Lecture Notes in Computer Science 5413

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Andrea De Lucia Filomena Ferrucci (Eds.)

Software Engineering

International Summer Schools
ISSSE 2006-2008, Salerno, Italy
Revised Tutorial Lectures

Springer

Volume Editors

Andrea De Lucia
Filomena Ferrucci
Università di Salerno
Dipartimento di Matematica e Informatica
Via Ponte Don Melillo, 84084, Fisciano, SA, Italy
E-mail: {adelucia,fferrucci}@unisa.it

Library of Congress Control Number: Applied for

CR Subject Classification (1998): D.2, D.1, F.3, K.6.3

LNCS Sublibrary: SL 2 – Programming and Software Engineering

ISSN 0302-9743
ISBN-10 3-540-95887-8 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-95887-1 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12605804 06/3180 5 4 3 2 1 0

Preface

Software Engineering is widely recognized as one of the most exciting, stimulating, and profitable research areas with significant practical impacts on the software industry. Thus, training the future generations of software engineering researchers and bridging the gap between academia and industry are especially important. The International Summer School on Software Engineering (ISSSE) aims to contribute both to training future researchers and to facilitating knowledge exchange between academia and industry. Beginning in 2003, it has become an annual meeting point that is now in its fifth edition (2003, 2005, 2006, 2007, and 2008).

ISSSE is intended for PhD students, university researchers, and professionals from industry. Attracting more than 60 participants each year, the program of the school includes six state-of-the-art tutorials given by internationally recognized research leaders on very relevant topics for the scientific community. Each tutorial provides a general introduction to the chosen topic, while also covering the most important contributions in depth and identifying the main research challenges for software engineers. The focus is on methods, techniques, and tools; in some cases theory is required to provide a solid basis.

The format of the school aims at creating extensive discussion forums between lecturers and industrial and academic attendees. Besides traditional tutorials, lab sessions are also included in the program together with student talks and tool demos to further stimulate the interaction. The school is held on the campus of the University of Salerno and benefits from the facilities of the university (e.g., laboratories) and from its location, very close to some of the most beautiful and historical places of Italy, such as the Amalfi Coast, Capri, Pompei, and Paestum.

This volume collects chapters originating from some tutorial lectures given at the last three editions of the school (2006, 2007, and 2008) and aims to provide a contribution on some of the latest findings in the field of software engineering. Several interesting topics are covered including software adaptability and dependability, autonomic computing, usability in requirements engineering, testing of service-oriented architectures, reverse engineering, collaborative development, web cost estimation and productivity assessment, empirical software engineering, and experience factory. The volume is organized in three parts, collecting chapters focused on software requirements and design, software testing and reverse engineering, and management.

In the first chapter, Paola Inverardi and Massimo Tivoli take into account two of the key requirements that software in the near ubiquitous future (Softure) will need to satisfy, namely, adaptation and dependability. Software systems need to be adaptable according to the context changes determined by the diversity of computing platforms where software systems are to get deployed and the different execution environments where they have to operate. Moreover, the pervasiveness of software systems and the highly dynamic nature of service provision make Softure dependability more complex. Thus, ensuring the dependability of self-adaptive systems poses numerous challenges

to software engineers. In the chapter, the authors analyze some of these challenges and describe possible solutions.

In the second chapter, Hausi A. Müller et al. focus on autonomic computing, an approach to building self-managing computing systems with the aim of addressing the management complexity of dynamic computing systems through technology simplification and automation. Autonomic computing is inspired by the autonomic nervous system of the human body that controls important bodily functions without any conscious intervention. It includes a broad range of technologies, models, architecture patterns, standards, and processes and is based on some key aspects, such as feedback control, adaptation, and self-management. In particular, at the core of an autonomic system are the feedback control loops that constantly monitor the system and its environment looking for events to handle. The authors address the problem of designing such highly dynamical systems, arguing that both a software architecture perspective and a feedback control perspective need to be taken into account. Moreover, they show how autonomic computing technology can solve continuous evolution problems of software-intensive systems. Finally, they illustrate some lessons learned and outline some challenges for the future.

In chap. 3, Natalia Juristo focuses on usability, which represents one of the most important quality factors of a software product but it is often insufficient in most software systems. In particular, the author proposes an approach to build software with higher usability. According to such an approach usability features need to be considered from a functional viewpoint at requirements stage, because as some empirical studies show, software design and usability are related. To address the difficulties of usability features elicitation and specification, the author proposes specific guidelines that support face-to-face communication among the different stakeholders and lead software developers to ask the appropriate questions so as to capture usability requirements information and cut down ambiguous usability details as early as possible.

Chapters 4 and 5 focus on software testing of service-oriented architectures. In the life-cycle of any software system, testing is a crucial activity to ensure adequate software dependability but it is also one of the most expensive. This is especially true with service-oriented architectures. Indeed, this emerging paradigm for distributed computing is radically changing the way in which software applications are developed and posing several new challenges for software testing. These mainly originate from the high flexibility and dynamic nature of service-oriented architectures and the use of some unique features such as service discovery and composition, ultra-late binding, automated negotiation, autonomic system reconfiguration, and so on. In chap. 4, Gerardo Canfora and Massimiliano Di Penta provide a broad survey of the recent research carried out on the topics of testing of service-oriented architectures. The authors analyze several challenges from the viewpoints of different stakeholders and present solutions for different levels of testing (unit, integration, and regression testing) and for both functional and non-functional testing. The authors conclude the chapter by exploring ways to improve the testability of service-oriented architectures. In chap. 5, Antonia Bertolino et al. deepen the discussion to challenges and solutions concerning testing activities by focusing on the validation framework developed in the European Project PLASTIC. In the framework, different techniques can be combined for the verification of functional and non-functional properties for both development time

testing and service live usage monitoring. The authors also describe some techniques and tools that fit within the proposed framework.

Chapter 6 focuses on software architecture reconstruction, i.e., the kind of reverse engineering where architectural information is reconstructed for an existing system. Software architectures are able to provide a global understanding of a software system that is essential to effectively carry out many maintenance and migration tasks. Nevertheless, frequently software architecture is not sufficiently described, or it is outdated and inappropriate for the task at hand. Thus, it is necessary to reconstruct it gathering information from several sources (source code, system's execution, available documentation, stakeholder interviews, and domain knowledge), applying appropriate abstraction techniques, and suitably presenting the obtained information. In this chapter, Rainer Koschke introduces a conceptual framework for architecture reconstruction that represents a combination of common patterns and best practices reported in the reverse engineering literature. The current state of the art of techniques and methods for software architecture reconstruction is also summarized here. Finally, the author discusses a number of research challenges that should be tackled in the future.

In chap. 7, Filippo Lanubile focuses on the globalization of software development that represents a trend in today's software industry. Indeed, developing software using teams geographically distributed in different sites can provide an important competitive advantage for software companies but at the same time it presents challenges that affect all aspects of a project. The author focuses on the collaboration issues that arise from the negative effects of distance, illustrates a taxonomy of software engineering tools that support distributed projects, and presents several collaborative development environments. Computer-mediated communication theories that can be useful to address the development of more effective tools supporting collaboration in distributed software development are also discussed. Finally, the author summarizes a family of empirical studies which have been carried out to build an evidence-based model of task–technology fit for distributed requirements engineering.

Web applications are constantly increasing both in complexity and number of features. As a result, Web effort estimation and productivity analysis are becoming crucial activities for software companies to develop projects that are finished on time and within budget. Indeed, they are essential elements for project managers to suitably plan the development activities, allocate resources adequately, and control costs and schedule. In chap. 8 Emilia Mendes introduces the main concepts and techniques for Web effort estimation and reports on a case study where the process for the construction and validation of a Web effort estimation model is illustrated step-by-step. Moreover, the author introduces the main concepts related to Web productivity measurement and benchmarking and describes a case study on productivity benchmarking.

In the final chapter, Giuseppe Visaggio addresses the relevant challenge of transferring research results in production processes and exchanging results between researchers and enterprises. Indeed, competition requires continuous innovation of processes and products and a critical issue for organizations is how to speed up knowledge creation and sharing. Knowledge management aims to provide an answer to this issue by managing the processes of knowledge creation, storage, and sharing. The author presents a widely known model of knowledge management, namely, the experience factory, that collects empirical knowledge in an experience package with

the aim of mitigating the consequences of knowledge loss due to staff turnover, improving products and processes, and assuring near-the-job learning. Based on previous technology transfer experiences, the author proposes a model for systematically structuring the content of a package and allowing for an incremental and cooperative production of packages. A platform that supports the collection and distribution of knowledge-experience packages is also described.

We wish to conclude by expressing our gratitude to the many people who supported the publication of this volume with their time and energy. First of all, we wish to thank the lecturers and all the authors for their valuable contribution. We also gratefully acknowledge the Scientific Committee Members, for their work and for promoting the International Summer School on Software Engineering. Thanks are due to our department (Dipartimento di Matematica e Informatica, Università di Salerno) for the administrative and organizational support we received every day. We are also grateful to Vincenzo Deufemia, Sergio Di Martino, Fausto Fasano, Rita Francese, Carmine Gravino, Rocco Oliveto, Ignazio Passero, Michele Risi, and Giuseppe Scanniello, who were of great help in organizing the school. Finally, we want to thank Springer for providing us with the opportunity to publish this volume.

We hope you will enjoy reading the chapters and find them relevant and fruitful for your work. We also hope that the tackled topics will encourage your research in the software engineering field and your participation in the International Summer School on Software Engineering.

October 2008

Andrea De Lucia
Filomena Ferrucci

Table of Contents

Software Requirements and Design

Software Testing and Reverse Engineering

Management