Knowledge Management in Requirement Elicitation: Situational Methods View

Deepti Mishra[1], Seçil Aydin[2], Alok Mishra[3], Sofiya Ostrovska[4]
[1]NTNU - Norwegian University of Science and Technology, Norway, E-mail: deepti.mishra@ntnu.no
[2]Argela Technologies, Ankara, Turkey. E-mail: secil.aydin@argela.com.tr
[3]Department of Software Engineering, Atilim University, Ankara, Turkey. E-mail: alok.mishra@atilim.edu.tr
[4]Department of Mathematics, Atilim University, Ankara, Turkey. E-mail: sofia.ostrovska@atilim.edu.tr

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.

Keywords: Knowledge management; Requirement engineering; Situational method engineering; Requirement elicitation

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 supporting 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 Agerfalk [11] addressed one of basic assumptions of many method-engineering approaches: that it is possible for project members to explicitly specify the requirements within a situational method. They proposed a computerized tool MC Sandbox in eliciting and negotiating method requirements between the method users and method engineers during configuration workshops. In this paper, two main SME approaches i.e., Assembly-based SME and Extension-based SME, are used for creating new methods for requirement phase.

2.4 Modular Constructs of a Method

Iacovelli and Souveyet [38] proposed an ontology-based approach in SME to design an ontology of method descriptors as a domain ontology. They defined the semantics of six most popular SME approach modular constructs in order to show their usage and relevance. A method repository containing different reusable method chunks is needed to practice method engineering successfully. Henderson-Sellers, Gonzalez-Perez and Ralyte [41] examined 'method fragment' and 'method chunk' as two main candidates for the atomic element to be used in SME in terms of their conceptual integrity and how they may be used in method construction. Henderson-sellers and Gonzalez-Perez [42] recommended some best practices that should be adopted so that the resultant method fragments are atomic and therefore likely to be consistent in quality, thus leading to higher quality constructed methodologies and paving the way for easier composition and interoperation of the fragments. Similarly, the present study follows the concepts of creating methods from smaller chunks.

Although there exist several studies in the literature which explore SME, there are few academic tool prototypes for this purpose. Most of the tools are originated from research studies, with a few launched by the industry. This is the reason why little contribution has been made to this methodology. Moreover, most of the tools existing in this field lack documentation, hence the industry's reluctance to apply them.

3. Situational method engineering for the requirement elicitation phase

The primary measure of success for a software system is the degree to which it meets the purpose which it was intended for [43]. Requirements engineering (RE) is the process of discovering that purpose by identifying the stakeholders and their needs, and documenting these in a form that is amenable to analysis, communication, and subsequent implementation [43]. Requirement engineering is one of the most important activities in any software development project and consists broadly of requirement elicitation, analysis, specification, verification and management.

There are various requirement elicitation techniques such as interviews, surveys/questionnaires, focus groups, brainstorming, ethnography, requirements workshops, prototyping, etc. Every technique has certain advantages and disadvantages, and none of these techniques may necessarily be ideal for all circumstances. These methods are selected according to the following: familiarity of the methods to requirement analysts and participants, preference of methods, conformance to methodology adopted for elicitation and analysts' mind-set, and relevance to the situation [44]. Therefore, it is important that requirement analysts be aware of every method and their strengths and weaknesses so that they can choose the most appropriate technique according to the project characteristics. Practitioners commonly believe that a combination of methods could help them in eliciting high-quality requirements from stakeholders and obtaining a better understanding of the problem domain [45]. Mishra et al. [46] advocated that a combination of requirement elicitation methods can be used effectively to elicit, verify and validate the requirements according to the project's specific situation. Therefore, the situational method engineering approach can be used for the requirement phase to reuse or create new methods by assembling parts of the existing methods which are specific to a particular project situation.

Karlsson and Agerfalk [11] argue that software developers are users of the methods in the same way as end-users are users of software. Therefore, method users set certain requirements on the methods in much the same way that end-users have requirements on software systems, since both are artifacts that the users are heavily dependent on in their work. There are various terminologies used for these requirements: method requirements [11], project characteristics [47] or descriptors [48]. First, the method requirements need to be formulated. Later, the method is defined using these method requirements and stored in a method repository, from where they can be retrieved depending on the method requirements [11], project characteristics [47], or descriptors [48]. One of the features of ME is to decompose these methods into modular parts for optimizing, reusing, and ensuring their flexibility and adaptability [37]. Using modular method construction, method requirements can be used in selecting method fragments [49], method chunks [50], work products [51], process components [52], or method services [53, 54].

There is a need for situational methods since there cannot be a single method that fits all situations [9, 55, 56]. Projects differ, for example, with respect to development context, delivery, project team, and development timeframe [47]. Abad et al. [18] observed that method development in tools consists of three distinct stages: Specifying the method requirements (descriptors) based on the situation of the project, selecting the appropriate method fragments, and assembling the fragments into a coherent methodology. Kornyshova et al. [57] used multi-criteria similarity matching as a basis for method selection. With this knowledge, various requirement elicitation methods have been explored to find their suitability under different project situations. The authors, with their experience in the software industry along with help from the literature established the following method requirements (descriptors) to classify these requirement elicitation techniques, which can be later used for method selection and creation of new ones:

Experience of the requirement engineer - If the requirement engineer is experienced, he/she is likely to be aware of what works and what does not, in a particular project situation and therefore, will choose the right technique given the project characteristics. For example, if the experience of the engineer is more than 5 years, the introspection method can be used for requirement elicitation; otherwise, a more collaborative method, such as workshops, can be applied where there is a higher chance to obtain inputs from multiple sources.

Experience level of the requirement engineer in similar projects - If the experience level of the

requirement engineer in similar projects is extensive, this means that he/she has data (requirements) from the similar projects in the past. In a sense, the requirement engineer has a prior idea about the customers' needs because of past project's available data. In this case, he/she may employ the introspection technique to collect the initial requirements followed by interviews with customer(s) so as to clarify the additional requirements.

Requirement elicitation period - some requirement elicitation techniques are more efficient in limited time spans, whereas other techniques work better when longer durations are dedicated for this purpose. For example, if we have only a few hours to spare for requirement elicitation, then the role-playing technique can be employed. If this period is 2-3 days, then workshops can be organized to gather requirements provided that the customer is experienced in the field as well. Otherwise, that is, if the customer is inexperienced, ethnography is a better option.

Experience level of the customer - This also affects the requirement elicitation method(s) to be used. If the customers are experienced and clear enough about their needs, workshops and brainstorming techniques are more effective. Otherwise, if the customer does not have enough knowledge about software and software environments, the requirements cannot be elicited easily from them; in which case, ethnography or a series of interviews will be required for this purpose.

Possibility of meeting between the development team and customer(s) - The customers' intensive involvement during the requirement elicitation phase can improve the quality of the requirements. Setting up meetings to make bilateral decisions and evaluating a host of ideas are more effective for requirement elicitation because they can lead to mature and stable requirements in the early phase of the development. If there is a possibility of joint meetings between the two sides, then brainstorming and workshop techniques are effective to elicit the requirements.

Project budget - This can be an important criterion for the selection of sound requirement elicitation technique(s). If the project budget is limited, role-playing can be used since it is both swift and economical. Moreover, storyboarding can also be used because it is inexpensive, easy, and user-friendly. Should there be an opportunity to spare more time and money during this phase, a combination of requirement elicitation techniques can be applied. Despite such time and monetary expenditure, better quality and stable requirements are certain to be achieved.

Project user's interaction level - Depending on the level of user interaction with the system, the technique for requirement elicitation should be different. For example, prototyping can be used if user interaction level is high.

Project complexity - Complex projects usually demand a combination of requirement elicitation techniques as it is difficult to gather requirements in one session alone. Thus, employing different requirement elicitation techniques increases the chances of collecting a variety of requirements. For example, interviews can be used to gather the initial requirements followed by prototyping/workshops to clarify ambiguities.

4. Situational Requirement Method System Tool Description

A tool - Situational Requirement Method System (SRMS) - is developed with PHP and MySQL and hosted by Apache Server. A tool is required that can support the creation, storage, and extraction of methods. The high-level architecture of SRMS is shown in Figure 1.

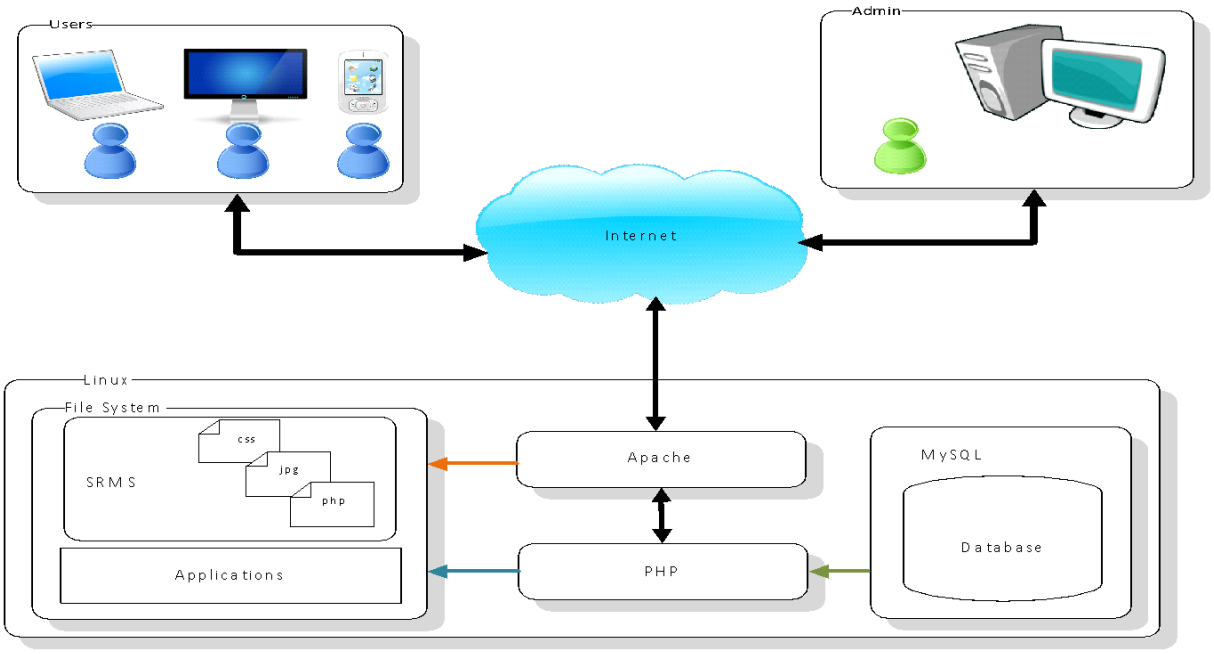


Figure 1. A high-Level Architecture

For our purpose, we preferred a web-based tool so that project members who are in different physical locations are able to access it easily. Typically, users of SRMS can store their experiences with the help of the SRMS tool, as represented in Figure 2.

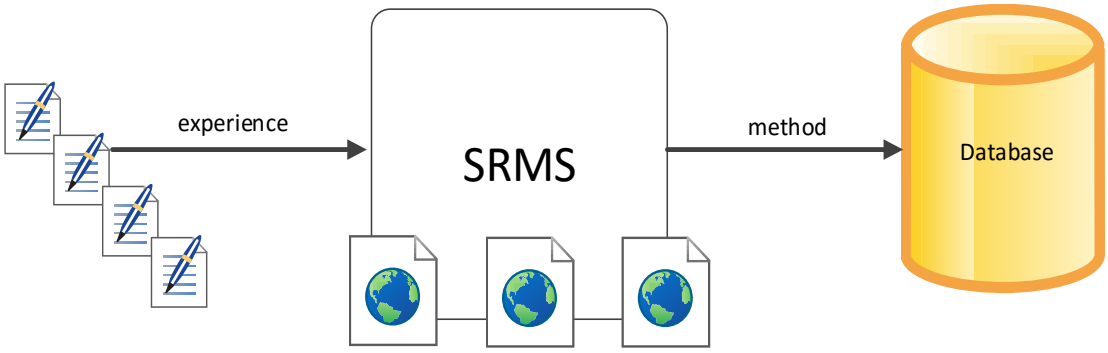


Figure 2. SRMS Tool

SRMS provides method requirements, as already explained in Section 3, that are related with project characteristics. As a result, by using SRMS, users can easily extract the method used in the requirement elicitation phase that is most suitable for their project situation. They can also create new methods according to their previous experience. Moreover, in SRMS, the method requirements that are filled by requirement engineer(s) are defined in advance, and the engineer(s) are expected to fill the complete information so that it can be used in the future.

SRMS has two types of actors: The user and the administrator.

The user can perform the following functions: add new method, add new method by extending an existing method, add new method by combining two methods, view method, update method, see report, view statistics, and get help.

In turn, the SRMS administrator, apart from the tasks performed by the users, can perform the following additional functions: add a new user, set a new password for the user, delete the user,

delete the Method, and delete comment.

First, the user logs into the system by entering their username and password. If the login is successful, then the main menu is displayed according to the user type. The methods that are stored can be viewed by the 'View Method' module. It is important to see all the methods in the repository before creating a new one. In the 'View Method' page, there are method requirements, as shown in Figure 3, which are already explained in Section 3.

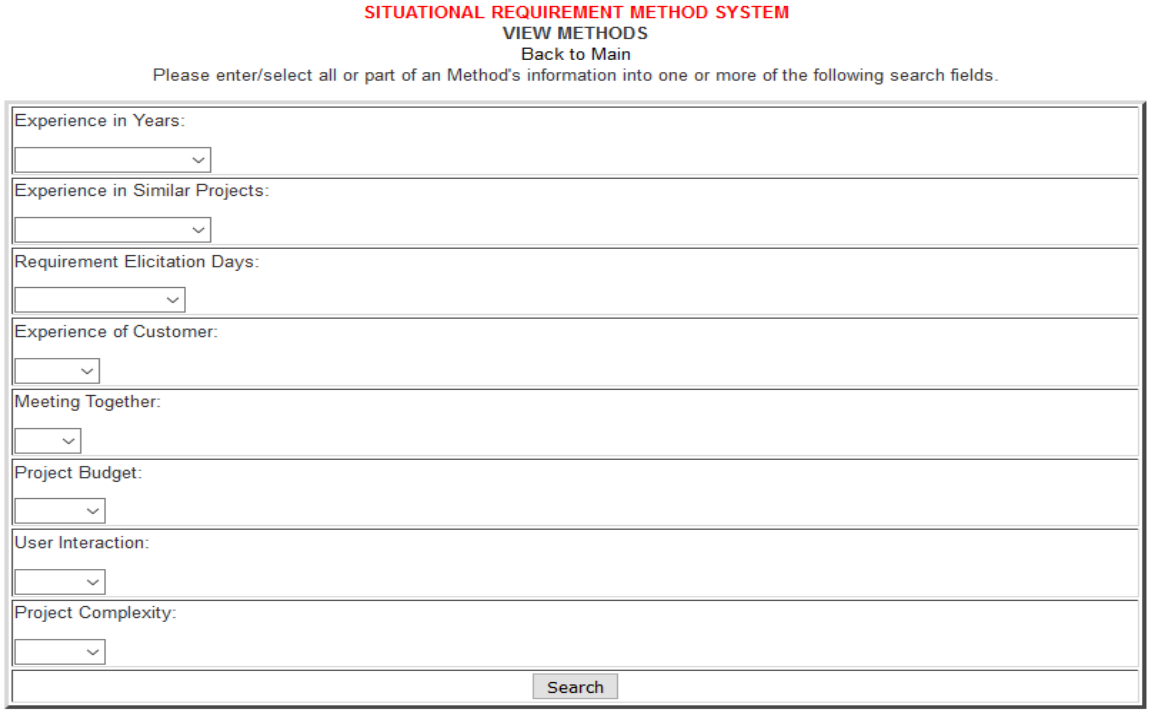


Figure 3. Method requirements' selection screen for view method

Users can search for the desired methods based on these method requirements to best suit the project situation at hand. Requirement engineers can select a value or range for each method requirement from various possible value/range in the system, after which requirement elicitation method(s) matching given method requirements will be displayed underneath as shown in Figure 4. A method consists of a sequence of activities or steps.

SITUATIONAL REQUIREMENT METHOD SYSTEM
VIEW METHODS
Back to Main

You did not state any search criteria, all methods are listed.
Back to View

Click on a Method Name to see the method's detail.

Method Name	Method Comments	Experience in Years	Experience in Similar Projects	Requirement Days	Customer Experience	Meetings Together	Project's Budget	User Interaction	Project's Complexity	Added By
Questionnaire	View/Add	5-10	5-10	morethan1day	75%	yes	middle	middle	middle	secil
Interview	View/Add	5-10	5-10	morethan1day	50%	yes	middle	middle	middle	secil
Quest_Intervi_Mixed	View/Add	0-5	0-5	morethan1day	50%	yes	high	middle	high	secil
TrialOne	View/Add	more10	more10	morethan1day	75%	yes	middle	low	high	secil
Questionnaire_secil	View/Add	5-10	5-10	lessthan1day	25%	yes	middle	high	middle	secil
Use_Cases	View/Add	0-5	0-5	morethan1day	50%	yes	high	middle	high	secil
Workshop	View/Add	5-10	5-10	morethan1day	50%	yes	middle	middle	high	secil
NewMethod	View/Add	0-5	0-5	lessthan1day	25%	yes	middle	low	high	secil
Quest_Inter_Mixed	View/Add	0-5	5-10	morethan1day	50%	yes	middle	middle	middle	secil

Figure 4. View Method Page (search result)

The user can also benefit from the View/Add Comment feature, which allows viewing and addition of important comments about each method. This feature can also be used for specifying the domain of the method. Each method listed in the table has a view/add link to see the

comments made regarding the method. When the user clicks on this link, he/she can view such comments and also add new ones on the same page as shown in Figure 5.

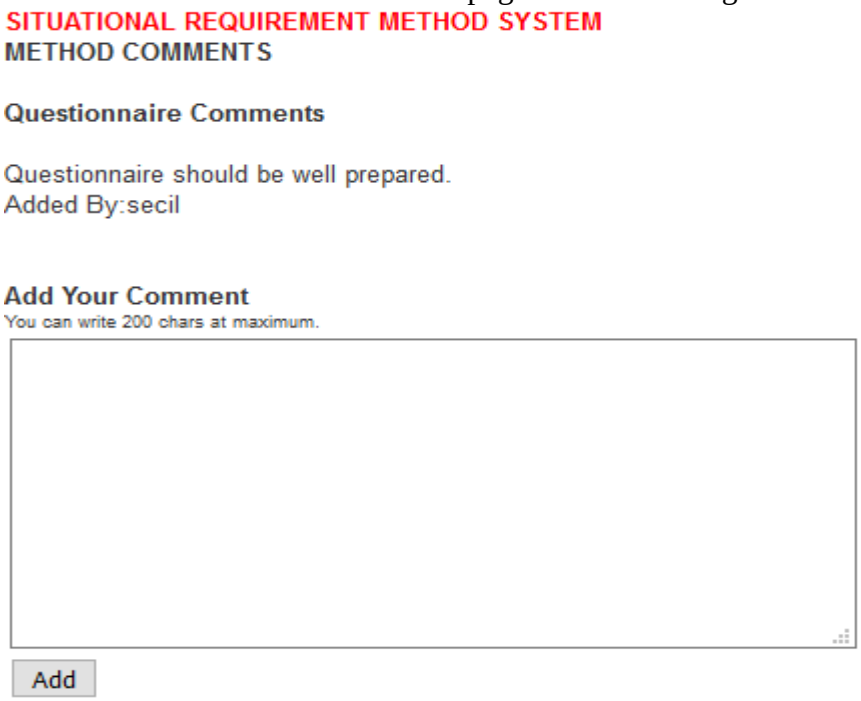
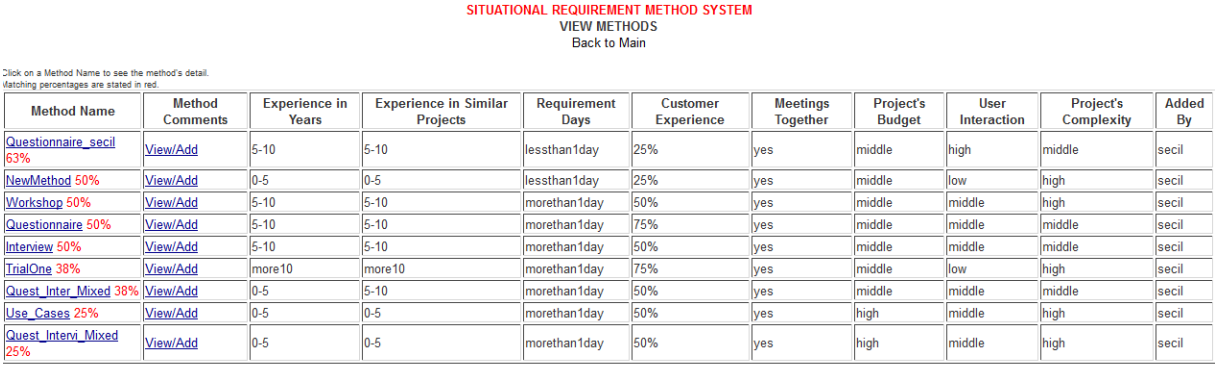


Figure 5. Method View/Add Comment Page

Moreover, selective method requirements instead of all can also be stated, such as experience in years, experience in similar projects, and requirement elicitation days. The results matching only the specified method requirements will be displayed with the percentage of the match. This feature allows the user to see how many of the method requirements have been matched with the user's given requirements as shown in Figure 6.



Method Name	Method Comments	Experience in Years	Experience in Similar Projects	Requirement Days	Customer Experience	Meetings Together	Project's Budget	User Interaction	Project's Complexity	Added By
Questionnaire_secil 63%	View/Add	5-10	5-10	lessthan1day	25%	yes	middle	high	middle	secil
NewMethod 50%	View/Add	0-5	0-5	lessthan1day	25%	yes	middle	low	high	secil
Workshop 50%	View/Add	5-10	5-10	morethan1day	50%	yes	middle	middle	high	secil
Questionnaire 50%	View/Add	5-10	5-10	morethan1day	75%	yes	middle	middle	middle	secil
Interview 50%	View/Add	5-10	5-10	morethan1day	50%	yes	middle	middle	middle	secil
TrialOne 38%	View/Add	more10	more10	morethan1day	75%	yes	middle	low	high	secil
Quest_Inter_Mixed 38%	View/Add	0-5	5-10	morethan1day	50%	yes	middle	middle	middle	secil
Use_Cases 25%	View/Add	0-5	0-5	morethan1day	50%	yes	high	middle	high	secil
Quest_Intervi_Mixed 25%	View/Add	0-5	0-5	morethan1day	50%	yes	high	middle	high	secil

Figure 6. View Method Page (results with matching rates)

When requirement engineers acquire a new project with specific project characteristics, they enter the method requirements as described above and, then, search which methods were used in the past projects in similar situations. The list of methods is displayed along with the percentage of method requirements matched. Suppose that there is a method (e.g. focus group) displayed with 80% match; this means 80% of the project characteristics are the same between the current project and the project in which the displayed method was used successfully. This can help requirement engineers to select the same method for the current project as most of the project characteristics are the same. Therefore that same method can be used successfully again. On the other hand, if the displayed methods have only 20% match in terms of project criteria,

then it is not advisable to use the displayed method in the current project because the current project is vastly different from the project in which the displayed method was used successfully. In such situation, requirement engineer may decide to create a new method for the current project from scratch. This process of creating a new method from scratch is explained in more detail later (figure 7, 8).

Similarly, if there are more than one method (e.g., questionnaire and interview) displayed with approximately 50% match in terms of project criteria, then the requirement engineer may decide to use a combination of both the displayed methods in the current project. For this purpose, he/she can create a new method for the current project by combining two existing methods. This new method can comprise an initial questionnaire to collect the requirements followed by a series of interviews to clarify the ambiguous requirements or add more. Again, this process of creating a new method by combining two methods is described in more detail later (figure 9 to figure 11). Two methods can also be combined to create a more efficient one if the current project requires a more comprehensive requirement phase and has the resources to do so.

If the desired method is not found in the system, then it enables the user to create a new method. There are three different method creation types in our system. A new method can be created by an empty method template (i.e., creating from scratch), by extension (i.e., by adding more steps and/or deleting some steps in an existing method), and by combining two methods (i.e., by selecting activities/steps from two existing methods) stored in the repository. This type of storage provides flexibility to the user to emphasize and transfer his/her experience.

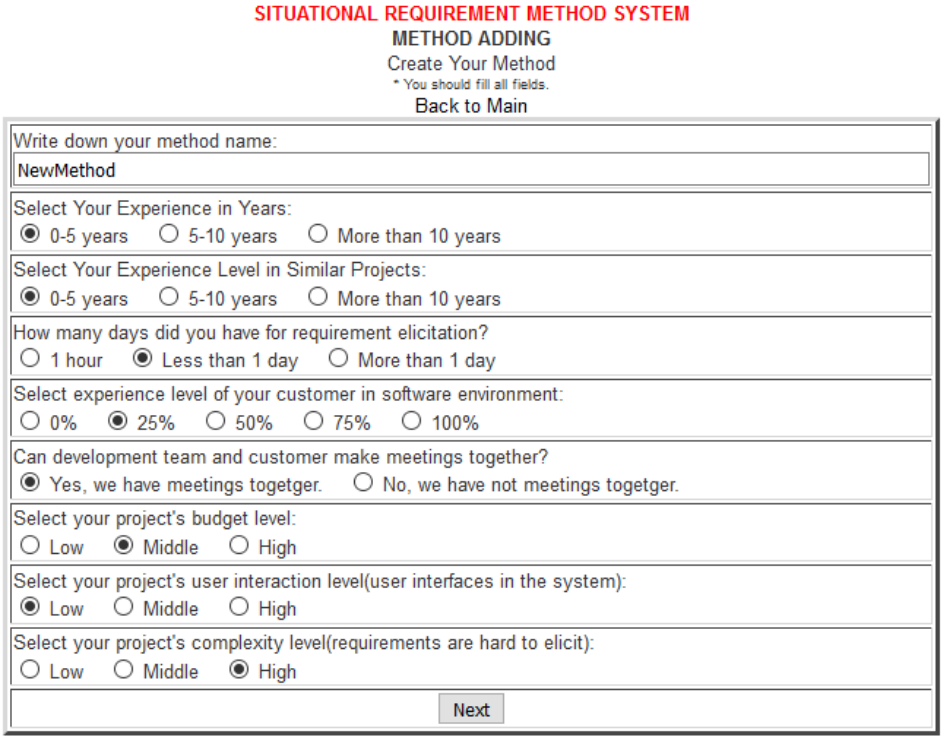


Figure 7. Create a new method from scratch

For creating a new method from an empty method template, users should fill in the method's requirements so as to specify the project's special case as shown in Figure 7. Then, click 'Next' button to save the added steps to the method as shown in Figure 8. The user can add multiple steps to the method by the 'AddMoreStep' button. After adding all the steps, the

'AddAndSaveAll' button is clicked to save the method as shown in Figure 8.

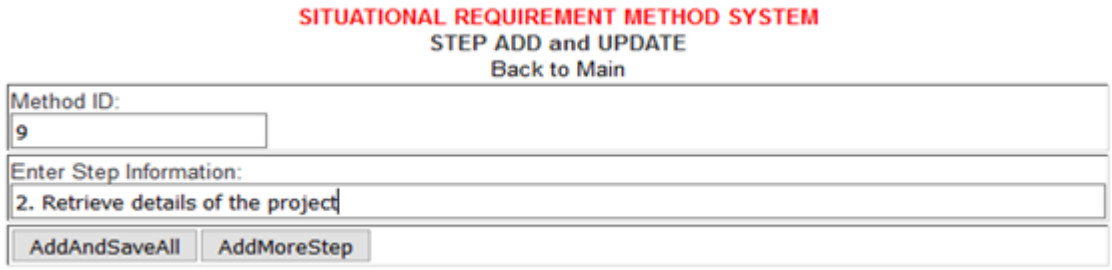


Figure 8. Add steps of the method

After saving the method, it can be viewed from the 'view method' page. Moreover, the user can use this new method to create new methods.

If the user wishes to add a new method by combining two existing methods stored in the system, the 'Add New Method by Combining Two' function is used. When the user clicks on this link, the system first lists all the methods stored in it as shown in Figure 9; then, the user selects a maximum of two methods to be combined. If a method is not selected by the user, he/she is redirected to the main page. If only one method is selected, the system informs the user to select one more. If more than two methods are selected, the system reminds the user to select only two methods.

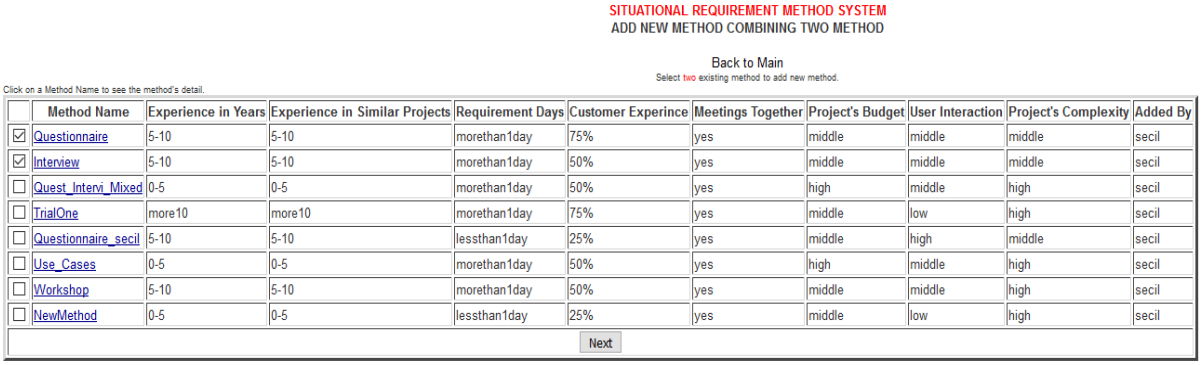


Figure 9. Add New Method by Combining Two (methods listed)

Once the user selects two methods and clicks the 'Next' button, a new page is displayed with the two methods' requirements together with a new blank criteria section. The user can see the two methods' requirements on the same page as shown in Figure 10. The user is required to fill out the blank criteria and assign a name to this new method. If one of the method requirements is left out, the system gives an error and reminder to complete all the requirements. Then, as shown in Figure 10, the user clicks on 'Add New Method' and the values of all the method requirements are added to the repository with the new method's name.

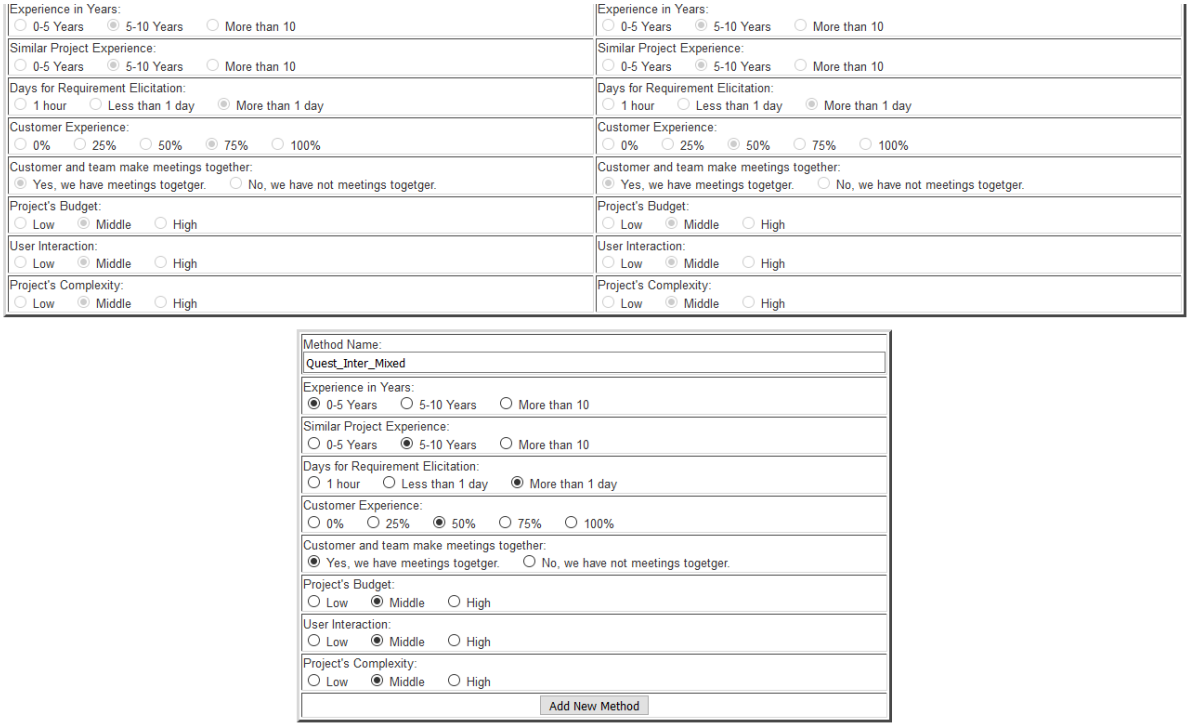


Figure 10. Add New Method By Combining Two (new method requirements filled)

After these method requirements are added to the repository, the steps of the selected two methods are displayed to the user, who is required at this stage to assign numbers to each step in order to add new method as shown in Figure 11.

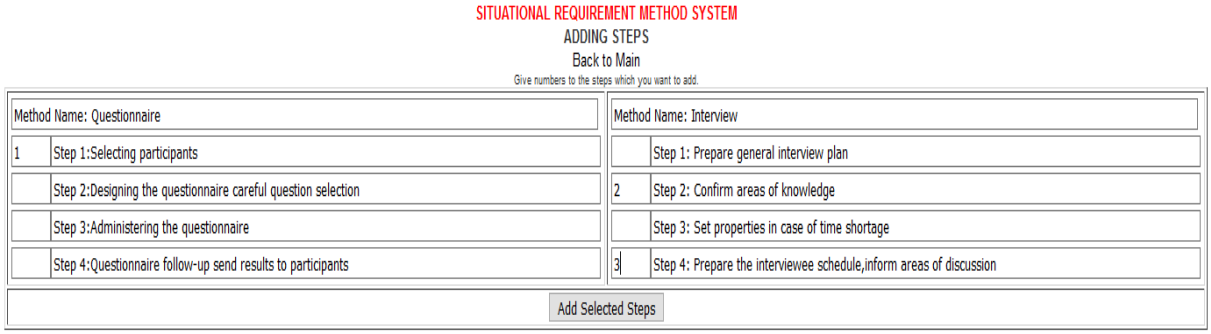


Figure 11. Add New Method By Combining Two (step adding)

When the user clicks 'Add Selected Steps', all the selected steps are added in a numbered sequence as previously assigned by the user to the system repository.

If a user wishes to update an existing method, the 'Update Method' module on the main page can be utilized. Any user can benefit from other author's methods by reusing; however, only the author of each specific method can make changes in that method in the SRMS.

Another feature of SRMS is showing the statistical data of the system. When users click 'View Statistics', five criteria are displayed. They can select the desired criteria to see the related statistics. These criteria are based on:

- The user: It shows the different users and the number of new methods each has created;
- The method type: It shows the number of new methods created with an empty method template, by modifying a method stored in the repository, and by combining two methods stored in the repository separately;

- The user and the method type: It shows the number of new methods created with an empty method template, by modifying a method stored in the repository and by combining two methods stored in the repository with respect to each user;
- The viewing rate: It shows the number of times an existing method has been viewed by users; and
- The usage for creation: It shows the number of times an existing method has been used to create a new one.

5. SRMS Tool Validation and Discussion

5.1 Background

The tool will be used by requirement engineers to create and use effective requirement elicitation methods according to the project situation at hand. As SME techniques are used not only to customize a software development process for a particular project context, but also to perform a continuous process improvement [16][17], it is equally relevant for the software quality assurance group in the organizations.

A questionnaire is designed with the aim to study the effectiveness of the proposed approach and the ease-of-use of the tool. SRMS is logically similar to the LL databases as the latter also keeps a knowledge base for future projects. Many software development organizations use LL databases for this purpose; hence, there is a high likelihood that respondents already have experience in using LL databases. Therefore, another aim of the questionnaire is to compare it with the proposed approach and tool.

The SRMS introduced in the present study was published on the World Wide Web to make it accessible to the respondents of the questionnaire. User accounts were opened for each individual to easily access and evaluate the system. After that, copies of the questionnaire were distributed among these people, who were mainly working in medium or large software development companies (totally 10) and are more likely to have experience with these types of systems. These companies provide solutions for avionics, defense, telecommunications, embedded systems, RFID systems and safety critical systems. These companies have middle (100 to 250) to large (600-900 and plus) size in terms of employees. They mostly believe keeping ‘knowhow’ is valuable for them so their employees were willing to answer our survey. Only those companies were chosen which had quality control groups for process improvement. The respondents are the ones with considerable experience and are, in some way, related to such groups in their organizations. Only a handful of respondents were selected who were believed to have some idea about these types of systems and who were more likely to have encountered LL databases or similar systems previously. This was done in order to receive more reliable feedbacks. In total, 28 out of 35 respondents returned the questionnaire, yielding 80% response rate.

5.2 Research Questionnaire

The following questions were put forth to the participants:
Q1: Have you ever used this type of method engineering system?
Q2: Do you have experience in ‘lessons learned databases’?
Q3(i): If your answered Q.2. as ‘yes’, do you think that SRMS is better than ‘lessons learned databases’?
Q3(ii): If your answer to Q.3(i) is ‘disagree’ or ‘strongly disagree’, please state your reasons in

brief.
Q4: Do you prefer to use SRMS in your intranet system?
Q5: Do you think that SRMS is easy to use?
Q6: Do you think that SRMS's method storage principle is useful?
Q7: Which module do you find most useful?
Q8: Do you like to suggest adding new functions to the SRMS? Please explain.
Q9: Do you think that the criteria used in the system are useful for separating methods situations?
Q10: Do you have any suggestion for adding new criteria to the system? Please explain with the reasons.
Q11: Do you think that the system provides knowledge transfer among project members?
Q12: Specify any other comments or issues.

5.3 Data Analysis and Discussion

In this section, the responses by the participants to each of the questions are analyzed separately.

To perform a statistical analysis of the collected data, it should be emphasized that the sample size n in all questions is small, namely, $n \leq 28$. As a result, Z-tests based on the Central Limit Theorem are not applicable here. Therefore, for all tests concerning proportions, an exact test based on binomial distribution (See, for example, [58], section 3.4) is used. The level of significance is $\alpha = 0.05$ for all tests. To obtain more detailed information on population proportions, it is beneficial to construct 95% confidence intervals for those proportions. An exact confidence interval for a binomial distribution can be obtained with Clopper-Pearson approach. Namely, for (1-α).100% level of confidence, the lower and upper confidence limits can be taken as:

$P_L = Beta^{-1}(\alpha/2 ; k, n-k+1)$, and

$P_U = Beta^{-1}(1-\alpha/2 ; k+1, n-k)$

Where $Beta^{-1}(c; a, b)$ denotes the c-th quantile of a Beta distribution with shape parameters a and b.
The values for P_L and P_U in each case have been obtained by using Table 5.2 in [59]. Furthermore, in some of the questions the needed confidence intervals are one-sided and, in this case, $P_L = \alpha^{1/n}.100\%$, while $P_U = 100\%$, where n is the sample size (see [58], section 3.5). With $\alpha = 0.05$ and $n = 28$, we obtain $P_L = 89.8\%$ as shown in Table 1.

The results of the analysis for each question are as follows:

Q1. 82.1% of the respondents stated that they have not used this type of method storage system (like SRMS) before. Only a few of the respondents had used this type of system previously because there are limited tools implementing the situational method engineering approach which are merely for research purposes.

Q2. 75% of the respondents stated that they have used the 'lessons learned databases' before.

Table 1: 95% confidence intervals for population proportions

Q	'Agree' or 'Strongly agree'				'Strongly agree'			
	No. of Responses n_i	k_i	P_L	P_u	No. of Responses n_i	k_i	P_L	P_u
3(i)	21	16	52.9%	91.8%	21	2	1.2%	30.4%
4	28	20	51.3%	86.8%	28	8	13.2%	48.7%
5	28	28	89.8%	100%	28	16	37.3%	75.5%
6	28	28	89.8%	100%	28	6	8.3%	41%
9	28	28	89.8%	100%	28	8	13.2%	48.7%
11	28	28	89.8%	100%	28	18	44%	81.4%

Q3(i). According to the responses, in a sample of 21 users who have experience in 'lessons learned databases', 16 believe i.e., 'strongly agree' or 'agree' that SRMS is better. To confirm that this data provides necessary statistical evidence that more than 50% of 'lessons learned databases' users prefer SRMS, the following test on the population proportion has been conducted:

H_0: $p \leq 50\%$

H_1: $p > 50\%$

The level of significance is $\alpha = 0.05$.

The p-value of the test is $P\{X \geq 16 \mid n=21, p_0 = 0.5\} = 0.0133$. Since $0.0133 < \alpha$, the null hypothesis will be rejected. It can be concluded, therefore, that the majority of 'lessons learned databases' users believe i.e., 'strongly agree' or 'agree' that SRMS is better.

Finding a confidence interval, we observe that the actual percentage of users who believe that SRMS is better than 'lessons learned databases' is between 52.9% and 91.8% with 95% level of confidence.

Q3(ii). It is found that a majority of the respondents agree that SRMS is better than the 'lessons learned databases' while a few among the rest stated that they already have a more customized tool than SRMS. The respondents did not provide much detail about the tool they are using. The respondents also stated that in the 'lessons learned databases', the topics are organized in a tree-view format, meaning that the relationships among the topics can be observed easily; yet, in SRMS the methods are organized in a flat style. Another suggestion was that it would be helpful to see which methods are extended – in other words, not created from an empty template - and also the source method used in the extension. It is possible to incorporate this suggestion. In the background of SRMS, it has been stored whether the method is new, extended, or combined. For the methods that are extended from the existing method, an additional field storing the source method name can be added to incorporate this suggestion.

Q4. First, we test whether the majority ('strongly agree' or 'agree') of users prefer SRMS in the intranet system. The following test concerning the population proportion of users who prefer SRMS is conducted:

H_0: $p \leq 50\%$

H_1: $p > 50\%$

In this case, the p-value can be estimated as

$P\{X \geq 20 \mid n=28, p_0 = 0.5\} \leq P\{X = 20\} + 8P\{X = 21\} = 0.046 < \alpha$.

Therefore, the null hypothesis is rejected, and it is confirmed that a majority of users prefer SRMS in the intranet system.

Now, to construct a 95% confidence interval for the true population proportion of users who prefer SRMS, by Table 5.2 from [59], one has:

$P_L = 51.3\%, \quad P_U = 86.8\%$

That is, the actual percentage of users who prefer SRMS in their intranet system is between 51.3% and 86.8% with 95% level of confidence.

Q5. Here, we have to consider a one-sided confidence interval for the proportion of people who claim i.e., 'strongly agree' or 'agree' that SRMS is easy to use - since, in this case, k = n = 28 (sample size). As it has already been mentioned $P_U = 100\%$ and $P_L = \alpha^{1/n}$ (see [58], sec 3.5) with $\alpha = 0.05$, yielding $P_L = \alpha^{1/28} \approx 89.8\%$.

Therefore, we state that at 95% level of confidence, the actual percentage of users who 'strongly agree' or 'agree' that SRMS is easy to use is between 89.8% and 100%.

Meanwhile, the test concerning 'strongly agree' can be stated as:
H_0: $p \leq 50\%$
H_1: $p > 50\%$

Since the p-value $P\{X \geq 16\} > P\{X = 16\} = 0.1133 > \alpha$, the null hypothesis cannot be rejected and, hence, there is no sufficient evidence that the majority of users 'strongly agree' that SRMS is easy to use.

Q6. In SRMS, methods are stored in two parts. The first part includes the characteristics of that method and, then, the second part contains the steps of that method. The respondents were questioned to learn about the usefulness and effectiveness of SRMS's method storage principle. As before, we have to consider a one-sided confidence interval for those proportion of people who think i.e., 'strongly agree' or 'agree' that SRMS's method storage principle is useful - in this case, k = n = 28 (sample size). As have been mentioned previously, this leads to $P_L = 89.8\%$ and $P_U = 100\%$.

Therefore, it can be claimed that at 95% level of confidence interval, the actual percentage of users who 'strongly agree' or 'agree' that SRMS's method storage principle is useful is above 89.8%. On the other hand, the actual percentage of users who 'strongly agree' that SRMS's method storage principle is useful with 95% level of confidence is within the interval 8.3% and 41%.

Q7. Most of them found that 'add new method' (either entirely new, or by extending an existing method, or by combining two existing ones) is the most useful module. The second most popular module was found to be the 'View Statistics' module which is very useful to learn which methods are mostly viewed or used for creating new methods implying that it can easily affect the usage and creation of a method.

Q8. Although SRMS is found to be an easy and useful system by most of the users, we examined if there is any missing function which the respondents would like to have in this type of system. Around half of the respondents stated that the tree-view function showing relationships can be added to the system. Some of the respondents also suggested that, while adding a new method to the system, apart from adding the steps of the methods some additional information – such as how to use the method and its purpose - should also be entered.

Q9. As mentioned in Section 3, the methods have been categorized based on eight criteria. The authors inquired whether the given criteria are sufficient when separating the methods according to the projects' situations. It is established that at 95% confidence interval for the proportion of users who think i.e., 'strongly agree' or 'agree' that criteria are useful for separating is between 89.8% and 100%. This demonstrates that, on the whole, the users think that criteria are useful.

However, in this case, the actual users who 'strongly agree' that criteria are useful is within the interval 13.2% and 48.7% at 95% level of confidence, which indicates that SRMS may need to incorporate more criteria for method categorization.

Q10. Users were asked to provide suggestions regarding the criteria for categorizing methods. Some suggested that another criterion, 'acquaintance level of the customer to the project domain', can also be added to the system because, even if the customer is the owner of the project, he/she may not have enough know-how in the application domain. This is generally the case in outsourced projects. Although there is a similar existing criterion in SRMS as 'experience level of your customer', this criterion is found to be too general and also, it is possible to have extensive experience as a customer in the software development field and, yet, possess little experience with a particular project domain. Another suggestion was that it would be helpful for users to be able to specify ranges for budget, interaction, and complexity levels - which was found to be constructive recommendation and, thus, can be incorporated.

Q11. Nowadays, knowledge transfer is the most valuable activity for organizations, and users were asked whether SRMS allows for knowledge transfer among project members. In this case, at 95% confidence level, the proportion of users who think i.e., 'strongly agree' or 'agree' that SRMS provides knowledge transfer is between 89.8% and 100%, indicating that this opinion is shared by the majority of users and, as such, it can be valuable if used effectively by team members.

Let us draw the attention to the users who 'strongly agree' that the system provides knowledge transfer. A 95% confidence interval for 'strongly agree' is within the bounds of 44% and 81.4%, allowing one to assume that a majority of users 'strongly agree' that the system enables knowledge transfer. As before, to check this statement rigorously, an exact test for proportions based on the binomial distribution is applied. Select

H_0: $p \leq 50\%$

H_1: $p > 50\%$.

With n=28 and k= 18, one has: $P\{X \geq 18\} > P\{X = 18\} + P\{X = 19\} = 0.048 + 0.025 > \alpha$. Therefore, we accept H_0 and there is no evidence suggesting that a majority strongly agrees that the system enables knowledge transfer.

Q12. The descriptions of modules are already defined in the 'Help' segment of SRMS. The respondents also suggested that a feedback system from the user can be added - that is, a rating from each user who is using a particular method. It was stated that it can be very beneficial to add this feature to the system so the users can easily view the ratings of the methods. Some respondents recommended that association between processes and SRMS methods should be allowed, what's more, it was further noted that keeping track of which methods are used in which projects, the results, conditions and the people involved, can be helpful as well. At this point, it should be stated that the characteristics of a project are important; yet, the main motivation is storing methods not project management related data.

6. Conclusion

In software development organizations, software developers are adapting software development methods according to different project conditions. There is a need to share this know-how with other developers, who may reuse it in similar contexts towards saving the efforts thus reducing costs and time, which are crucial in software development. A systematic approach to method tailoring and storage can solve this problem and developers can learn from each other's experiences, thereby enhancing their productivity as well. The present paper develops a simplified approach for method storage and retrieval according to project characteristics with the help of a web-based tool, SRMS. Here, new methods can be developed through inception, extension or assembling existing methods. Lessons learned databases are, by far, closest to SRMS in terms of objectives and context, aside from their usefulness to the project managers working in the industry in terms of similar projects context.

The proposed approach and tool was validated by distributing SRMS-related questionnaires to software professionals working in large software companies. Moreover, SRMS was published online to make it accessible to the respondents of the questionnaire. A majority of respondents had not used such type of systems earlier, but they had extensive experience with 'lessons learned databases'. A majority of users found SRMS better than 'lessons learned databases' and stated their preference for using it. It has been also established that they consider SRMS easy-to-use and its method storage principle to be useful. They agree that it provides knowledge transfer among project members. Some respondents provided various general suggestions to improve it further, whereas there were a few regarding the criteria for methods' categorization. These are:

- Which methods are extended - that means, not created from scratch - and the source method used in the extension.
- While entering a new method, information such as how to use the method and its purpose should be described.
- Add a tree-view function that shows the relationships
- A new criterion 'acquaintance level of the customer to the project domain' should be added. It can replace the existing criterion 'experience level of your customer' because the current one is too general.
- Specify ranges for project criteria such as project's budget, complexity and level of user's interaction instead of discrete values.

All of the above suggestions are valuable for tool improvement and therefore, can be incorporated into the system. Some of the respondents also asked for permission to try SRMS in their projects, so that it can be tried and tested in real environment. According to their feedbacks, SRMS can be modified to match the needs of real-life projects. Since it focuses on the requirements engineering phase, with little modifications, it can also be customized and used in other stages of software development.

References

[1] P. Carreteiro, J.B. de Vasconcelos, A. Barao, A. Rocha (2016). A Knowledge Management Approach for Software Engineering Projects Development. New Advances in Information Systems and Technologies, Springer International Publishing, 59-68.

[2] A.C.C. Natali, R.D.A. Falbo (2002). Knowledge Management in Software Engineering Environments. Proc. of the 16th Brazilian Symposium on Software Engineering, Gramado, Brazil. pp. 238–253.

[3] M. Alavi, D.E. Leidner (2001). Review: knowledge management and knowledge management systems: conceptual foundations and research issues. MISQ, 25(1), 107–136.

[4] I. Rus, M. Lindvall, C. Seaman, and V. Basili (2003). Packaging and Disseminating Lessons Learned from COTS-Based Software Development. Proceedings of the 27 th Annual NASA Goddard/IEEE Software Engineering Workshop, pp. 131-138.

[5] M.I. Hisatomi, A.D.S. Góes, R.M. de Barros (2013). Applying Questionnaire to Assess the Lessons Learned Process in Software Project Management: a Case Study at GAIA. ICSEA 2013 : The Eighth International Conference on Software Engineering Advances, 258-264.

[6] PMI (2013). A Guide to the Project Management Body of Knowledge (PMBoK® Guide), Fifth ed. Newtown Square, USA: Project Management Institute.

[7] M. Marlin (2008). Implementing an Effective Lessons Learned Process in a Global Project Environment, UTD 2nd Annual Project Management Symposium Proceedings, Dallas, Texas available at http://www.westney.com/wp-content/uploads/2014/05/Implementing-an-Effective-Lessons-Learned-Process-In-A-Global-Project-Environment.pdf

[8] R. Weber, D.W. Aha, I. Becerra-Fernández (2001). Intelligent lessons learned systems. Expert System with Applications, 20, 17–34.

[9] S. Brinkkemper (1996). Method Engineering: Engineering of Information Systems Development Methods and Tools. Inf. Software Technol., 38(4), 275–280.

[10] D. Mirandolle, I. van de Weerd, S. Brinkkemper (2011). Incremental Method Engineering for Process Improvement - A Case Study, 4th IFIP WG 8.1 Working Conference on Method Engineering, ME 2011, Paris, France, April 20-22, 4-18.

[11] F. Karlsson, P. Ågerfalk (2012). MC Sandbox: Devising a tool for method-user-centered method configuration. *Inf. Softw. Technol.* 54, 5 (May 2012), 501-516.

[12] J. Ralyte, R. Deneckere, C. Rolland (2003). Towards a Generic Model for Situational Method Engineering, Proceedings of CAISE03, 15th International Conference on Advanced Information Systems Engineering,. Lecture Notes in Computer Science, Vol. 2681 (2003), 95-110.

[13] J.A. Hurtado, M.C. Bastarrica, S.F. Ochoa, J. Simmonds (2013). MDE software process lines in small companies. *J. Syst. Softw.* 86, 5 (May 2013), 1153-1171.

[14] B. Henderson-Sellers, J. Ralyté (2010). Situational method engineering: state-of-the-art review, Journal of Universal Computer Science 16 (3) (2010) 424–478.

[15] J.P. Tolvanen, M. Rossi, H. Liu (1996). Method engineering: current research directions and implications for future research. In: Proceedings of the IFIP TC8, WG8.1/8.2 Working Conference on Method Engineering: Principles of Method Construction and Tool Support. Chapman & Hall Ltd., London, UK, pp. 296–317.

[16] M. Bajec, D. Vavpotič, S. Furlan, M. Krisper (2007). *Software process improvement based on the method engineering principles*, pp. 283-297, Springer US.

[17] M. Bajec, D. Vavpotič, M. Krisper (2007). Practice-driven approach for creating project specific software development methods. Information and Software Technology, 49 (4), pp. 345–365.

[18] Z.S.H. Abad, A. Alipour, R. Ramsin (2012). Enhancing Tool Support for Situational Engineering of Agile Methodologies, R. Lee (Ed.): Software Engineering Research, Management and Appl. 2012, SCI 430, pp. 141–152.

[19] M. Cervera, M. Albert, V. Torres, V. Pelechano (2012). Turning Method Engineering Support into Reality, J. Ralyté, I. Mirbel, and R. Deneckère (Eds.): ME 2011, IFIP AICT 351, pp. 138–152.

[20] A. Niknafs, R. Ramsin (2008). Computer-Aided Method Engineering: An Analysis of Existing Environments, Advanced Information Systems Engineering, Volume 5074, Springer Berlin/Heidelberg, pp. 525-540..

[21] Z.S.H. Abad, A. Alipour, R. Ramsin (2010). Towards Tool Support for Situational Engineering of Agile Methodologies. In: Proc. Asia-Pacific Software Engineering, Conference (APSEC 2010), 326–335.

[22] D. Gupta, N. Prakash (2001). Engineering Methods from Method Requirements Specifications. Requir. Eng.

6(3), pp 133-160.

[23] B. Zvanut, M. Bajec (2010). A tool for IT process construction. Information and Software Technology, 52 (4), pp. 397–410.

[24] S. Aydin, D. Mishra (2009). A tool to enhance cooperation and knowledge transfer among software developers. In *Proceedings of the 6th international conference on Cooperative design, visualization, and engineering* 2009 (CDVE'09), Yuhua Luo (Ed.). Springer-Verlag, Berlin, Heidelberg, pp. 257-260.

[25] D. Mishra, S. Aydin, A. Mishra (2012). Situational Requirement Method System: Knowledge Management in Business Support. New Trends in Databases and Information Systems, Workshop Proceedings of the 16th East European Conference, ADBIS 2012, Poznań, Poland, September 17-21, pp. 349-359.

[26] A. Aurum, F. Daneshgar, J. Ward (2008). Investigating knowledge management practices in software development organisations – an Australian experience. Information and Software Technology, 50, pp. 511–533.

[27] P. Secchi, R. Ciaschi, D. Spence (1999). A concept for an ESA lessons learned system. in: Proceedings of Alerts and LL: An Effective Way to Prevent Failures and Problems, The Netherlands, pp. 57–61.

[28] F. Harmsen, S. Brinkkemper, H. Oei (1994). Situational Method Engineering for Information System Project Approaches. Methods and Associated Tools for the Information System Life Cycle, A.A. Verrijn-Stuart and T.W. Olle (Eds), Elsevier Science, pp. 169-194.

[29] A. Aamodt, E. Plaza (1994). Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communications. IOS Press, Vol. 7 (1), pp. 39-59.

[30] S. H. Park, D.H. Bae (2013). Tailoring a large-sized software process using process slicing and case-based reasoning technique. *IET Software*, vol. 7, no. 1, pp. 47-55.

[31] J. Kato, S. Komiya, M. Saeki, A. Ohnishi, M. Nagata, S. Yamamoto, H. Horai (2001). A Model for Navigating Interview Processes in Requirements Elicitation. In *Proceedings of the Eighth Asia-Pacific on Software Engineering Conference* (APSEC '01). IEEE Computer Society, Washington, DC, USA, pp. 141-148.

[32] J. Andrade, J. Ares, M.Au. Martínez, J. Pazos, S. Rodríguez, J. Romera, S. Suárez (2013). An architectural model for software testing lesson learned systems. Information and Software Technology, Volume 55, Issue 1, pp. 18-34, ISSN 0950-5849, http://dx.doi.org/10.1016/j.infsof.2012.03.003.

[33] A. Vizcaino, J. Favela, M. Piattini (2003). A Multi-agent System for Knowledge Management in Software Maintenance. In: Palade V., Howlett R.J., Jain L. (eds) Knowledge-Based Intelligent Information and Engineering Systems. KES 2003, pp. 415-421, Lecture Notes in Computer Science, vol 2773. Springer.

[34] C. Rolland, N. Prakash, A.Benjamen (1999). A multi-model view of process modeling. Requirements Engineering, Vol. 4, No. 4., pp. 169-187.

[35] B. Henderson-Sellers, J. Ralyte (2010). Situational Method Engineering: A state of the art review. Journal of Universal Computer Science 16(3), pp. 424-478.

[36] K. Kumar, R.J. Welke (1992). Methodology Engineering: a proposal for situation-specific methodology construction. In: Challenges and Strategies For Research in Systems Development, pp. 257–269. John Wiley & Sons, Inc., Chichester.

[37] P. Agerfalk, S. Brinkkemper, C. Gonzalez-Perez, B. Henderson-Sellers, F. Karlsson, S. Kelly, J. Ralyté (2007). Modularization Constructs in Method Engineering: Towards Common Ground? In: Ralyté, J., Brinkkemper, S., Henderson-Sellers, B. (eds.) Situational Method Engineering: Fundamentals and Experiences, vol. 244, pp. 359–368. Springer, Boston.

[38] A. Iacovelli, C. Souveyet (2011). Towards Common Ground in SME: An Ontology of Method Descriptors. In J. Ralyté, I. Mirbel & R. Deneckère (eds.), *ME*, pp. 77-90, Springer. ISBN: 978-3-642-19996-7

[39] S.J. Casare, A. Brandao, Z. Guessoum, J.S. Sichman (2014). Medee Method Framework: a Situational Approach for Organization-Centered MAS, Journal of Autonomous Agents and Multi-Agent Systems, 28(3), pp. 430-473.

[40] J. Ralyte, S. Brinkkemper, B. Henderson-Sellers (eds.) (2007). *Situational method engineering: fundamentals and experiences*. Proceedings of the IFIP WG 8.1 Working Conference, 12–14 September 2007, Geneva, Switzerland, IFIP Series, Vol. 244, Springer, Berlin, pp. 380-.

[41] B. Henderson-Sellers, C. Gonzalez-Perez, J. Ralyte (2008). Comparison of Method Chunks and Method Fragments for Situational Method Engineering. In Proceedings of the 19th Australian Conference on Software Engineering (ASWEC 2008), Mar.26-28, IEEE Computer Society, Washington, DC, pp. 479-488.

[42] B. Henderson-Sellers, C. Gonzalez-Perez (2011). Towards the use of granularity theory for determining the size of atomic method fragments for use in situational method engineering. *4th Working Conference on Method Engineering (ME)*, Apr 2011, Lisbon, Portugal. Springer, IFIP Advances in Information and Communication Technology, AICT-351, pp.49-63.

[43] B. Nuseibeh, S. Easterbrook (2000). Requirements engineering: a roadmap. In *Proceedings of the Conference on the Future of Software Engineering*, pp. 35-46.

[44] A.M. Hickey, A.M. Davis (2004). A unified model of requirements elicitation. *Journal of Management Information Systems*, *20*(4), pp. 65-84.

[45] F. Anwar, R. Razali (2012). A Practical Guide to Requirements Elicitation Techniques Selection-An Empirical Study. *Middle-East Journal of Scientific Research*, *11*(8), pp.1059-1067.

[46] D. Mishra, A. Mishra, A. Yazici (2008). Successful requirement elicitation by combining requirement engineering techniques. In *Applications of Digital Information and Web Technologies, 2008. ICADIWT 2008*, pp. 258-263. IEEE.

[47] K. Van Slooten, B. Hodes (1996). Characterizing IS development projects. In *Method Engineering*, pp. 29-44, Springer US.

[48] C. Rolland, N. Prakash (1996). A proposal for context-specific method engineering. In *Method Engineering*, pp. 191-208, Springer US.

[49] S. Brinkkemper, M. Saeki, F. Harmsen (1999). Meta-modelling based assembly techniques for situational method engineering, Information Systems, 24(3), pp. 209–228.

[50] C. Rolland, V. Plihon, J. Ralyté (1998). Specifying the Reuse Context of Scenario Method Chunks, in Advanced Information Systems Engineering, in: B. Perniciand C. Thanos (Eds.), 10th International Conference CAiSE'98, Pisa, Italy, June 8–12, pp. 191-218.

[51] J. Cameron (2002). Configurable development processes, Communications of the ACM, 45(3), pp. 72–77.

[52] B. Henderson-Sellers (2002). Process metamodelling and process construction: examples using the OPEN Process Framework (OPF), Annals of Software Engineering, 14 (1–4), pp. 341–362.

[53] R. Deneckere, A. Iacovelli, E. Kornyshova, C. Souveyet (2008). From method fragments to method services. In: The 13th Workshop on Exploring Modeling Methods for Systems Analysis and Design (EMMSAD'08) held in conjunction with the CAiSE'08 Conference, Montpellier, France, pp. 80-96.

[54] C. Rolland (2009). Method engineering: towards methods as services. Software Process Improvement and Practice, 14(3), pp. 143–164.

[55] B. Fitzgerald, N.L. Russo, T. O'Kane (2003). Software development method tailoring at Motorola. Communications of the ACM, 46 (4) (2003), pp. 65–70

[56] C. Janiesch (2010). Situation vs. context: considerations on the level of detail in modelling method adaptation. in: Proceedings of the 43rd Hawaii International Conference on System Sciences (HICSS 2010), IEEE Computer Society, pp. 1–10.

[57] E. Kornyshova, R. Deneckère, C. Salinesi (2007). Method chunks selection by multicriteria techniques: an extension of the assembly-based approach. In *Situational Method Engineering: Fundamentals and Experiences*, pp. 64-78, Springer US.

[58] K. Krishnamoorthy (2006): Handbook of statistical distributions with applications Chapman & Hall/CRC, Boca Raton FL, pp. 346.

[59] L.N. Bolshev, N.V. Smirnov (1983). Mathematical Statistical Tables (in Russian), 3rd Edition, Nauka, Moscow, 1983.

Appendix

Table 2. Results of the survey

Questions	Options						
1. Have you ever used this type of method engineering system?	Yes **5**			No **23**			
2. Do you have experience in 'lessons learned databases'?	Yes **21**			No **7**			
3(i). If your answered Q.2. as 'yes', Do you think that SRMS is better than 'lessons learned databases'?	Strongly Agree **2**	Agree **14**		Disagree **5**	Strongly Disagree **0**		
3(ii). If your answer to Q.3. is 'disagree' or 'strongly disagree', please state your reasons in brief.	Reason Available 1			Reason Not Available **4**			
4. Do you prefer to use SRMS in your intranet system?	Strongly Agree **8**	Agree **12**		Disagree **8**	Strongly Disagree **0**		
5. Do you think that SRMS is easy to use?	Strongly Agree **16**	Agree **12**		Disagree **0**	Strongly Disagree **0**		
6. Do you think that SRMS method storage principle is useful?	Strongly Agree **6**	Agree **22**		Disagree **0**	Strongly Disagree **0**		
7. Which module do you find most useful?	Add New Method **5**	Add New Method By Extending **3**	Add New Method By Combining **6**	View Methods **2**	Update Method **1**	Take Report **4**	View Statistics **7**
8. Do you like to suggest adding new functions to the SRMS? Please explain.	Suggestions Available **14**			Suggestions Not Available **14**			
9. Do you think that the criteria used in the system are useful for separating methods' situations?	Strongly Agree **8**	Agree **20**		Disagree **0**	Strongly Disagree **0**		
10. Do you have any suggestion for adding new criteria to the system? Please explain with the reasons.	Suggestions Available **7**			Suggestions Not Available **21**			
11. Do you think that the system provides knowledge transfer among project members?	Strongly Agree **18**	Agree **10**		Disagree **0**	Strongly Disagree **0**		
12. Specify any other comments or issues.	Comment Available **5**			Comment Not Available **23**			