Chapter 14
Agile Development Processes and Knowledge Documentation

Eran Rubin
Holon Institute of Technology, Israel

Hillel Rubin
Israel Institute of Technology (Technion), Israel

ABSTRACT

Agile processes emphasize operational system code rather than its documentation. Ironically, however, some traditional documentation artefacts come to support system-stakeholders interaction, which is another core aspect of agile development processes. In this chapter, the authors examine the relationship between system development and knowledge documentation. They develop an approach that enables incorporating domain documentation to agile development while keeping the processes adaptive. The authors also provide a system design that actively uses domain knowledge documentation.

INTRODUCTION

Agile development processes come to enable a more flexible and adaptive system development process than the traditional development processes do (Maurer & Hellmann 2013, Fowler 2005).

Agile methods require less documentation for tasks, and promote implementation based on informal collaborations between system stakeholders (Fowler 2005). While traditional software engineering methods emphasize careful planning and design, agile methods emphasize the actual software implementation.

However, this shift of emphasis is not without cost. Documentation which is lost under agile development processes could have helped, among other things, to facilitate knowledge sharing and reduce knowledge loss when team members become unavailable (Abrahamsson et al. 2003). Indeed compromising on documentation is not a key point, but rather a consequence of the agile objective of being adaptive (Paetch et al. 2003), which agile methods attempt to overcome by significantly relying on constant collaboration between developers and users (Abrahamsson et al. 2003). While such approaches may well lead to the release of a system that fits customer needs, the knowledge extracted under such approaches will be hard to access after development is complete.

DOI: 10.4018/978-1-4666-6026-7.ch014

The premise of this chapter is that it should be possible to support documentation in agile development methods without compromising the agile manifesto. If documentation is adaptive, and if the documentation supports people collaboration rather than replacing it, then documentation can be well aligned with agile development principles.

In this work we discuss the kind of documentation that can support collaboration and the way to integrate such documentation in agile development. Namely, we propose a way of creating an adaptive system for documenting the knowledge necessary for the interaction and collaboration between system stakeholders.

Our approach provides agile documentation of domain knowledge gathered during systems analysis in traditional processes. Specifically, we provide an approach to document in agile processes the type of knowledge, which under traditional processes would have typically been documented during systems analysis. We aim to document this type of knowledge as the traditional system analysis stage is the stage in which all system stakeholders interact and a common understanding of the domain is established and documented. Therefore, in the traditional system analysis phase we are able to find documents supporting collaboration, which are missing in agile development processes. Although such documents are missing in agile processes, they have great potential to facilitate these processes' effectiveness. For example, Daneva et al (2013) point out that understanding requirements dependencies and vendor's domain knowledge is a key asset for setting up successful client-developer collaboration in agile methodologies.

Accordingly, we identify a set of collaboration supporting documents of the traditional development processes, and we establish a method to incorporate such documents in agile processes. Namely, since agile development emphasizes the reference to working system code, we develop a way of having the identified documentation as part of the executable system code. More specifically, we suggest a system architectural design which enables adaptable documentation as part of the source code. We term our proposed system design Active Documentation Software Design (ADSD). Under this design, source code execution incorporates the execution of documentation statements, which in turn drive the processing of the system. Stated differently, with ADSD changes in the documentation change executable code and vice versa, changes in source code change the relevant documentation.

This chapter is developed as follows:

- The background describes the motivation for this work.
- The section "supporting agile documentation" elaborates on principles guiding the development of the architecture.
- The section "representation of domain knowledge in the system code" describes the architecture and the implementation of components for its support.
- The section "the ADSA system design" provides a description of experience in using and applying the architecture.
- Finally, come discussion and summary of the manuscript.

BACKGROUND

Traditionally, the design process gradually moves from the "problem space" to the "solution space" (Booch 1987). However, as different paradigms of programming emerged, the difference between the problem and solution domain became less and less distinct (Henderson-Sellers & Edwards 1990). Initially, programs did not exhibit problem level information that could be understood or modified. The programmer was not required to make a program understandable and modifiable, but rather programs were merely measured by their efficiency and whether they could accurately solve a specific problem. Presently, software engineering