Chapter 1

Product Backlog and Requirements Engineering for Enterprise Application Development

Chung-Yeung Pang

https://orcid.org/0000-0002-7925-4454

Seveco AG, Switzerland

ABSTRACT

The combination of software engineering and agile development process can bring great benefits to the development and maintenance of enterprise applications. This chapter introduces the engineering of requirements in an agile Scrum development process. In Scrum, requirements are usually defined as user stories in a product backlog. Since many enterprise applications do not have intensive user actions, stories cannot be linked to users. Topics such as user and enabler stories, non-functional requirements, considerations of architecture and business components, business workflow, and breakdown of user stories are addressed. The requirements engineering of PBIs is illustrated by practical examples. Using UML use cases and collaboration models is recommended to transform the stories in the product backlog into formal requirements specifications. The proposed approach can be part of the agile development framework for flexible software products that are easy to customize and maintain.

INTRODUCTION

An important aspect of software development is the requirements analysis. “You need to know what you need to program before you can program it,” is the reason for the requirements analysis. Once you have identified the need in the application, you must also document it. The document would become the requirement specification. It is a daunting task to create a proper requirement specification for a business application. There has been a lot of research in this area (Wiegers & Beatty, 2013). The way to do requirements specification, which we usually call requirements engineering, is a discipline in itself.

DOI: 10.4018/978-1-7998-2531-9.ch001
In the traditional development approach, business analysts would try to begin with requirements analysis and specification (Bell & Thayer, 1976). Once this task is completed, the specification is signed off and development begins. This approach has not proven effective as requirements are often moving targets. In fact, many projects failed because the development teams did not develop suitable requirements specifications.

As software technology advances, we recognize that software development is an iterative process. Requirements analysis and specification are also part of the iterative process. Business applications need to be flexible at all times to meet new requirements and changes. The agile approach to software development has been developed for this reason (Larman, 2003). It is widely used in the software community, especially in the IT development teams of large companies.

There are many methods for an agile development process. This chapter focuses on Scrum (Stellman & Greene, 2015) which is widely used in the software community. For Scrum, there is no activity for the requirement specification. Requirements in the Scrum product backlog are usually user stories. These user stories are taken out in the iterative development process with sprints from the product backlog. There are many literatures for writing user stories (Abdou et al., 2014; Rehkopf, 2019; User Story, 2019).

User stories are not detailed requirements specifications (what a system should do), but negotiable declarations of intent (it needs to do something about like this) (Scrum Expert, 2016). They are short, easy to read and understandable to developers, stakeholders and users. Therefore, user stories should be turned into more stringent requirements specifications before implementation. Another problem is that user stories are great for capturing functional requirements, but they do not work well with design constraints and non-functional requirements (Cohn, 2015; Galen, 2013; Badri, 2016). Solutions to these problems are presented in this chapter.

The materials in this chapter are based on the techniques given in the literature and 15 years of experience of the author in the agile development. The aim of this chapter is to provide concepts and mechanisms for agile requirements engineering based on practical examples. Emphasis is placed on applications with architectural and design constraints and non-functional requirements, as well as formal requirements documentation that is not normally covered by user stories. The chapter is structured as follows. The Background section discusses the historical background of software development and software / system life cycle (SDLC). The next section focuses on both traditional and agile requirements engineering. The following section is about product backlogs and user stories. This is followed by two sections with practical examples on agile requirements engineering based on user stories. The following section provides an explanation of the UML use cases introduced as an approach to formal specification. This section also describes the implementation approach using UML collaboration models. The chapter closes with a vision for the future section and a final conclusion.

BACKGROUND

As background, the following subsections discuss software engineering and the traditional and agile development process lifecycle.