Computer Standards & Interfaces 000 (2017) 1–13

Contents lists available at ScienceDirect

Computer Standards & Interfaces

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

Knowledge management in requirement elicitation: Situational methods view

Deepti Mishra [a], Seçil Aydin [b], Alok Mishra [c,*], Sofiya Ostrovska [d]

[a] NTNU - Norwegian University of Science and Technology, Norway
[b] Argela Technologies, Ankara, Turkey
[c] Department of Software Engineering, Atilim University, Ankara, Turkey
[d] Department of Mathematics, Atilim University, Ankara, Turkey

ARTICLE INFO

Keywords:
Knowledge management
Requirement engineering
Situational method engineering
Requirement elicitation

ABSTRACT

In small-scale software development organizations, software engineers are beginning to realize the significance of adapting software development methods according to project conditions. There is a requirement to proliferate this know-how to other developers, who may be facing the same settings/context, so that they too can benefit from others' experiences. In this paper, the application of situational method engineering in requirements elicitation phase is investigated. A novel, simple and dynamic web-based tool, Situational Requirement Method System (SRMS), is developed which can aid in conception/formulation, repository, and elicitation/derivation of methods related with this stage. The proposed approach and tool are validated by distributing a questionnaire among software professionals working in large software companies, and making SRMS accessible to them. The results indicate that a majority of the participants finds SRMS useful and provides various suggestions to improve it.

1. Introduction

Knowledge is a crucial resource for organizations and it should be managed in a way that helps organizations to cope with different situations and increase productivity as well as competitiveness. A portion of knowledge is usually recorded, but the rest may still remain in the individuals' mind. Software development is a knowledge-intensive activity, in which a majority of the knowledge remains with those who acquired it through experience over time. Various tools and techniques are required to capture and process such knowledge in order to benefit in subsequent projects. Knowledge management processes fit software development like a glove [1] and can function as a complement to support individuals during the software development phases to enhance productivity and quality [2]. This process consists of activities such as creating, storing/retrieving, transferring, and applying knowledge [3]. The creation of a knowledge repository or an experience base can assist relatively inexperienced software developers in accomplishing their tasks with greater efficiency. Knowledge is contextual, which means that positive outcomes can only be achieved when it is applied in certain contexts or situation.

Such knowledge can be stored using the Lesson Learned (LL) database, which is being increasingly used and considered as one of the best practices for capturing organizational knowledge in software development organizations. In software engineering, the process of knowledge dissemination is based on the lessons learned [4] in order to maintain a community of interest. The process management of the lessons learned is increasing, especially in the area of Information Technology, aiming at consolidating this process in software projects [5]. Project Management Institute (PMI) [6] defines LL as "the knowledge gained during a project which shows how project events were addressed or should be addressed in the future with the purpose of improving future performance", and the LL knowledge base as a "store of historical information and LL about both the outcomes of previous project selection decisions and previous project performance". Unfortunately, many well-meaning LL databases focus more on the problem rather than providing the solution, also these are difficult to search and provide little help for future projects [7]. It is important that the data entered into the database is clear, concise and has the appropriate keywords to facilitate effective searches [7]. It also helps to reference the individuals who can be contacted for more information [7]. The purpose of an LL system is to collect and supply lessons that can benefit those who encounter situations where the lesson can be applied [8].

Method engineering (ME) is the discipline to design, construct and adapt methods, techniques and tools for the development of information systems [9], and if a method is tuned to the project at hand, this is called 'situational ME' [10]. Methods exist for the purpose of sup-

* Corresponding author.
E-mail addresses: deepti.mishra@ntnu.no (D. Mishra), secil.aydin@argela.com.tr (S. Aydin), alok.mishra@atilim.edu.tr (A. Mishra), sofia.ostrovska@atilim.edu.tr (S. Ostrovska).

http://dx.doi.org/10.1016/j.csi.2017.09.004
Received 6 February 2017; Received in revised form 9 September 2017; Accepted 10 September 2017
Available online xxx
0920-5489/

porting project members during system development projects, both in their individual roles and their collaboration efforts [11]. The situational method engineering (SME) approach focuses on project-specific construction [12], where pre-existing pieces of methods are selected and combined in an attempt to produce the most appropriate process for an organization or a project [13]. This is an active research area (see, for instance, [14] and [15]). SME techniques can be used not only to customize a software development process for a particular project context, but also to perform continuous process improvement [16,17].

The application of SME approaches facilitates in producing project-specific methodologies that are tailored to fit specific development situations and, similar to all engineering disciplines, efficient application of SME methods is dependent on the availability of adequate tools [18]. Specifically, within the ME and SME fields, method and software engineers mainly deal with: (1) the definition of methods (method design), and (2) the construction of the supporting software tools (method implementation) [19]. Therefore, any recommendation aimed at supporting ME should cover these two phases of the ME process [19]. Computer Aided Method Engineering (CAME) environments have been developed for this purpose [20,21]. The two most significant challenges for the SME community are the rate of industry adoption and how to automate the method construction process [14]. As SME follows a bottom-up approach, it is considered a creative task and, hence, requires the knowledge and experience of a method engineer [13]. Working with method requirements is a challenge [14]. According to Henderson-Sellers and Ralyte [14], this is still a major challenge when working with situational methods. The question is how one can handle the method requirements process [14]. Method requirements are often formulated based on interviews with project members [22,23]. In this respect, one of the major constraints is the fact that these requirements are often vague, poorly understood, and difficult to express - which is true for requirements in general - and also they affect the methods that are used within system development projects [11].

Aydin and Mishra [24] explored the use of SME in requirement elicitation phase and a preliminary proposal was put forth in the form of a brief communication on the topic. Mishra et al. [25] presented a web-based prototype that can assist in creating methods related with this phase and compared it with all the available tools in the literature. The present paper extends the work of Mishra et al. [25] and Aydin and Mishra [24] in a substantial way. Here, the proposed approach and tool is validated by making it accessible to a group of senior software professionals working in 10 large software development organizations. Later, a questionnaire is distributed to assess the ease-of-use, usefulness, and effectiveness of the approach and tool. This also helps in collecting feedback from those experts to improve the proposed approach and tool.

The paper is organized as follows: In the next section, the literature is reviewed. In Section 3, the use of situational method engineering in the requirement elicitation phase is explained, as well as how the criteria for categorizing the methods developed in this study. Section 4 explains the use of the Situational Requirement Method System (SRMS) tool. Section 5 describes the tool validation and discussion of the result. Finally, the paper concludes in Section 6.

2. Related research

Software engineering is a knowledge-intensive activity [26], which involves other knowledge-intensive sub-activities such as requirements elicitation, architectural design, risk management and testing, maintenance, etc.

2.1. Knowledge management in software development

Software development organizations are trying to find different ways to store knowledge created during different phases of software development in past projects by using LL databases or similar systems so that this knowledge can be used in future projects. A lesson learned (LL) is the knowledge or understanding gained by experience [27]. The experience may be positive, as in a successful test or mission, or negative, as in a mishap or failure [28]. LLs are rooted in experience, describe both failures and successes and target organizational reuse [8]. Similarly, case-based reasoning is a problem-solving paradigm, in which a new problem is solved by finding a similar past case, and is reused in the new problem situation [29]. Park and Bae [30] proposed an approach to process the tailoring of software process using process slicing and utilises past experience by a case-based reasoning technique to reduce the effort and defects during process tailoring for a large-sized software process. Kato et al. [31] established a supporting system and method with which novice analysts can acquire the necessary requirements through interviews with stakeholders - just like expert analysts - by modeling several kinds of knowledge, such as project-specific knowledge, how to ask questions in a certain situation during interviews, etc. Andrade et al. [32] proposed an architectural model for software testing lesson learned systems with two basic goals: usefulness and applicability to improve and promote the dissemination and reuse of individual experiences gained throughout technical and managerial software testing activities. Vizcaino et al. [33] described a multi-agent system to manage the information and knowledge generated during the software maintenance process in which agents can learn from previous experience and share their knowledge with other agents or communities. This paper attempts to create knowledge base for requirements phase by using situational methods engineering approach.

2.2. Situational method engineering

Rolland et al. [34] observed that a main hallmark element in real processes is the project situation. This situation has a strong bearing in selecting the task best suited to handle it, and also the strategy to be adopted in carrying out this task [34].

Situational method engineering [9,35,36] is the software engineering discipline that describes the creation and use of a software development method from small, atomic methodological pieces, known as ‘method fragments’ or, at a larger scale (i.e. non-atomic), ‘method chunks’ [37]. Gupta and Prakash [22] categorized method engineering into three main phases: method requirement engineering, method design, and method construction and implementation. This study presents the method requirements for methods used in requirements phase in the next section.

Information systems development methods are a subject of study in ME [38]. Casare et al. [39] presented a situational approach, called Medee Method Framework, which allows the development of an organization-centered, multi-agent system in a disciplined way. Harmsen et al. [28] made an analogy between information systems development and method engineering.

2.3. SME approaches

There are three main SME approaches [40]: Assembly-based SME, in which a method is constructed from reusable method components that are extracted from existing methodologies and stored in a repository called the ‘method base’; Extension-based SME’ in which the existing methods are extended and enriched by applying extension patterns approaches [20]; and Paradigm-based SME, in which a new method is constructed by instantiating a meta model or applying abstraction to existing methods [18]. Abad et al. [18] observed that among the different approaches to SME, Assembly-based SME is the most commonly used one and has become the basis of method construction in CAME tools. Abad et al. [18] further argued that method development in these tools consists of three distinct stages: Specifying the method requirements based on the situation of the project, selecting the appropriate method fragments, and assembling the fragments into a coherent methodology. Karlsson and Ågerfalk [11] addressed one of basic assumptions of many method-engineering approaches: that it is possible for project members