Software Evolution
To Inge, Sara and Paulien, 
for being there – Tom Mens

To Ann, Sara, Niels and Jens, 
for illustrating the value of life – Serge Demeyer
Foreword

by Mehdi Jazayeri

Faculty of Informatics, University of Lugano, Switzerland
Distributed Systems Group, Technical University of Vienna

The phenomenon of software evolution was observed back in the 1970s when the first large software systems were being developed, and it attracted renewed attention in the 1990s. Software evolution is now a common phrase and an accepted research area in software engineering. There are conferences and workshops devoted to the topic, and evolution papers appear frequently in the traditional software engineering conferences and journals. The 2004 ACM/IEEE Software Engineering Curriculum Guidelines list software evolution as one of ten key areas of software engineering education. And there are several research groups and international networks working on software evolution. As perhaps may be expected, there are diverging research efforts in sub-areas of software evolution, spanning theoretical studies, empirical studies, tools, visualization, and so on.

Since the classic and insightful work of Lehman and Belady [320], “software evolution” has been accepted as a phenomenon worth studying and one that we acknowledge poses serious problems to software projects. The problems are complex because they involve many dimensions, affecting, among others, all phases of the software process, managerial and economic aspects, and programming languages and environments. Further, as software engineering advances and new technologies (e.g., Web applications) and processes (e.g., open source) are introduced, software evolution faces different problems and challenges. At the same time, some new advances (e.g. agile and model-driven processes) enable novel solutions to software evolution.

Evolution in general parlance implies that something has changed for the better. The Merriam-Webster Dictionary defines evolution as “a process of continuous change from a lower, simpler, or worse to a higher, more complex, or better state,” which captures our intuitive notion of something improving. With software, evolution is multi-faceted because certainly according to some metric the software gets better, for example it acquires a new feature or its performance improves or it is ported to a new platform. Unfortunately, most improvements come with some deterioration in some other dimension, for example, size of software or its performance or its structure.
In biology, the traditional area of evolution, evolution deals with species. Is there something analogous to “species” when we talk about software? The answer is definitely yes. The species are the high-level models that we use to describe (aspects of) software. An architectural description, in fact, describes a whole species of software systems. The family architecture (or product line) approach to software development makes this explicit by capturing a whole family (species) of systems in terms of their commonalities and differences. If evolution does take place in software, we can hope that it occurs at these meta-levels, where new architectures are created as improvements to previous architectures, leading to evolved species. Individual elements in the family certainly change over time but this change is hardly evolutionary in the sense that it leads to long-term improvement. What we do know about software and even Lehman’s laws of evolution is that any individual software system will eventually reach an old age when it is no longer cost-effective to modify it and it is better to retire it. But even when we retire a software product, the associated knowledge about that product, captured in higher level models such as requirements and specifications lives on and influences the evolution of the species. Thus, understanding and capturing the way software evolves offers a fascinating and rich area of study.

With this wide range of issues involved in software evolution, where would a researcher new to the field turn to for an introduction and comprehensive overview of the state of the art? This book attempts to be that source. For example, this book is a good starting point for a PhD student looking for a research topic. It can also form the basis for a seminar course on software evolution. The book covers most areas of software evolution and many current problems and representative research approaches. I recommend the book to any researcher interested in software evolution.

The book, however, has value beyond the world of research. Because of the key role that evolution plays in software engineering, knowledge of the problems, approaches and solutions in software evolution is useful to anyone involved in software engineering. Thus, if you are a software engineer, or software engineering researcher, interested or just curious about what happens to software once it is developed, or how to develop software that is evolvable, this book offers you plenty of insights.
Preface

In October 2002, on a cold wintry Monday in Antwerp, we kicked off the RELEASE network, a research network aiming to establish “Research Links to Explore and Advance Software Evolution”. This research network (funded by the European Science Foundation) was an attempt to intensify the collaboration between a number of European research groups active in the field of software evolution. At that time, software evolution was steadily becoming a subject of serious academic study, because more and more researchers started to recognise that building software that lasts is one of the key challenges for our society in general and for the software engineering community in particular. The RELEASE network succeeded in fostering a community of European researchers who continue to meet on a regular basis, despite ceasing of funding in 2005. The book you are holding right now is one of the products of this continued activity and we sincerely hope that it will inspire you to become part of the active software evolution community as well.

What Is this Book About?

This book is a collection of chapters written and peer reviewed by renowned experts in the field of software evolution. The book does not cover all research topics in software evolution—given the wealth of information in this field that would be an impossible task. Instead, we focus on novel trends in software evolution research and its relation with other emerging disciplines such as model-driven software engineering, service-oriented software development, aspect-oriented software development. Also, we do not restrict ourselves to the evolution of source code only, but address evolution of other equally important software artefacts such as databases and database schemas, design models, software architectures, and so on. As such, this book provides a representative selection of the research topics under study in this field. Even better, it also demonstrates the diverse ways on how to conduct research in this field, so you will see various examples of tools, case studies (mainly open-source systems), empirical validation and formal models. All contributing authors did their
very best to provide a broad overview of the related work, contribute to a comprehensive glossary and a list of acronyms used within the community, and—last but not least—collect a list of books, journals, web-sites, standards and conferences that together represent the community. So reading this book should give you a head start when diving into the field of software evolution.

As such, we hope that this book will become a key reference in the field, providing a basis for the next generation of software evolution researchers.

Who Should Read this Book?

This book is of interest to everyone working in the field of software engineering and wishing to acquire more knowledge on the state-of-the-art in software evolution, software maintenance and re-engineering. In particular, we target this book to researchers, teachers, students and practitioners that need up-to-date information on this very important research field.

So, whether you are a PhD researcher exploring a research topic, a student writing a master’s thesis, a teacher in need of an overview, a practitioner looking for the state-of-the-art, or if you are simply curious about what the field of software evolution has to offer, this should be the book for you.

Why this Book?

Software has become omnipresent and indispensable in our information-based society. Almost all devices, machines and artefacts surrounding us incorporate software to some extent. The numerous organisations, businesses and enterprises we face on a daily basis could not function without software. As such, software is vital to our society and consequently we—the software engineering community—should take up our responsibility to produce reliable software. For a long, long time, reliable software was seen as software “without bugs”. As a result, most of the software engineering research effort has concentrated on preventing, detecting and repairing mistakes in various stages of software development. However, more and more, reliable software has come to mean “easy to adapt”. Indeed today’s global society, with its extreme complexity and diversity imposes constant pressure to change . . . to adapt. Hence all the software that surrounds us is forced to keep pace or is bound to be replaced by something else . . . something new.

Software evolution is the subdomain of the software engineering discipline that investigates ways to adapt software to the ever-changing user requirements and operating environment (i.e., it addresses the How? question). However, software evolution also studies the change process itself, analysing remnants of the software (for instance in version repositories) to extract trends, make predictions or understand the very nature of the software evolution phenomenon itself (i.e., it explores the What and Why? questions). With the recent interest in agile software development, finding good answers for the How? question is necessary. On the other hand, the emergence
of open-source software development with its sheer unlimited access to a wealth of data has provided an extra opportunity to address the *What and Why?* questions in a scientific way. Consequently, research in software evolution has seen a recent boost, and this book provides an up-to-date view on the ideas emerging from our research labs.

**Acknowledgements**

We would like to thank all persons that have contributed to this book, either directly or indirectly. There are many people that we are indebted to:

- The contributors of the chapters of this book;
- The Springer staff (in particular, Ralf Gerstner and Ulrike Stricker);
- Mehdi Jazayeri, who was so kind to write a very nice foreword for this book;
- David Notkin, Michael Godfrey, Václav Rajlich and Anne Keller, who spent their precious time to review this book in its entirety, and provided numerous suggestions for improvement;
- Joris Van Geet, Pieter Van Gorp, Filip Van Rysselberghe, Bart Van Rompaey, Bart Du Bois, Matthias Rieger and Hans Schippers, who provided valuable feedback on several chapters of this book;
- Last but not least, we would like to thank you, reader of this book.

Many of the results that are published in this book have been achieved in the context of research projects or research collaborations. In particular we would like to mention:

- The Scientific Network “Research Links to Explore and Advance Software Evolution” (RELEASE), financed by the *European Science Foundation* (ESF) from July 2002 to December 2005.
- The ongoing ERCIM Working Group on Software Evolution, a network of research institutes from all over the world working on the topic of software evolution, supported by the *European Research Consortium on Informatics and Mathematics* (ERCIM) since December 2004.
- The Interuniversity Attraction Poles Programme (IUAP) on “Modelling, Verification and Evolution of Software” (MOVES), financed by the *Belgian State - Belgian Science Policy* from January 2007 to December 2011.
- The Belgian FRFC project “Research Centre on Structural Software Improvement”, financed by the *Fondation Nationale de Recherche Scientifique* (FNRS - Belgium) from January 2005 to December 2008.
- The Swiss joint research project “Controlling Software Evolution” (COSE), financed by the *Swiss National Science Foundation* from July 2005 to September 2007
- The Swiss joint research project “Multi-dimensional Navigation Spaces for Software Evolution” (EvoSpaces), financed by the *Hasler Foundation* from January 2006 to December 2007.
The German “Bauhaus” Project on Software Architecture, Software Reengineering, and Program Understanding (see www.bauhaus-stuttgart.de)

The European Leg2Net project: “From Legacy Systems to Services in the Net”, supported by Marie Curie Fellowships for the Transfer of Knowledge - Industry Academia Partnership (MTK1-CT-2004-003169).


The Dutch “Reconstructor” project sponsored by the NWO Jacquard programme, project number 638.001.408.

We hope you enjoy reading, and we welcome any comments you may have on the contents, structure or quality of this book.

Mons and Antwerp,汤姆·门斯
September 2007

Serge Demeyer
# Contents

**Foreword**
*Mehdi Jazayeri* ....................................................... VII

1 **Introduction and Roadmap:**
*History and Challenges of Software Evolution*
*Tom Mens* ......................................................... 1

**Part I Understanding and Analysing Software Evolution**

2 **Identifying and Removing Software Clones**
*Rainer Koschke* ....................................................... 15

3 **Analysing Software Repositories to Understand Software Evolution**
*Marco D’Ambros, Harald C. Gall, Michele Lanza, Martin Pinzger* ............ 37

4 **Predicting Bugs from History**
*Thomas Zimmermann, Nachiappan Nagappan, Andreas Zeller* ............... 69

**Part II Reengineering of Legacy Systems**

5 **Object-Oriented Reengineering**
*Serge Demeyer* ....................................................... 91

6 **Migration of Legacy Information Systems**
*Jean-Luc Hainaut, Anthony Cleve, Jean Henrard, Jean-Marc Hick* .......... 105

7 **Architectural Transformations:**
*From Legacy to Three-Tier and Services*
*Reiko Heckel, Rui Correia, Carlos Matos, Mohammad El-Ramly,*
*Georgios Koutsoukos, Luís Andrade* .................................. 139
Part III Novel Trends in Software Evolution

8 On the Interplay Between Software Testing and Evolution and its Effect on Program Comprehension  
Leon Moonen, Arie van Deursen, Andy Zaidman, Magiel Bruntink .............. 173

9 Evolution Issues in Aspect-Oriented Programming  
Kim Mens, Tom Tourwé .................................................. 203

10 Software Architecture Evolution  
Olivier Barais, Anne Françoise Le Meur, Laurence Duchien, Julia Lawall . . . 233

11 Empirical Studies of Open Source Evolution  
Juan Fernandez-Ramil, Angela Lozano, Michel Wermelinger, Andrea Capiluppi .......................................................... 263

Appendices

Resources ................................................................. 291

Glossary of Terms .................................................. 295

List of Acronyms ...................................................... 307

References ............................................................. 309

Index ................................................................. 341
List of Contributors

**Luis Andrade**  
ATX Software  
Rua Saraiva de Carvalho 207C  
1350-300 Lisboa  
Portugal  
luis.andrade@atxsoftware.com

**Olivier Barais**  
Université de Rennes 1  
IRIA/INRIA Triskell project  
Campus de Beaulieu  
35042 Rennes Cédex  
France  
barais@irisa.fr

**Magiel Bruntink**  
CWI  
P.O. Box 94079  
1090 GB, Amsterdam  
and  
Software Engineering Research Group  
Technische Universiteit Delft  
Mekelweg 4, 2628 CD, Delft  
The Netherlands  
Magiel.Bruntink@cwi.nl

**Andrea Capiluppi**  
Department of Computing and Informatics  
Faculty of Technology  
University of Lincoln  
Brayford Pool, Lincoln LN6 7TS  
United Kingdom  
acapiluppi@lincoln.ac.uk

**Anthony Cleve**  
PReCISE Research Centre  
Laboratory of Database Engineering  
University of Namur  
Rue Grandgagnage 21, 5000 Namur  
Belgium  
acl@info.fundp.ac.be

**Rui Correia**  
Department of Computer Science  
University of Leicester  
University Road LE1 7RH  
Leicester  
United Kingdom  
and  
ATX Software  
Rua Saraiva de Carvalho 207C  
1350-300 Lisboa  
Portugal  
rmc20@mcs.le.ac.uk
Marco D’Ambros  
Faculty of Informatics  
University of Lugano  
Via G. Buffi 13, 6904 Lugano  
Switzerland  
marco.dambros@lu.unisi.ch

Serge Demeyer  
Lab on Re-Engineering (LORE)  
Department of Mathematics and Computer Science  
Universiteit Antwerpen  
Middelheimlaan 1, 2020 Antwerpen  
Belgium  
serge.demeyer@ua.ac.be

Laurence Duchien  
Université de Lille 1  
LIFL/INRIA ADAM project  
Cité Scientifique  
59655 Villeneuve d’Ascq Cedex  
France  
duchien@lifl.fr

Mohammad El-Ramly  
Computer Science Department  
Cairo University  
Egypt  
m.elramly@fci-cu.edu.eg

Juan Fernandez-Ramil  
Computing Department and Centre for Research in Computing  
The Open University  
Walton Hall, Milton Keynes MK7 6AA  
United Kingdom  
j.f.ramil@open.ac.uk

Harald Gall  
Software Engineering Group  
Universität Zürich  
Binzmühlstrasse 14  
8050 Zürich  
Switzerland  
gall@ifi.unizh.ch

Jean-Luc Hainaut  
PReCISE Research Centre  
Laboratory of Database Engineering  
University of Namur  
Rue Grandgagnage 21, 5000 Namur  
Belgium  
jlh@info.fundp.ac.be

Reiko Heckel  
Department of Computer Science  
University of Leicester  
University Road LE1 7RH, Leicester  
United Kingdom  
reiko@mcs.le.ac.uk

Jean Henrard  
REVER s.a.  
Boulevard Tirou 130, 6000 Charleroi  
Belgium  
jean.henrard@rever.eu

Jean-Marc Hick  
REVER s.a.  
Boulevard Tirou 130, 6000 Charleroi  
Belgium  
jean-marc.hick@rever.eu

Rainer Koschke  
Fachbereich 03  
Universität Bremen  
Postfach 33 04 40, 28334 Bremen  
Germany  
koschke@informatik.uni-bremen.de

Georgios Koutsoukos  
Department of Computer Science  
University of Leicester  
University Road LE1 7RH, Leicester  
United Kingdom  
georgios.koutsoukos@atxsoftware.com

Michele Lanza  
Faculty of Informatics  
University of Lugano  
Via G. Buffi 13, 6904 Lugano  
Switzerland  
michele.lanza@unisi.ch
Julia Lawall
DIKU
University of Copenhagen
2100 Copenhagen
Denmark
julia@diku.dk

Anne Françoise Le Meur
Université de Lille 1
LIFL/INRIA ADAM project
Cité Scientifique
59655 Villeneuve d’Ascq Cedex
France
lemeur@lifl.fr

Angela Lozano
Computing Department
and Centre for Research in Computing
The Open University
Walton Hall, Milton Keynes MK7 6AA
United Kingdom
a.lozano-rodriguez@open.ac.uk

Carlos Matos
Department of Computer Science
University of Leicester
University Road LE1 7RH, Leicester
United Kingdom
and
ATX Software
Rua Saraiva de Carvalho 207C
1350-300 Lisboa
Portugal
cmm22@mcs.le.ac.uk

Tom Mens
Institut d’Informatique
Université de Mons-Hainaut
Av. du Champ de Mars 6, 7000 Mons
Belgium
tom.mens@umh.ac.be

Leon Moonen
Software Engineering Research Group
Technische Universiteit Delft
Mekelweg 4, 2628 CD, Delft
The Netherlands
leon.moonen@computer.org

Nachiappan Nagappan
Microsoft Research
Redmond, Washington
USA
nachin@microsoft.com

Martin Pinzger
Department of Informatics
Universität Zürich
Winterthurerstrasse 190, 8057 Zürich
Switzerland
pinzger@ifi.unizh.ch

Tom Tourwé
Department of Software Engineering and Technology
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven
The Netherlands
t.tourwe@tue.nl

Arie van Deursen
Software Engineering Research Group
Technische Universiteit Delft
Mekelweg 4, 2628 CD, Delft
and
CWI
P.O. Box 94079, 1090 GB, Amsterdam
The Netherlands
Arie.van.Deursen@cwi.nl
Michel Wermelinger
Computing Department and Centre for Research in Computing
The Open University
Walton Hall, Milton Keynes MK7 6AA
United Kingdom
m.a.wermelinger@open.ac.uk

Andreas Zeller
Dept. of Informatics
Saarland University
Postfach 15 11 50, 66041 Saarbrücken
Germany
zeller@cs.uni-sb.de

Andy Zaidman
Software Engineering Research Group
Technische Universiteit Delft
Mekelweg 4, 2628 CD, Delft
The Netherlands
A.E.Zaidman@tudelft.nl

Thomas Zimmermann
Department of Computer Science
University of Calgary
2500 University Drive NW Calgary
Alberta, T2N 1N4 Canada
tz@acm.org