Undergraduate Topics in Computer Science

Orit Hazzan and Yael Dubinsky

Agile Software Engineering

Springer

Orit Hazzan, BSc, MSc, PhD, MBA
Department of Education in Technology and Science
Technion – Israel Institute of Technology
Haifa, Israel

Yael Dubinsky, BSc, MSc, PhD
Department of Computer Science
Technion – Israel Institute of Technology
Haifa, Israel

Undergraduate Topics in Computer Science ISSN 1863-7310
ISBN: 978-1-84800-198-5 e-ISBN: 978-1-84800-199-2
DOI: 10.1007/978-1-84800-199-2

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

Library of Congress Control Number: 2008933449

Printed on acid-free paper

Springer Science+Business Media
springer.com

Orit Hazzan dedicates this book to her agile family—
Shimon, Yael and Dan

Yael Dubinsky dedicates this book to her parents, Helen and Yaakov,
life partner David, and lovely children Tal and Or

Preface

Overview and Goals

The agile approach for software development has been applied more and more extensively since the mid nineties of the 20th century. Though there are only about ten years of accumulated experience using the agile approach, it is currently conceived as one of the mainstream approaches for software development.

This book presents a complete software engineering course from the agile angle. Our intention is to present the agile approach in a holistic and comprehensive learning environment that fits both industry and academia and inspires the spirit of agile software development.

Agile software engineering is reviewed in this book through the following three perspectives:

- The **H**uman perspective, which includes cognitive and social aspects, and refers to learning and interpersonal processes between teammates, customers, and management.
- The **O**rganizational perspective, which includes managerial and cultural aspects, and refers to software project management and control.
- The **T**echnological perspective, which includes practical and technical aspects, and refers to design, testing, and coding, as well as to integration, delivery, and maintenance of software products.

Specifically, we explain and analyze how the explicit attention that agile software development gives these perspectives and their interconnections, helps it cope with the challenges of software projects. This multifaceted perspective on software development processes is reflected in this book, among other ways, by the chapter titles, which specify dimensions of software development projects such as quality, time, abstraction, and management, rather than specific project stages, phases, or practices.

HOT

H**O**T

HO**T**

To share with the readers this multifaceted perspective, we use the **H**uman, **O**rganizational, and **T**echnical (HOT) scale for software development approaches. For example, when we refer to teamwork or abstraction levels, we emphasize the Human perspective; when software management issues are addressed, the Organizational perspective is emphasized; similarly, when the actual performance of test-driven development is described, the Technological aspect is highlighted. When the HOT? sign appears, the readers are invited to suggest their own HOT perspective.

Agile software development values these three perspectives. Therefore, in many cases, more than one perspective is illuminated by the agile approach with respect to a specific topic. Yet even when more than one perspective is significant with respect to a specific topic, we discuss from time to time only one or two main perspective(s), and the readers are invited to complete the picture.

The book is based on the authors' comprehensive experience of teaching and implementing agile software development over the past six years. A course on agile software engineering has been shaped during these years, in an iterative process that was accompanied by an ongoing research project. This course is presented in this book. In parallel to the course creation and shaping process, the agile approach has emerged and spread, becoming one of the worldwide mainstream approaches for software project management.

Organization and Features

This textbook guides a fourteen-week/session course on software engineering from the agile perspective and can be used on a weekly basis. It is intended for all who practice, research, teach, and learn software development both in academia and industry. It discusses how agile teams live and function in software development environments, how they achieve their goals, and how they act professionally in their environments. Specifically, the themes presented in the book, such as teamwork, time, quality, learning, trust, and culture, are reviewed from human, organizational, and technological perspectives, at the individual, team, and organizational levels, and are illustrated with case studies taken from industry and academia.

The fourteen chapters of the book are organized in three iterations. This structure enables us to revisit the various subjects several times during the course,

Table 1. Book structure

Iteration	Chapter #	Topic
I	1	Introduction to Agile Software Development
	2	Teamwork
	3	Customers and Users
	4	Time
	5	Measures
	6	Quality
	7	Learning
II	8	Abstraction
	9	Trust
	10	Globalization
	11	Reflection
III	12	Change
	13	Leadership
	14	Delivery and Cyclicality

as well as to guide the development of a one-release software product. Table 1 presents the book's structure book and the topic of each chapter.

Each chapter includes a theoretical approach to a specific topic, a section that refers to the given topic in learning environments, and a variety of questions and tasks for further elaboration.

The Academic Community

This book on agile software engineering can be used by instructors, academic coaches, and students as a textbook during a fourteen-week semester, either for the commonly titled "Introduction to Software Engineering" course or the "Software Engineering Methods" course.

The course is based on two main components that progress in parallel and are closely correlated with each other. The first component is theoretical and can be used in the lecture hall or the class; the second is software project development guided by the agile approach that takes place in a physical learning environment that we call a studio or lab.

This book is written for the entire course community—students, instructors, and academic coaches. Students are the learners who become familiar with the agile approach both from a theoretical perspective (in the lectures) and from a practical perspective (in the studio). Instructors are the teachers of the course's theoretical ideas, who usually teach in a class or in a lecture hall; yet, interactive teaching and active learning can be facilitated in this setting as well. The academic coaches are the practitioners who guide the software project development in the studio (we elaborate on this role in Chapter 1, Introduction to Agile Software Engineering).

The positive results of agile software projects, as elaborated throughout the various chapters of the book, are not the only motive for this course, which presents the field of software engineering from the agile perspective. There are three additional characteristics of the course, which are especially relevant when it is taught in academia.

First, the agile approach was developed by practitioners working in the software industry, and has become mainstream in that industry. Therefore, it makes sense to articulate its nature and main concepts to prospective software engineers in the framework of a course that deals with software engineering.

Second, teaching a software engineering course within the framework of agile software development emphasizes a comprehensive image of the field. This is because agile software development explicitly addresses human, organizational, and technological aspects of the software development process with respect to all players participating in that process. Thus, the agile approach serves as an opportunity to draw this comprehensive and complex picture of the field.

Third, according to the Software Engineering 2004 Curriculum, developed by the IEEE Computer Society and the Association for Computing Machinery Joint Task Force (see http://sites.computer.org/ccse/SE2004Volume.pdf), software engineering students should acquire additional skills beyond the technical and scientific ones. One illustrative example is teamwork-related skills. Since teamwork is one of the basic ideas of agile software development, it is only natural to integrate teamwork-oriented skills in the teaching and learning process of software engineering from the agile perspective. Furthermore, since it is natural to teach agile software development in a teamwork-oriented environment, there is no need to introduce the topic of teamwork artificially; rather, a teamwork-based learning environment can be used to teach this topic. This element is emphasized mainly, but not only, in the studio element of the course.

Suggested Uses in an Academic Environment

Each chapter presents a full week of the course: two weekly lecture hours and a four-hour weekly studio meeting. The first part of each chapter includes contents suitable to be presented in the lecture. This part usually presents material beyond what it is possible to teach in a two-hour lecture. Therefore, it is advisable not to try to deliver all the content in two hours; rather, we suggest selecting from each chapter the most relevant topics to be discussed with each particular class of students. It is also advisable to encourage in the lectures some active learning elements, as is suggested in the various chapters. The second part of each chapter addresses the teaching and learning of the chapter topic. It presents teaching and learning principles and the activities conducted in the studio each week.

As preparation for the next week's lectures and studio meeting, instructors and academic coaches can ask the students to read the relevant chapter and to work on selected activities presented throughout the body of each chapter. The students' preparation for the lecture will also partially solve the time limitation problem of addressing all the ideas presented.

Finally, though the book presents a full fourteen-week semester course, which consists of two weekly lecture hours and four-hour weekly studio meetings, it is possible to teach only one component of the course. The material provided in this book enables each instructor/academic coach to make the needed adjustments.

The Industrial Community

Since agile development has become one of the mainstream approaches for managing software projects, more and more software organizations of different sizes and types ask themselves whether the agile approach fits them. Even when it is found that agile software development is relevant for a given organization, questions such as the following are usually asked: How can we manage a transition to the agile software development process? How can our organization cope with the changes required for such a transition? How can we teach agile software development to all the software practitioners and all the other software project's stakeholders?

This book, when used in an industrial setting, aims to answer these and other relevant questions which software organizations face when dealing with the transition to agile software development. For example, in Chapter 12, Change, we discuss how to initiate a transition process to agile software development in an organization. When the organization has already transitioned to agile software development, the book can also be used for answering questions related to the actual implementation of agile software development in the organization. For example, in Chapter 2, Teamwork, we discuss how teams can be formed to exploit their potential, to avoid conflicts, and to solve dilemmas.

Suggested Uses in an Industrial Environment

This book can be used in industrial settings by coaches of software teams, software team leaders, and facilitators of agile software development workshops, both for the teaching and learning of agile software development, as well as for its implementation. The book can also be used by interested software practitioners who are not necessarily within a formal teaching framework.

We propose two ways to use the book in industrial environments.

First, the book can be used for a course which is based on 14 sessions. This course format fits for organizations that wish to expand their members' professional knowledge by becoming familiar with agile software development, without necessarily implementing the agile approach. If the course also contains the development of a software project using the agile approach, which in academia takes place in the studio, a new software system should be developed for learning process purposes, with respect to which the different activities are facilitated. The development of a new software project should be undertaken whether the course is taught to a real team or to a group of people from different teams or organization. In the case of a real team, the development of another project than the team's real project will enable the team not to confuse their current work habits with agile practices.

Second, for organizations which wish to start implementing agile software development right away or in the near future, we suggest that the agile approach be taught first in a short format of a two-day workshop to a team that has been carefully selected to start the transition to agile software development within the organization. Chapter 12, Change, elaborates on such a transition process, explains the motivation and rationale for this intense workshop format, and outlines the workshop schedule. After the team members have participated in that workshop, and when the team starts implementing agile software development with its real project, the book can be used for clarifications and elaborations.

In both cases, as well as in other learning environments in industry, the teaching and learning principles presented in the book can naturally be applied.

Acknowledgments

We would like to thank all the practitioners, researchers, students, and mangers, both in academia and in the software industry, who during the past six years shared with us their professional knowledge, experience, thoughts, and feelings with respect to agile software development. They all contributed to our understanding of the nature of agile software engineering and fostered our shaping of the approach presented in this book.

Contents