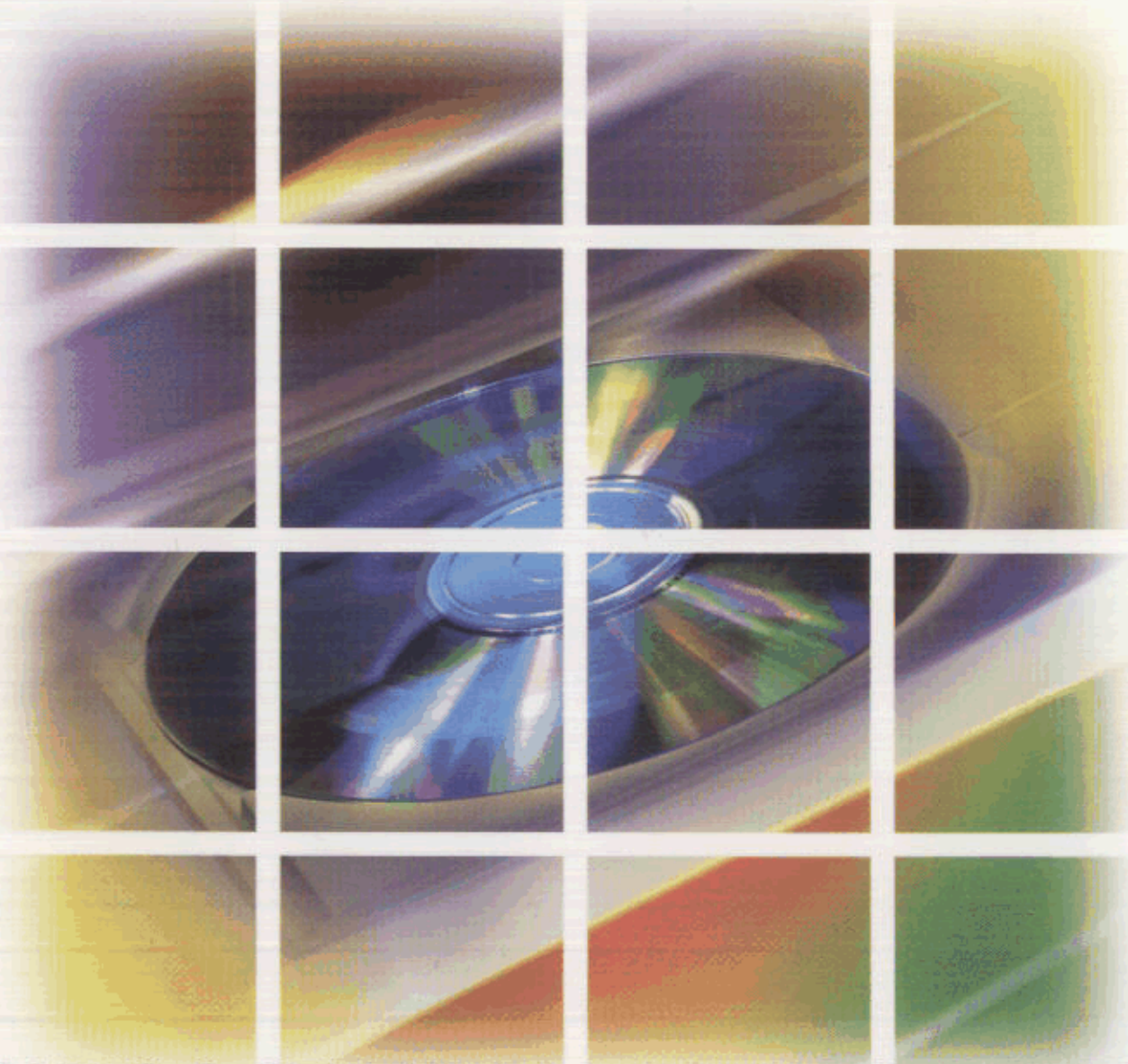
PREMIER REFERENCE SOURCE

Agile Software Development Quality Assurance



IOANNIS G. STAMELOS & PANAGIOTIS SFETSOS

Detailed Table of Contents

proposes the quantification approach Planguage. Interesting results from Planguage application in the context of a Norwegian organization are given.

Section II
Quality within Agile Development

In this chapter, the authors describe a number of approaches for managing user requirements (namely software requirements specification, use cases, interaction design scenarios). Requirements are subject to constant changes in modern software development and the text shows how agile methods promote the involvement of customers/users in the process of requirement modification. The tool for assuring requirements quality are user stories and is thoroughly discussed and illustrated in this chapter.

This chapter discusses refactoring, an agile procedure during which, among other activities, quality defect removal takes place. Because of time constraints, quality defects can not be removed in just one refactoring phase. Documentation of detected quality defects is therefore necessary and the text proposes a process for the recurring and sustainable discovery, handling, and treatment of quality defects in software systems. The process is based on an annotation language, capable to register information about quality defects found in source code.

This chapter proposes a process-based approach for assuring quality while developing in agile mode. The authors propose a new concentric loop-based technique, which effectively utilizes resources during iterative development. It is based on three types of testing, namely crash testing, smoke testing, and comprehensive testing. The overall approach is illustrated on the development of graphical user interfaces. The GUI model used to implement the concentric-loop technique is given in detail.

Section III
Quality within Agile Process Management

Because of the frequent changes, multiple iterations, and software versions that occur in agile development, software configuration management is a crucial activity. This chapter discusses the additional requirements for software configuration management with respect to the traditional development. Typical agile activities for configuration management are described along with general guidelines. It is argued that an agile project can assure better quality according to the agile method and configuration management it applies and the project particular context.

This chapter explores the management of the human resources that are involved in agile development. Because evidently human factors are critical for the success of agile methods, there is an urgent need for managing agile people effectively 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. In addition, the chapter proposes a model for allocating and rotating developers in pairs while pair programming. The model is based on the fact that different types of personalities and temperaments allow pairs that produce better quality results.

This chapter differs from the rest of the book in the sense that it deals with the education of software engineers and managers to form a culture for agile quality assurance. The text proposes a teaching framework focusing on the way quality issues are perceived in agile software development environments. It consists of nine principles, which can be adjusted according to different specific teaching environments. The teaching framework is based on the differences between agile and traditional software development. Overall, this chapter provides a particularly useful tool for instructors of Agile Methods.

This chapter examines one of the hottest issues in modern software development, namely the adoption of agile methods by highly disciplined and highly structured software development environments. It appears that up to now, agile methods have been applied mostly to non-critical projects. The text describes how one IBM software development team has applied simultaneously several individual agile development techniques. The authors report encouraging results, stating that they obtained increased quality in shorter than normal time. Throughout the chapter, it is shown that the adoption of individual agile techniques can be achieved with no additional risks.

This chapter describes the practice of test-driven development and the benefits it brings to quality assurance in an agile organization. The practice is illustrated through details of two real development projects in an industrial setting. The author gives an industry practitioner's perspective and discusses various practical considerations about the adoption of the practice. Overall, it is claimed that test-driven development is well accepted by practitioners and is a successful quality assurance technique.

In this chapter, the experience of another large company, namely Siemens, with agile methodologies is reported. The authors report that Siemens has applied agile processes in several projects with varying characteristics. They also report that significant quality achievements have been observed. The text discusses briefly project quality goals and practices and summarizes the lessons learned from successes and failures while working for quality assurance in their projects. This chapter is important because it shows how a large company pursues quality assurance results when applying agile methods.