, without whom this peer-reviewed book would not be possible.

Contributors

Nasir Ali DGIGL, École Polytechnique de Montréal, Montréal, QC, Canada, nasir.ali@polymtl.ca

Giuliano Antoniol École Polytechnique de Montréal, Montréal, QC, Canada, antoniol@ieee.org

Hazeline U. Asuncion Computing and Software Systems, University of Washington, Bothell, WA, USA, hazeline@u.washington.edu

John Burton Vitalograph Ireland Ltd., Ennis, Ireland, John.burton@vitalograph.ie

Valentine Casey Regulated Software Research Group, Lero, Dundalk Institute of Technology, Dundalk, Ireland, Val.casey@dkit.ie

Jane Cleland-Huang DePaul University, School of Computing, 60604 Chicago, USA, jhuang@cs.depaul.edu

Gerry Coleman Regulated Software Research Group, Lero, Dundalk Institute of Technology, Dundalk, Ireland, Gerry.coleman@dkit.ie

Wouter De Borger DistriNet Research Group, K.U. Leuven, B-3001 Heverlee, Belgium, wouter.deborger@cs.kuleuven.be

Alex Dekhtyar Cal Poly State University, San Luis Obispo, CA, USA, dekhtyar@calpoly.edu

Andrea De Lucia University of Salerno, Fisciano (SA), Italy, adelucia@unisa.it

Peter Donnelly Regulated Software Research Group, Lero, Dundalk Institute of Technology, Dundalk, Ireland, Peter@biobusinessni.org

Alexander Egyed Johannes Kepler University, Linz, Austria, alexander.egyed@jku.at

Anthony Finkelstein University College London, London, UK, a.Finkelstein@cs.ucl.ac.uk

Holger Giese Hasso-Plattner-Institute at the University of Potsdam, 14482 Potsdam, Germany, holger.giese@hpi.uni-potsdam.de

Orlena Gotel New York, NY 10014, USA, olly@gotel.net

Paul Grünbacher Systems Engineering and Automation, Johannes Kepler University, Linz, Austria, paul.gruenbacher@jku.at

Yann-Gäel Guéhéneuc DGIGL, École Polytechnique de Montréal, Montréal, QC, Canada, yann-gael.gueheneuc@polymtl.ca

Jane Huffman Hayes University of Kentucky, Lexington, KY, USA, hayes@cs.uky.edu

Regina Hebig Hasso-Plattner-Institute at the University of Potsdam, 14482 Potsdam, Germany, regina.hebig@hpi.uni-potsdam.de

Wolfgang Heider Christian Doppler Laboratory for Automated Software Engineering, Johannes Kepler University, Linz, Austria, heider@ase.jku.at

Claire Ingram Newcastle University, NE1 7RU, England, UK, claire.ingram@ncl.ac.uk

Waraporn Jirapanthong Faculty of Information Technology, Dhurakij Pundit University, Bangkok 10210, Thailand, waraporn.jir@dpu.ac.th

Wouter Joosen DistriNet Research Group, K.U. Leuven, B-3001 Heverlee, Belgium, wouter.joosen@cs.kuleuven.be

Bert Lagaisse DistriNet Research Group, K.U. Leuven, B-3001 Heverlee, Belgium, bert.lagaisse@cs.kuleuven.be

Martin Lehofer Siemens VAI Metals Technologies, Linz, Austria, martin.lehofer@siemens.com

Patrick Mäder Institute for Systems Engineering and Automation (SEA), Johannes Kepler University, Linz, Austria, patrick.maeder@jku.at

Jonathan Maletic Kent State University, Kent, OH, USA, jmaletic@cs.kent.edu

Andrian Marcus Wayne State University, Detroit, MI 48202, USA, amarcus@wayne.edu

Fergal Mc Caffery Regulated Software Research Group, Lero, Dundalk Institute of Technology, Dundalk, Ireland, Fergal.McCaffery@dkit.ie

Andrew Meneely Department of Software Engineering, Rochester Institute of Technology, andg@se.rit.edu

Mehdi Mirakhorli Depaul University, Chicago, IL, USA, m.mirakholi@acm.org

Rocco Oliveto University of Molise, Pesche (IS), Italy, rocco.oliveto@unimol.it

Denys Poshyvanyk The College of William and Mary, Williamsburg, VA 23185, USA, denys@cs.wm.edu

Rick Rabiser Christian Doppler Laboratory for Automated Software Engineering, Johannes Kepler University, Linz, Austria, rabiser@ase.jku.at

Steve Riddle Newcastle University, NE1 7RU, England, UK, steve.riddle@ncl.ac.uk

Andreas Seibel Hasso-Plattner-Institute, The University of Potsdam, 14482 Potsdam, Germany, andreas.seibel@hpi.uni-potsdam.de

M.S. Sivakumar Regulated Software Research Group, Lero, Dundalk Institute of Technology, Dundalk, Ireland, Smadh09@studentmail.dkit.ie

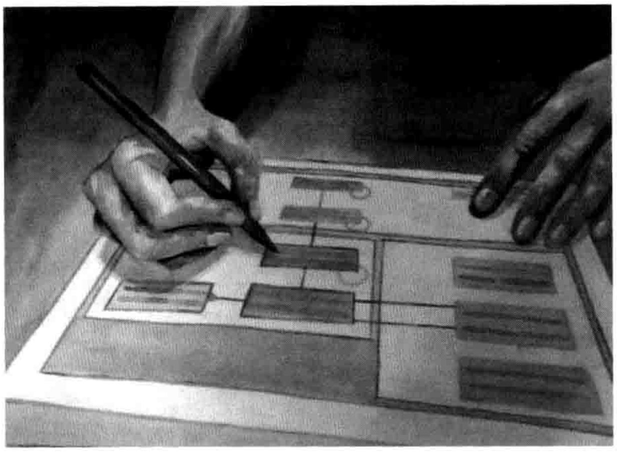
Ben Smith Department of Computer Science, North Carolina State University, Raleigh, NC 27695-8206, USA, bhsmith3@ncsu.edu

Richard N. Taylor Institute for Software Research, University of California, Irvine, CA, USA, Taylor@ics.uci.edu

Laurie Williams Department of Computer Science, North Carolina State University, Raleigh, NC 27695-8206, USA, lawilli3@ncsu.edu

Andrea Zisman School of Informatics, City University London, London, EC1V 0HB, UK, a.zisman@soi.city.ac.uk

Contents



Part I
Traceability Strategy

Traceability needs to be planned for and managed if it is to be effective and remain effective in any particular project context. Stakeholders need to be identified and requirements determined. A suitable traceability process needs to be designed and potential support from tooling explored. However, all this initial effort is mute if there is no clear understanding of the anticipated return on investment from implementing traceability within an organisation. Traceability strategy comprises all those activities associated with traceability planning and traceability management.

In this first part of the book, the chapter "Traceability Fundamentals" defines a number of traceability-related terms and concepts, as they will be used throughout the remainder of the book. A simple process for analysing the cost-benefit of traceability and selecting a strategy accordingly is described in the chapter "Cost-Benefits of Traceability". A cautionary seven-step guide for making informed decisions about tool acquisition is presented in the chapter "Acquiring Tool Support for Traceability". In combination, the chapters "Cost-Benefits of Traceability" and "Acquiring Tool Support for Traceability" highlight important considerations to help plan and manage traceability in practice.

Traceability Fundamentals

Orlena Gotel, Jane Cleland-Huang, Jane Huffman Hayes, Andrea Zisman, Alexander Egyed, Paul Grünbacher, Alex Dekhtyar, Giuliano Antoniol, Jonathan Maletic, and Patrick Mäder

1 Introduction

The role of traceability was recognised in the pioneering NATO working conference held in 1968 to discuss the problems of software engineering (Naur and Randell, 1969). One of the working papers in this conference examined the requirements for an effective methodology of computer system design and reported on the need to be able to ensure that a system being developed actually reflects its design. In a critique of three early projects focused on methodology, each was praised for the emphasis they placed on making "the system that they are designing contain explicit traces of the design process" (Randell, 1968).

Traceability was subsequently noted as a topic of interest in one of the earliest surveys on the state of the art and future trends in software engineering (Boehm, 1976), and its practice was certainly evident in those domains concerned with developing early tool support (Dorfman and Flynn, 1984; Pierce, 1978). By the 1980s, traceability could be found as a requirement in a large number of national and international standards for software and systems development, such as the high-profile DOD-STD-2167A (Dorfman and Thayer, 1990). Published research began to proliferate and diversify in the area of traceability in the late 1990s, spurred somewhat by renewed interest in the topic arising from two newly formed International Requirements Engineering professional colloquia, with two early papers focusing on the issues and problems associated with traceability (Ramesh and Edwards, 1993; Gotel and Finkelstein, 1994), the latter providing for the first systematic analysis of the traceability problem. The topic of traceability continues to receive growing research attention in the twenty-first century, with a particular focus on automated trace generation (Cleland-Huang et al., 2007; Hayes et al., 2006) and with concomitant advances in model-driven development (Aizenbud-Reshef et al., 2006; Galvao and Goknil, 2007; Winkler and von Pilgrim, 2010).

O. Gotel (✉)
New York, NY 10014, USA
e-mail: olly@gotel.net