Preface

Agile methods drastically alter the software development processes. Agile software processes, such as extreme programming (XP), Scrum, etc., rely on best practices that are considered to improve software development quality. It can be said that best practices aim to induce software quality assurance (SQA) into the project at hand. Proponents of agile methods claim that because of the very nature of such methods, quality in agile software projects should be a natural outcome of the applied method. As a consequence, agile software development quality assurance (ASDQA) is hoped/expected/supposed to be more or less embedded in the agile software processes, while SQA practices are integrated across the entire life-cycle development, from requirements through the final release. Thus, agile methods introduce a different perspective on QA in software development.

Agile practices are expected to handle unstable and volatile requirements throughout the development lifecycle, to deliver software with fewer defects and errors, in shorter timeframes, and under predefined budget constraints. The iterative and incremental way of development allows both customer requirements revision mechanisms and customer active participation in the decision-making process. Customer participation provides the needed feedback mechanism, ensuring customer perceived satisfaction for the final product. It is also known that agile methods make the key business users a very strong partner in assuring quality. Rather than completely leaving quality to the professionals, agile projects make these key users responsible for ensuring that the application is fit for purpose. Agile development embraces test driven development and test first design, both coming from the arena of good practices, introducing them into mainstream development, and minimizing errors and defects of the final product. Some other practises, such as simple planning and designing, pair programming, short iteration cycles, small releases, continuous integrations, common code ownership, and metaphor potentially reinforce quality assurance.

It is interesting to note that the previously mentioned practices cover and support, to a significant extent, total quality management (TQM) (see Crosby, 1979; Deming, 1986; Feigenbaum, 1961, 1991; Ishikawa, 1985; Juran & Gryna, 1970, all referenced in Chapter II). We remind the reader that a TQM system comprises four key common elements: (1) customer focus, (2) process improvement, (3) human side of quality, and (4) measurement and analysis. Agile methods deal in one way or another with all four elements. Many reports support and evangelize the advantages of agile methods with respect to quality assurance, even if the term "quality assurance" is avoided as coming from traditional, bureaucratic development.

Is it so? For example, is it the case that agile methods assure quality by default, and software managers/developers need not be concerned with quality issues, such as quality planning, quality audits,

or quality reports? Proponents of agile methods must provide convincing answers to questions such as "What is the quality of the software produced?" or "Which hard/soft evidence supports the superiority of agile quality?" There has been little published work that focuses on such agile software development quality issues. In particular, there is a literature gap in providing a critical view of agile quality, pinpointing areas where agile methods are strong, but also areas that need improvement.

OVERALL OBJECTIVE OF THE BOOK

This book pursues an ambitious goal: it attempts to provide answers to the questions and issues previously raised. It provides original academic work and experience reports from industry related to agile software development quality assurance. Its mission is to describe fundamentals of ASDQA theory and provide concrete results from agile software development organizations. To understand how quality is or should be handled, the whole development process must be analyzed, measured, and validated from the quality point of view, as it is claimed to be the rule when traditional methods are employed. It is precisely from the quality point of view that the book looks at agile methods. The area is wide and entails many facets that the book attempts to clarify, including:

- Differences and similarities between the traditional quality assurance procedures and ASDQA.
- Identification and evaluation of quality metrics in agile software development.
- Reports on the state of the art regarding quality achievements in agile methods.
- Investigation on how practices and tools affect the quality in agile software development.
- Human issues in ASDQA.
- Education in ASDQA concepts and techniques.

Book chapters provide theoretical discussion on ASDQA issues and/or results and lessons from practical ASDQA application. Eventually, the book is expected to provide successful quality management tips that can help participants in the agile software development process avoid risks and project failures that are frequently encountered in traditional software projects. Because such task is extremely difficult, given the variety of agile methods, the relatively limited time they have been exercised and the scattered, often vague, information regarding agile quality from the field, this book could only be edited, and not be written by a small authors' group.

The book takes the form of a collection of edited chapters. Authors of the chapters cover all kinds of activities related to agile methods: they are academicians, practitioners, consultants, all involved heavily in practicing, researching, and teaching of agile methods. Authors come from almost all over the world (North America, Europe, Asia, Africa) and are employed by all kinds of organizations involved in agile development (universities, research institutes, small or large agile development/consulting companies).

ORGANIZATION OF THE BOOK

This book is made up of 12 chapters, organized in four sections. Section titles are the following:

Section I: Introduction: Agile Methods and Quality
Section II: Quality within Agile Development

Section III: Quality within Agile Process Management
Section IV: Agile Methods and Quality: Field Experience

Section I: Introduction: Agile Methods and Quality provides the framework for the rest of the book. It is particularly useful for readers not familiar with all aspects of agile methods. It reviews agile methods and compares them with traditional approaches. Section I starts posing questions about the quality achieved and potential problems with agile methods today. It also starts to propose solutions for certain identified issues.

Section II: Quality within Agile Development examines how quality is pursued throughout software development. It gives a flavour of how developers achieve quality in an agile fashion. Chapters in this section review quality assurance when specifying requirements, when handling defects, and when user interfaces are designed and implemented.

Section III: Quality within Agile Process Management examines how quality is pursued throughout the handling of agile software processes. This section deals with activities that run parallel to development or prepare the development teams for effective work. It gives a flavour of how managers achieve quality in an agile fashion. Two chapters in this Section review quality assurance when managing agile software configurations and when agile people are managed. Finally, a critical theme for the future is addressed, namely the education of next generations of agile developers and managers in ASDQA issues.

Section IV: Agile Methods and Quality: Field Experience provides feedback from agile method application. Although all chapters up to now try to capture experiences from agile projects and to incorporate them in theoretical frameworks, chapters of this section come right from agile companies. Interestingly, two of the Chapters come from quite large companies, signalling the expansion of agile methods into the realm of traditional software development. Chapters provide invaluable information about agile project management, quality measurement, test driven development and, finally, lessons learned from ASDQA real world application.

A brief description of each chapter follows. Chapters are organized according to the sections they belong.

Section I: Introduction: Agile Methods and Quality

Chapter I: Agile Software Methods: State-of-the-Art

In Chapter I, Ernest Mnkandla and Barry Dwolatzky (South Africa) analyze and define agile methodologies of software development. They do so by taking a software quality assurance perspective. The chapter starts by defining agile methodologies from three perspectives: a theoretical definition, a functional definition, and a contextualized definition. Next, a brief review of some of the traditional understandings of quality assurance is given, and the author proceeds with certain innovations that agility has added to the world of quality. Doing so, the text provides an understanding of the state-of-the-art in agile methodologies and quality, along with expectations for the future in this field. An analysis framework is used for objectively analyzing and comparing agile methodologies. The framework is illustrated by applying it to three specific agile methodologies.

Chapter II: Agile Quality or Depth of Reasoning? Applicability vs. Suitability with Respect to Stakeholders' Needs

In Chapter II, Eleni Berki (Finland), Kerstin Siakas (Greece), and Elli Georgiadou (UK) provide an in-depth discussion and analysis of the quality characteristics of the agile information systems develop-

ment process. They question ASDQA by exposing concerns regarding the applicability and suitability of agile methods in different organisational and national cultures. They argue based on recent literature reviews and published reports on the state-of-the-art in agile Methodologies. A unique feature of this chapter is that its authors draw their experience from different European countries (Denmark, England, Finland, Greece) with diverse academic and work values, and information systems development (ISD) industrial practices based on different principles. They relate and compare traditional, agile**,** managed, and measured ISD processes, they explore human dynamics that affect success and consensus acceptance of a software system and propose a critical framework for reflecting on the suitability and applicability of agile methods in the development and management of quality software systems. To achieve this, the authors examine the different European perceptions of quality in the agile paradigm and compare and contrast them to the quality perceptions in the established ISD methodological paradigms.

Chapter III: What's Wrong with Agile Methods? Some Principles and Values to Encourage Quantification

In Chapter III, Tom Gilb (Norway) proposes the quantification of agile processes to reinforce ASDQA. He claims that agile methods could benefit from using a more quantified approach across the entire implementation process (that is, throughout development, production, and delivery). He discusses such things as quantification of the requirements, design estimation, and measurement of the delivered results. He outlines the main benefits of adopting such an approach, identifying communication of the requirements, and feedback and progress tracking as the areas that are most probable to benefit. The chapter presents the benefits of quantification, proposes a specific quantification approach (Planguage), and finally describes a successful case study of quantifying quality in a Norwegian organization.

Section II: Quality within Agile Development

Chapter IV: Requirements Specification user Stories

In this chapter, Vagelis Monochristou and Maro Vlachopoulou (Greece) review quality assurance in the requirements specification development phase. Such phase is known to give a lot of problems and injects hard to detect and correct defects in the documentation and the software itself. The authors discuss several approaches, which suggest ways of managing user's requirements (software requirements specification, use cases, interaction design scenarios, etc.). They emphasize the fact that many real users requirements appear in development phases following the initial ones. One way to cope with this situation is to involve customers/users in these development phases as well. When provided with insight about the various sub-systems as they are developed, customers/users can re-think and update their requirements. However, to accommodate such customer/user role within the development cycle, software organizations must take a non-traditional approach. Agile methods are this alternative approach because of the iterative and incremental way of development they propose. Allowing for iteration and gradual system building, user requirements revision mechanisms, and active user participation is encouraged and supported throughout the development of the system. User stories are the agile answer to the problem and they are thoroughly discussed and illustrated in this chapter.

Chapter V: Handling of Software Quality Defects in Agile Software Development

Although the previous chapter told us how to capture and avoid problems in user requirements, defects can still be injected in the software code. In agile software development and maintenance, the phase

that allows for continuous improvement of a software system by removing quality defects is refactoring. However, because of schedule constraints, not all quality defects can be removed in just one refactoring phase. Documentation of quality defects that are found during automated or manual discovery activities (e.g., pair programming) is necessary to avoid waste of time by rediscovering them in later phases. However, lack of documentation and handling of existing quality defects and refactoring activities is a typical problem in agile software maintenance. In order to understand the reason for modifying the code, one must consult either proprietary documentations or software versioning systems. Jörg Rech (Germany), the author of this chapter, describes a process for the "recurring and sustainable discovery, handling, and treatment of quality defects in software systems." His proposed tool for assuring quality in this context is an annotation language, capable to register information about quality defects found in source code, representing the defect and treatment activities of a software sub-system. One additional benefit from using such annotation language is that it can also be useful during testing and inspection activities.

Chapter VI: Agile Quality Assurance Techniques for GUI-Based Applications

In this chapter, Atif Memon and Qing Xie (USA) adopt a strong, process-based approach for assuring quality while developing in agile mode. They discuss the need for new agile model-based testing mechanisms, neatly integrated with agile software development/evolution and propose a new concentric loop-based technique, which effectively utilizes resources during iterative development. They call the inner loop "crash testing," applied on each code check-in of the software. The second loop is called smoke testing and operates on each day's build. The outermost loop is called the "comprehensive testing" loop, executed after a major version of the software is available. The authors illustrate their approach on a critical part of today software systems, namely graphical user interface (GUI). They choose GUI front-ends because GUI development is quite suitable for agile development and because rapid testing of GUI-based systems is particularly challenging. They describe in detail the GUI model used to implement the concentric-loop technique.

Section III: Quality within Agile Process Management

Chapter VII: Software Configuration Management in Agile Development

Chapters in this section focus on project activities that are parallel to development and software configuration management (SCM) is an essential part of any software process. Because of frequent changes, multiple iterations and software versions, SCM is of particular importance for any agile project. In this chapter, Lars Bendix and Torbjörn Ekman (Sweden) discuss the peculiarities of agile SCM and argue that SCM needs to be done differently and in a more extended fashion than during traditional development. They also define the ways in which quality is assured through the application of SCM. To do so, they first provide a brief introduction to the focal SCM principles and list a number of typical agile activities related to SCM. Next, they explain the reader that it is possible to define certain general SCM guidelines for how to support and strengthen these typical agile activities. They describe the characteristics of an agile method that are necessary in order to take full advantage from SCM and, as a consequence, to better assure quality. Following the proposed guidelines, any agile project can obtain the best result from SCM according to the agile method it applies and the project particular context.

Chapter VIII: Improving Quality by Exploiting Human Dynamics in Agile Methods

This chapter deals with a completely different process issue than previous chapter, namely the management of the human resources that are involved in agile development. Panagiotis Sfetsos and Ioannis Stamelos (Greece) argue that human factors are still critical for the success of software engineering in general. In particular, agile methods are even more sensitive to human factors because they are heavily based on the contribution and effort of the individuals working in the agile project. Documentation is limited with respect to traditional development and effective inter-personal communication is necessary for successful project completion. The authors describe how a large agile organization can cope with human resource management both at the corporate level and the project level. First part of the chapter proposes and discusses a model for personnel management based on the well-known People-CMM assessment and improvement model. The agile organization can pursue higher model levels by assessing its current situation and by introducing advanced human resource management practices. In doing so, the organization must take profit from the distinguished way in which humans are involved in agile methods and activities. Second part proposes a model that exploits developer personalities and temperaments to effectively allocate and rotate developers in pairs for pair programming. The rationale is that by mixing different types of personalities and temperaments, pairs become more productive and quality is more easily assured.

Chapter IX: Teaching Agile Software Development Quality Assurance

This chapter ends the section on agile process issues dealing with the preparation of software engineers and managers to address agile quality assurance. Authors Orit Hazzan and Yael Dubinsky (Israel) provide a teaching framework that focuses on the way quality issues are perceived in agile software development environments. The teaching framework consists of nine principles, which can be adjusted according to different specific teaching environments and therefore implemented in various ways. The chapter outlines these principles and addresses their contribution to learners' understanding of agile quality. The authors enrich the discussion of their teaching framework by identifying the differences between agile and traditional software development in general, and with respect to software quality in particular. The material of the chapter can be used by software engineering instructors who wish to base students learning on students' experience of the different aspects involved in software development environments.

Section IV: Agile Methods and Quality: Field Experience

Chapter X: Agile Software Development Quality Assurance: Agile Project Management, Quality Metrics, and Methodologies

In the first chapter of the section with results end experiences from agile companies, James F. Kile and Maheshwar R. Inampudi (IBM, USA) deal with a really hot issue, crucial for the further expansion of agile methods. They ask whether "the adaptive methods incorporated within many of the most popular agile software development methodologies can be successfully implemented within a highly disciplined and highly structured software development environment and still provide the benefits accorded to fully agile projects." They observe that agile methods have been applied mostly to non-critical projects, by small project teams, with vague requirements, a high degree of anticipated change, and no significant availability or performance requirements. It is therefore questionable whether agile methods can be applied in situations with strong quality requirements. The authors report an extremely interesting

experience: they describe how one team adopted not one single agile method, but several individual agile development techniques. They manage to achieve software development quality improvements, while in parallel reducing overall cycle time. The authors propose that all is needed is a common-sense approach to software development. Overall, they demonstrate that the incorporation of individual agile techniques may be done in such a way that no additional risk is incurred for projects having high availability, performance, and quality requirements.

Chapter XI: Test-Driven Development: An Agile Practice to Ensure Quality is Built from the Beginning

This chapter is written by Scott Mark (Medtronic, USA) and describes the practice of test-driven development (TDD) and its impact on the overall culture of quality and quality assurance in an organization. The discussion on this popular practice is based on the author's personal experience introducing TDD into two existing development projects in an industrial setting. He discusses basic concepts of TDD from an industry practitioner's perspective and he proceeds with an elaboration of the benefits and challenges of adopting TDD within a development organization. He reports to the reader that TDD was well-received by team members, and he is optimistic, in the sense that other teams will behave in the same manner, provided that they are prepared to evaluate their own experiences and address the challenges imposed by TDD.

Chapter XII: Quality Improvements from using Agile Development Methods: Lessons Learned

This chapter, ending the session with experiences from industry (and the book), comes from another large company, namely Siemens (USA). Beatrice Miao Hwong, Gilberto Matos, Monica McKenna, Christopher Nelson, Gergana Nikolova, Arnold Rudorfer, Xiping Song, Grace Yuan Tai, Rajanikanth Tanikella, and Bradley Wehrwein report that "in the past few years, Siemens has gained considerable experience using agile processes with several projects of varying size, duration, and complexity." The authors build on this invaluable experience for the agile world and report that they have observed "an emerging pattern of quality assurance goals and practices across these experiences." They describe the projects in which they have used agile processes. They also provide information on the processes themselves. They discuss briefly project quality goals and practices and present (as the chapter title promises) the lessons learned from the successes and failures in practicing quality assurance in agile projects. The material they provide is informative about the methods they employed for achieving the established quality goals, leading to a first-hand understanding of the current state of ASDQA.