

Preface

In 2008 the predecessor of this book, entitled “Software Evolution” [592] was published by Springer, presenting the research results of a number of researchers working on different aspects of software evolution. Since then, the software evolution research has explored new domains such as the study of socio-technical aspects and collaboration between different individuals contributing to a software system, the use of search-based techniques and metaheuristics, the mining of unstructured software repositories, techniques to cope with the evolution of software requirements, and dealing with the dynamic adaptation of software systems at runtime. Moreover, while the research covered in the book pertained largely to evolution of individual software projects, more and more attention is currently being paid to the evolution of collections of inter-related and inter-dependent software projects, be it under the form of web systems, software product families, software ecosystems or systems of systems. We therefore felt that it was time to “release” a new book that complements its predecessor by addressing a new series of highly relevant and active research topics in the field of software evolution. As can be seen in Table 1, both books together aim to cover most of the active topics in software evolution research today. We are very grateful to the authors of contributed chapters for sharing this feeling and having accepted to join us in this endeavour.

Where does software evolution fit in a computing curriculum?

The Computer Science Curricula body of knowledge aims to provide international guidelines for computing curricula in undergraduate programs. The CS2013 version [446] has been created by a joint task force of the ACM and the IEEE Computer Society, and constitutes a major revision of the CS2008 and CS2001 versions. This body of knowledge is organized into 19 knowledge areas, software engineering being one of the essential ones (it accounts for 8.8% of the total core hours, namely 27 out of 307 hours).

Table 1: Comparison of, and partial overlap between, the software evolution topics addressed in chapters of [592] (second column) and this book (third column), respectively. Chapters written in **boldface** have the indicated topic as their principal theme.

Topic addressed	Software Evolution 2008 [592]	Evolving Software Systems 2014
software cloning, code duplication defect, failure and bug prediction re-engineering, restructuring and refactoring	Ch. 2 [475] Ch. 4 [954] Ch. 5 [237], Ch. 1 [590], Ch. 2 [475], Ch. 8 [617] Ch. 6 [358] Ch. 6 [358], Ch. 7 [378] Ch. 7 [378] Ch. 8 [617] Ch. 9 [589], Ch. 10 [70] Ch. 10 [70], Ch. 7 [378] Ch. 1 [590], Ch. 8 [617]	
database evolution migration, legacy systems service-oriented architectures testing aspect-oriented software development software architecture evolution reverse engineering, program understanding, program comprehension incremental change, impact analysis, change propagation evolution process software visualisation component-based software development	Ch. 1 [590] Ch. 1 [590] Ch. 3 [216] Ch. 10 [70]	
open source software software repository mining	Ch. 11 [291] Ch. 3 [216], Ch. 4 [954], Ch. 11 [291] Ch. 6 [358], Ch. 7 [378], Ch. 10 [70] Ch. 1 [590] Ch. 4 [954]	Chapter 10 Chapter 5
software transformation, model transformation, graph transformation model evolution, metamodel evolution software measurement, software quality		Chapter 2 Chapter 2 Chapter 3
software requirements search-based software engineering socio-technical networks, Web 2.0 web-based systems dynamic adaptation, runtime evolution software product line engineering software ecosystems, biological evolution		Chapter 1 Chapter 4 Chapter 6 Chapter 7 Chapter 7.6 Chapter 9 Chapter 10

Within the software engineering knowledge area, 10 themes have been identified as being the most important to be taught, and software evolution is one of these themes. In particular, the CS2013 body of knowledge recommends the following learning outcomes for this theme:

- Identify the principal issues associated with software evolution and explain their impact on the software life cycle.
- Discuss the challenges of evolving systems in a changing environment.
- Discuss the advantages and disadvantages of software reuse.
- Estimate the impact of a change request to an existing product of medium size.
- Identify weaknesses in a given design, and remove them through refactoring.

- Outline the process of regression testing and its role in release management.

To achieve these outcomes, it is suggested that at least the following topics be taught: software development in the context of large, pre-existing code bases (including software change, concerns and concern location, refactoring), software evolution, software reuse, software reengineering, and software maintainability characteristics.

What is this book about?

This book goes well beyond what is expected to be part of an undergraduate software evolution course. Instead, the book is a coherent collection of chapters written by renowned researchers in the field of software evolution, each chapter presenting the state of the art in a particular topic, as well as the current research, available tool support and remaining challenges. All chapters have been peer reviewed by at least five experts in the field.

We did not aim to cover all research topics in software evolution. Given the wealth of information in the field that would be an impossible task. Moreover, a number of important and actively studied software evolution topics have been profoundly covered by the predecessor of this book [592]. Therefore, we have primarily focused on new domains currently being explored most actively by software evolution researchers.

This book is divided into four parts. Part I, *Evolving Software Artefacts*, focuses on specific types of artefacts that are used during software evolution. Chapter 1 focuses on the evolution of software requirements, and its impact on the architecture and implementation of a software system. Chapter 2 focuses on the coupled evolution of software models and their metamodels. Chapter 3 explores the use of maintainability models for assessing the quality of evolving software.

Part II of the book focuses on techniques used by software evolution researchers. Chapter 4 explores the use of search-based techniques and metaheuristics to address untractable software evolution activities. Chapter 5 explores how to mine unstructured data stored in repositories containing historical information that may be relevant to understand the evolution of a software system. Chapter 6 discusses how socio-technical information, available thanks to Web 2.0 technology, can be leveraged to help developers in performing evolution activities.

Part III focuses on how specific types of software systems, or collections of software systems, evolve. Chapter 7 looks at the evolution of web systems. Chapter 7.6 explores the runtime evolution of adaptive software systems. Chapter 9 overviews the product line evolution of families of software products. Chapter 10 focuses on the evolution of software ecosystems and how we can learn from the evolution from natural, biological ecosystems.

A large appendix complements this work. It starts by outlining the emerging and future directions in software evolution research. It also contains a list of acronyms used in the different chapters, a glossary of important terms used in each

chapter, pointers to additional resources that may be useful to the reader (books, journals, standards and major scientific events in the domain of software evolution), and datasets.

Who Should Read this Book?

This book is intended for all those interested in software engineering, and more particularly, software maintenance and evolution. Researchers as well as software practitioners will find in the contributed chapters an overview of the most recent research results covering a broad spectrum of software evolution topics.

While this book has not been written as a classical textbook, we believe that it can be used in teaching advanced (graduate or postgraduate) software engineering courses on e.g., software evolution, requirements engineering, model-driven software development or social informatics. This is exactly what we, book editors, intend to do in our own advanced software engineering and evolution lectures as well.

For researchers, the book and its contributed chapters can be used as a starting point for exploring the addressed topics in more depth, by gaining an understanding in the remaining research challenges and by diving into the wealth of literature referenced in each chapter. For ease of reference, and to avoid duplication, all bibliographic references are bundled together at the end of the book rather than on a per chapter basis, with back references to the chapters where the references have been cited. There is also an index of terms that makes it easy to find back the particular sections or chapters in which specific topics have been addressed.

Practitioners might be interested in numerous tools discussed in the forthcoming chapters. Such tools as Software QUALity Enhancement (SQUALE, Chapter 3) have been successfully applied in the industry, some others are still waiting to be discovered and used by practitioners. We hope that our book can help practitioners in this endeavour.

Happy reading!

Mons, Eindhoven, Namur,
November
2013

Tom Mens
Alexander Serebrenik
Anthony Cleve



<http://www.springer.com/978-3-642-45397-7>

Evolving Software Systems

Mens, T.; Serebrenik, A.; Cleve, A. (Eds.)

2014, XXIII, 404 p., Hardcover

ISBN: 978-3-642-45397-7