Asia Pacific Management Review xxx (2017) 1–8

HOSTED BY

Contents lists available at ScienceDirect

Asia Pacific Management Review

journal homepage: www.elsevier.com/locate/apmrv

Snowman: Agile development method with institutionalized communication and documentation for capstone projects

Wen-Lung Tsai [a, *], Chung-Yang Chen [b], Chun-Shuo Chen [b]

[a] Department of Information Management, Oriental Institute of Technology, New Taipei City, Taiwan
[b] Department of Information Management, National Central University, Taoyuan City, Taiwan

ARTICLE INFO

Article history:
Received 20 February 2015
Accepted 11 January 2017
Available online xxx

Keywords:
Agile method
Software development
Software engineering education
Capstone project
Snowman approach

ABSTRACT

Numerous organizations currently implement agile methods in practical software development processes. Using agile methods involves mutually strengthening communication to ease the burden involved in documenting plan-driven projects. A focus on people-oriented communication has resulted in agile methods being diverted from development processes that codify formal documents. This situation makes follow-up software maintenance difficult. This paper proposes an appropriate and concrete approach for streamlining communication and documentation in software engineering education called the Snowman approach. The results of evaluations conducted indicate that the proposed Snowman approach enables undergraduate students to learn agile methods and resolves misconceptions surrounding agile methods.

1. Introduction

Agile methods in software development (Martin, 2003), including Extreme Programming (XP), Scrum, and Crystal Family, have recently appeared as popular approaches in the software development arena. These approaches resolve many problems that have been plaguing traditional software development methods for over a quarter of a century (Baskerville & Pries-Heje, 2004)—namely, "that systems cost too much, take too long to develop, and do not serve their intended purpose when eventually delivered" (Conboy & Fitzgerald, 2010 [P.2:2]). The new tendency of software development implicates its corresponding adoption in the approach of project courses in computer-related education. Therefore, agile methods need to be recommended in software engineering education.

Senior projects provide appropriate and integral training in software engineering education. In this regard, senior projects serve as capstone projects on the basis of curricular computing-related education policy. These projects are implemented one or two semesters prior to graduation in countries such as the United States and Taiwan, and most involve short-cycle-time software development (Chen, 2009, 2011; Laplante, 2006; Robillard & Dulipovici, 2008). Undergraduate students need to implement an entire capstone project via observation and application. Capstone projects advocate the training of software development approaches (specifically, XP, Scrum, and Crystal Family) to reflect practical requirements in industry (Boehm & Turner, 2005). However, many people (particularly, practical engineers and students) have misconceptions concerning agile methods without documentation (Boehm & Turner, 2005; DeMarco & Boehm, 2002; Laplante, 2006; Omoronyia, Ferguson, Roper, & Wood, 2010; Pikkarainen, Haikara, Salo, Abrahamsson, & Still, 2008).

In general, agile methods provide and suggest practices for interaction among software project stakeholders (Begona, Maite, & Isabel, 2013). Although agile methods increase communication capabilities in a software development project, Boehm and Turner (2005), Pikkarainen et al. (2008) and Robillard and Dulipovici (2008) stated clearly that the practices using agile methods overemphasized informal or verbal communication. Boehm and Turner (2005) and Cohn and Ford (2003) reported that informal or verbal communication increases the chasm among stakeholders in software development projects and can even lead to software project failure. Thus, formal communication is absolutely necessary in agile methods. Furthermore, Beck (2007), Mahnic (2010, 2012), Williams (2010), and Mahnic and Casar (2016) considered that the content of

* Corresponding author.
E-mail address: wltsai@mail.oit.edu.tw (W.-L. Tsai).
Peer review under responsibility of College of Management, National Cheng Kung University.

http://dx.doi.org/10.1016/j.apmrv.2017.01.002

informal or formal communication needs to be composed into documentation as a lesson learned paradigm. In particular, it is necessary to break away from the misconceptions of communication and documentation by teaching agile methods through computing related capstone projects.

Meeting flow approach (MFA) was proposed by Chen (2009, 2011) to deal with communication problems in capstone projects in computing-related fields. MFA institutionalizes communication in the form of formal meetings (Chen, 2009, 2011). This communication differs from the interaction communication in existing agile methods as the objective of MFA is to institutionalize effective group communication in the collaborative development of software projects. Therefore, we consider the concept of MFA to facilitating verbal or informal communication in existing agile methods. Moreover, MFA still lacks an appropriate documentation method. This paper proposes an approach that applies change or challenge during a software development project that is suitable for computing-related capstone projects. The proposed approach was conceptualized with the following objectives:

- Institutionalization of the communication in existing agile methods based on MFA.
- Facilitate documentation in computing related capstone projects.

The purposed approach, called Snowman, focuses on institutionalized group communication through the holding of meetings. According to the nature of each meeting, there are different project roles in which to participate. Furthermore, Snowman can also help to train undergraduate students in how to compose documents. Currently, Snowman is being applied in capstone projects in a university. To validate Snowman, this study was conducted over two years (by two graduates). Subsequently, the conclusion was reached that the operation of Snowman can improve communication and team collaboration in capstone projects. Further, it disabuses the misconceptions that occur without documentation in agile methods. The insight gained in this study also indicates that Snowman can aid undergraduate students in time management of their projects.

The remainder of this paper is organized as follows. Section 2 discusses related work that gives an overview of the nature of agile methods and capstone projects. Section 3 outlines the design and the underlying concept of the proposed approach—Snowman. Section 4 explains its implementation and procedure. Section 5 discusses the evaluations conducted of the proposed approach and the results obtained in capstone projects. Finally, Section 6 summarizes the contributions made in this paper and outlines the directions for future work.

2. Related work

Many studies have been conducted on the adoption of agile methods in industry. These studies have revealed that the management of the development processes has improved (Pikkarainen et al., 2008; Williams, 2010). Using processes to develop artifacts (e.g., software or system) in capstone projects in computing-related education reflected industry improvements in practice (Laplante, 2006; Robillard & Dulipovici, 2008; Williams, 2010; Mahnic, 2010, 2012; Mahnic & Casar, 2016). Although it is tendentious, misconceptions still surround the implementation of agile methods in capstone projects. Fig. 1 depicts the context of the related work in this research in conjunction with the proposed solution.

2.1. Agile methods and misconceptions

Agile methods for software development appeared in opposition to document-oriented heavyweight software development processes (Cohen, Lindvall, & Costa, 2004) that are characteristic of the traditional disciplined approach recommended by software quality models (Moe, Dingsøyr, & Dybå, 2010; Mahnic, 2010, 2012) such as CMMI (SEI, 2006). According to the Manifesto for Agile Software Development (Beedle et al., 2001), these methods valued individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan.

Utilizing statements made by Glazer, Dalton, Anderson, Konrad, and Shrum (2008) and Miller (2001), we synthesized the characteristics of agile methods for software development as follows. (1) Small self-management development teams. (2) Units of one to six weeks for short-cycle iterative development, which involves rapid verification and validation. (3) Simplification of all development processes except program development activities. (4) Adjustment of contingent risks. (5) Incremental development of small-scale function modules. (6) Acceptance of changes in demand during development. (7) People-oriented, collaborative, and communicative model.

In light of the nature of agile methods, many people (especially practical engineers and students) have misconceptions concerning agile methods without documentation (Boehm & Turner, 2005; DeMarco & Boehm, 2002; Laplante, 2006; Omoronyia et al., 2010; Robillard & Dulipovici, 2008). Although agile methods increase communication capabilities in software development projects, as clearly stated by Boehm and Turner (2005), Pikkarainen et al. (2008), and Robillard and Dulipovici (2008), practices using agile methods overemphasized informal or verbal communication. Boehm and Turner (2005) and Cohn and Ford (2003) reported that informal or verbal communication increased the chasm among stakeholders in software development projects and can even result in software project failure. Thus, formal communication is absolutely necessary in agile methods. Furthermore, Beck (2007), Mahnic (2010, 2012) and Mahnic and Casar (2016) considered that the content of informal or formal communication needs to be composed into documentation as a lesson-learned paradigm. In particular, it is necessary to break away from the misconceptions of communication and documentation in the teaching of agile methods through computing-related capstone projects.

2.2. Agile methods in computing-related capstone projects in engineering education

Traditionally, capstone project teams in computing-related education developed a software or system for different customers and domains. However, these capstone project teams were similar in nature (Umphress, Hendrix, & Cross, 2002). They acquired requirement, analyzed and designed artifacts (i.e., system or software), and even implemented them, but scarcely verified or validated their achievements and qualities (Dingsoyr, Jaccheri, & Wang, 2000; Jaccheri, 2001). Further, documentation in capstone projects was for the most part sparing and vacuous. Researchers have also reported existence of the following constraints in capstone projects (Umphress et al., 2002; Laplante, 2006; Ras, Carbon, & Decker, 2007; Strode & Clark, 2007; Mahnic, 2010, 2012; Scharf & Koch, 2013; Rodriguez, Soria, & Campo, 2015; Mahnic & Casar, 2016). (1) Time and commitment: Projects must be completed within 12–16 weeks, and even two semesters. (2) Experience levels: The technical ability and development experience of team members are not uniform. (3) Scope and complexity: Although the scope and

Please cite this article in press as: Tsai, W.-L., et al., Snowman: Agile development method with institutionalized communication and documentation for capstone projects, Asia Pacific Management Review (2017), http://dx.doi.org/10.1016/j.apmrv.2017.01.002