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

Printed on acid-free paper

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

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