Chapter 7.3
Agile Software Development Quality Assurance:
Agile Project Management, Quality Metrics, and Methodologies

James F. Kile
IBM Corporation, USA

Maheshwar R. Inampudi
IBM Corporation, USA

ABSTRACT

Of great interest to software development professionals is whether the adaptive methods found in agile methodologies can be successfully implemented in a highly disciplined environment and still provide the benefits accorded to fully agile projects. As a general rule, agile software development methodologies have typically been applied to non-critical projects using relatively small project teams where there are vague requirements, a high degree of anticipated change, and no significant availability or performance requirements (Boehm & Turner, 2004). Using agile methods in their pure form for projects requiring either high availability, high performance, or both is considered too risky by many practitioners (Boehm et al., 2004; Paulk, 2001). When one investigates the various agile practices, however, one gets the impression that each may still have value when separated from the whole. This chapter discusses how one team was able to successfully drive software development quality improvements and reduce overall cycle time through the introduction of several individual agile development techniques. Through the use of a common-sense approach to software development, it is shown that the incorporation of individual agile techniques does not have to entail additional risk for projects having higher availability, performance, and quality requirements.

INTRODUCTION

Traditional software development approaches, perhaps best represented by the capability maturity model for software (SW-CMM) (Paulk, Curtis, Chrissis, & Weber, 1993) and its successor the capability maturity model for software integration (CMMI®) (Chrissis, Konrad, & Shrum, 2003), focus on a disciplined approach to software development that is still widely used by organizations as a foundation for project success. While the strength of traditional development methods is their ability to instill process repeatability and standardization, they also require a significant amount of organizational investment to ensure their success. Organizations that have done well using traditional approaches can also fall victim of their success through a strict expectation that history can always be repeated (Zhiying, 2003) when the environment becomes uncertain.

Agile development practices have frequently been presented as revolutionary. There is some evidence, however, that they can offer an alternative common-sense approach when applied to traditional software engineering practices (Paulk, 2001). Perhaps they can be used in part to improve the development processes of projects that do not fit the usual agile model (e.g., critical systems with high availability requirements)? Indeed, it has been suggested that project risk should be the driving factor when choosing between agile and plan-driven methods (Boehm et al., 2004) rather than overall project size or criticality. This implies that certain components of *any* project may be well suited to agility while others may not.

This chapter discusses how agile methods were used on one team to successfully drive software development quality improvements and reduce overall cycle time. This is used as a framework for discussing the impact of agile software development on people, processes, and tools. Though the model project team presented is relatively small (eight people), it has some decidedly non-agile characteristics: It is geographically distributed, it has no co-located developers, the resulting product has high performance and reliability requirements, and the organization's development methodology is decidedly waterfall having gained CMM® Level 5 compliance. Therefore, some of the fundamental paradigms that serve as the basis for successful agile development—extreme programming (Beck & Andres, 2005), for example—do not exist. Nevertheless, they were successfully able to implement several agile practices while maintaining high quality deliverables and reducing cycle time.

Chapter Organization

This chapter is organized as follows:

1. **Background:** Some history is given about our model project team and what led them to investigate agile methods. The concept of using a hybrid plan- and agile-driven method is also introduced.
2. **Approaching Selection:** How did our model project team decide which agile practices to use and which ones to discard? This section discusses the risk-based project management and technical approach used.
3. **Implementation:** This section presents how each selected agile practice was incorporated into the software development process.
4. **Impact:** How did the project team know the implemented agile practices were providing some benefit? This section talks generically about some of the metrics that were used to compare the project to prior projects performed by the same team and the impact the selected methods had on the project.
5. **Future Trends:** A brief discussion about what path will be taken to approach follow-on projects.
6. **Conclusion**.