Information and Software Technology 61 (2015) 163–181

Contents lists available at ScienceDirect

Information and Software Technology

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

Exploring principles of user-centered agile software development: A literature review

Manuel Brhel [a], Hendrik Meth [b], Alexander Maedche [a,b,*], Karl Werder [a]

[a] Chair of Information Systems IV, University of Mannheim, L 15, 1-6, 68131 Mannheim, Germany
[b] Institute for Enterprise Systems, University of Mannheim, L 15, 1-6, 68131 Mannheim, Germany

ARTICLE INFO

Article history:
Received 5 September 2013
Received in revised form 7 January 2015
Accepted 7 January 2015
Available online 17 January 2015

Keywords:
Agile software development
User-centered design
Systematic literature review

ABSTRACT

Context: In the last decade, software development has been characterized by two major approaches: agile software development, which aims to achieve increased velocity and flexibility during the development process, and user-centered design, which places the goals and needs of the system's end-users at the center of software development in order to deliver software with appropriate usability. Hybrid development models, referred to as user-centered agile software development (UCASD) in this article, propose to combine the merits of both approaches in order to design software that is both useful and usable.
Objective: This paper aims to capture the current state of the art in UCASD approaches and to derive generic principles from these approaches. More specifically, we investigate the following research question: Which principles constitute a user-centered agile software development approach?
Method: We conduct a systematic review of the literature on UCASD. Identified works are analyzed using a coding scheme that differentiates four levels of UCASD: the process, practices, people/social and technology dimensions. Through subsequent synthesis, we derive generic principles of UCASD.
Results: We identified and analyzed 83 relevant publications. The analysis resulted in a comprehensive coding system and five principles for UCASD: (1) separate product discovery and product creation, (2) iterative and incremental design and development, (3) parallel interwoven creation tracks, (4) continuous stakeholder involvement, and (5) artifact-mediated communication.
Conclusion: Our paper contributes to the software development body of knowledge by (1) providing a broad overview of existing works in the area of UCASD, (2) deriving an analysis framework (in form a coding system) for works in this area, going beyond former classifications, and (3) identifying generic principles of UCASD and associating them with specific practices and processes.

Contents

* Corresponding author at: Institute for Enterprise Systems, University of Mannheim, L 15, 1-6, 68131 Mannheim, Germany. Tel.: +46 621 181 3606; fax: +49 621 181 3627.
E-mail addresses: brhel@es.uni-mannheim.de (M. Brhel), meth@es.uni-mannheim.de (H. Meth), maedche@es.uni-mannheim.de (A. Maedche), werder@es.uni-mannheim.de (K. Werder).

http://dx.doi.org/10.1016/j.infsof.2015.01.004
0950-5849/

1. Introduction

The question of how to meet the challenge of organizing software development activities has concerned researchers and practitioners ever since software engineering (SE) emerged as an independent scientific discipline in the 1970s [1,2]. In the last decade, the mechanistic view of software development prevalent in earlier phase-based linear approaches has been replaced by an understanding of development activities as a dynamic process characterized by iterative cycles and the active involvement of all stakeholders [3]. This is reflected in *agile software development* (ASD), which responds to unpredictable change by relying on people and their creativity rather than on processes [4], and limit software development strictly to activities that add business value for the customer [5–7].

With agile methods becoming mainstream even for large-scale organizations in the software industry [8,9], software is being delivered on time and in budget, and customer demands are being met increasingly often [10,11]. Nevertheless, agile methods focus on the question of how *useful* software can be developed, with customer value being understood as primarily driven by providing an appropriate functional scope. They do not necessarily focus on developing software that is considered *usable* [12–14], i.e. usability defined as the extent to which a software can be used by specified users to achieve specified goals effectively, efficiently, and satisfactorily in a specified use context [15]. While usability is not a central topic in SE, in which it is considered one of many non-functional requirements and quality attributes [16], it has become crucial for economic success in highly competitive markets and can be used to set the product apart from that of the competition [17–19]. *User-centered design* (UCD) ensures that the goals and needs of the system's end-users is the focus of the product's development. In the field of human–computer interaction (HCI), terms like UCD, usability engineering, and interaction design often have a very similar meaning [20]. UCD is driven by continuous end-user evaluation and the iterative refinement of design concepts and prototypes [21].

Given the need to deliver business value to the customer in a rapidly changing environment while taking the needs of end-users into account, the integration of UCD and ASD seems to be a promising endeavor and has received increasing attention in recent years [20,22,23]. In contrast to plan-driven SE, whose properties often impede the integration of UCD [25,26], similarities between ASD and UCD provide a common ground, which eases integration. Moreover, owing to these similarities, a multitude of integration approaches has been suggested in the literature [20,21,24]. Thereby, some authors focus on the benefits of integrating particular usability practices into ASD [23,47] or vice versa, while other authors propose integrated *user-centered agile software development* (UCASD) approaches [20,22].

Several literature reviews on UCASD research have been conducted. Although we have identified a plethora of existing works in these reviews, they suffer from several shortcomings: First, existing reviews lack a comprehensive coverage of the different dimensions of UCASD. They mainly focus on the actual software development practices to be used, and rarely describe further dimensions, such as the overall process to be followed, the people and social aspects involved or the technology that may be leveraged. Second, findings have been insufficiently abstracted and systematized, impeding a generalization of the results and applicability in specific contexts. For example, while it might be helpful to know the most elaborated practice to conduct end-user evaluations in a usability lab, this specific practice might be inappropriate when an according infrastructure is not available. In these cases, a more generic principle, i.e. a rule based on the examination of underlying concepts [6], might be a more suitable starting point. Finally, from a methodological point of view, the conducted literature reviews on UCASD lack clear quality criteria for paper selection and need to be complemented by current findings. In order to address these research gaps, this paper aims at capturing and analyzing the current state of the art in UCASD. More specifically, we investigate the following research question: *Which principles constitute a user-centered agile software development approach?* Thus, following an approach similar to the *agile manifesto* [134], we derive a set of grounded principles for UCASD from the literature.

The remainder of the paper is organized as follows: In Section 2, we establish the foundations of our work by summarizing related work and outlining the existing research gap. Section 3 introduces the research method applied in this paper, describing the sourcing and search strategy, the paper selection process, and the final analysis. In the following sections, we present and discuss the results of the systematic review: First, Section 4 presents an overview of the results. The identified principles of UCASD are subsequently discussed in Section 5. Finally, Section 6 provides a summary of the paper and outlines the limitations and contributions of our work.

2. Foundations

ASD and UCD evolved from different motivations. Software engineers aim to satisfy customers through timely releases and responsiveness to change requests without compromising software quality. These goals are difficult to achieve through plan-driven SE approaches, resulting in proposals for ASD [27,28]. UCD aims at ensuring appropriate usability of the implemented software, a characteristic that has not been considered sufficiently in traditional, plan-driven approaches or in agile approaches. UCD addresses this issue but does not consider agile principles. Therefore, first attempts to integrate ASD and UCD approaches were made about a decade ago. In the following, we present and analyze