International Journal of
Information Systems and Project Management
ISSN (print):2182-7796, ISSN (online):2182-7788, ISSN (cd-rom):2182-780X
Available online at ijispm.sciencesphere.org

Improving the evaluation of change requests using past cases

Otávio da Cruz Mello
Programa de Pos-Graduação em Ciência da Computação, Federal University of Santa Maria
Av. Roraima 1000, Santa Maria, 97105-900
Brazil
odmello@inf.ufsm.br

Lisandra Manzoni Fontoura
Programa de Pos-Graduação em Ciência da Computação, Federal University of Santa Maria
Av. Roraima 1000, Santa Maria, 97105-900
Brazil
lisandra@inf.ufsm.br

Abstract:
As one of the leading causes of project failures, requirements changes are inevitable in any software project. Hence, we propose an intelligent approach to facilitate the risk analysis of a change request by providing information about past cases found in similar change requests, the solutions adopted, and a support tool. The proposed approach uses case-based reasoning to retrieve previous cases similar to the current case. This approach also uses association rules to analyze patterns in the dataset and calculate the probability of risks associated with change requests. We prepared a case study to validate the proposal by analyzing the most frequent challenges in change management and considering how it can solve or minimize such problems. Results show that the proposed approach successfully assists decision-making, predicts potential risks, and suggests coherent solutions to the user. We have developed a support tool to evaluate this approach in practice with experts and obtained four different outcomes. Only a small set of cases failed to provide relevant results to the user. The use of case-based reasoning and association rules has proven to be advantageous in change management despite validity threats associated with the small number of test cases and experts involved.

Keywords:
change management process; change request; risk management; case-based reasoning; association rules.

DOI: 10.12821/ijispm110104

Manuscript received: 3 June 2022
Manuscript accepted: 20 November 2022

1. Introduction

Requirements changes are inevitable to satisfy the project stakeholders' needs and goals. As software evolves, this scenario becomes a problem. Unplanned changes can result in inconsistencies and errors, requiring the work to be redone and the system to be redesigned. This situation causes an increase in costs and the need for deadline rescheduling and team task reallocation. Apart from additional development phases, the team needs to test to ensure the software will work with the changes made [1]. Requirements change is one of the primary causes of failure in software projects [2]. Therefore, defining a systematic process ensuring that changes are controlled from request to delivery and that stakeholders are informed about the progress and impacts caused by them is necessary. Andrew et al. [3] performed a Systematic Literature Review to identify the most significant studied challenges of the Requirements Change Management (RCM) process. According to the authors, the most cited challenge is impact analysis, and cost and time estimates are also critical aspects. Therefore, searching for techniques that facilitate these tasks for professionals in the area is necessary.

Learning from past experiences is common in software engineering [4][5] and is particularly useful in facilitating cost and risk analysis activities [6]. From another perspective, this learning dynamic takes time and usually provides subjective results, so artificial intelligence (AI) techniques can improve it [4][5]. Intelligent systems have assisted human beings in performing tasks by replicating a person's reasoning to solve problems [7]. AI's relationship with software engineering subareas, such as change management, tends to generate beneficial results [8]. Software engineering is a knowledge-based area, but professionals have to deal with uncertainty most of the time. With the aid of techniques and algorithms provided by AI, such uncertainty can be reduced [4]. In RCM processes, AI techniques have improved tools that support managers and software engineers [8].

An intelligent tool capable of predicting and evaluating possible risks of incorporating a new change in a project can help professionals make a better choice when deciding, thereby minimizing the negative impacts or the probability of problems associated with risks caused by a change request.

Given these considerations, our goal is to propose an intelligent approach that facilitates the risk analysis of a change request by providing information about past problems encountered in similar change requests, the solutions adopted, and a tool to support it. This approach uses Case-Based Reasoning (CBR) and association rules techniques.

Change requests are represented as cases and stored on a case base. When evaluating a new change request, the manager can retrieve similar cases to reuse past information as identified risks and associated estimates. We chose to use the CBR technique as it is suitable for decision support systems for helping to identify possible problems and how they can be solved through past case retrieval. In addition, this technique is suitable for the goals of this research owing to its particularities, such as low-cost incremental learning and recovery of solutions in a language understandable by the user based on the human logic of solving problems [9][10].

We use a hybrid approach between CBR and Association Rule Mining (ARM), a popular technique for knowledge discovery in data mining, to improve the accuracy of the results when evaluating the risks of a change. The integration of both can provide positive results, particularly in complex applications that do not have a large case base, making it difficult for the case retrieval algorithm to obtain relevant results. Therefore, ARM can be a technique to discover interesting relations between attributes in the case base and thus enhance the problem-solving capabilities of CBR through the analysis of these associations [11][12][13].

In the validation, according to the literature, we focused on evaluating whether the proposed approach helps the team solve the most common problems in change management. We invite project developers to assess change requests using our approach to achieve this goal. They entered project change requests and provided feedback on the responses' usefulness.

The paper is organized as follows. Section 2 describes background information about the change management process and AI techniques, and Section 3 presents the related work. Section 4 details a proposed approach and its activities. Section 5 describes the research validation methods and discusses the results obtained. Finally, Section 6 concludes the study by describing the final considerations and discussing future works.

2. Background

This section briefly describes the concepts necessary to understand the rest of the article.

2.1 Requirement change management

Changes are inevitable in any project. Users are unlikely to be able to define all software requirements at the beginning of development. Hence, the team often requests changes [14]. Change requests address not only new or changed requirements but also bug fixes and defects. Change requests are reviewed to determine the impact the change will have on related artifacts (source code, models, diagrams) as well as budget and schedule [1][15].

According to the CHAOS Report, the frequent change in requirements is one of the main challenges of projects, only behind incomplete requirements and the need for more cooperation between the team and the client [2]. In addition to being a significant difficulty for the team, how changes are managed can impact the success or failure of a project, leading to business losses owing to lost time and the need for rework. Iriate and Bayona [16] conducted a systematic review of the literature, where they collected 263 success factors, and the second most cited factor was change management.

Researchers estimated that the impact of changing requirements on industrial projects is between 40% and 90% of the total project cost [14]. Therefore, defining a change management process can minimize the negative impacts of a change over the life cycle of the software project [1][15].

The RCM process begins when a stakeholder completes and submits a change request that describes the required change to the system. After submitting a change request, the team evaluates its validity. This verification is needed because not all change requests require action [1]. For example, a reported bug may already have been fixed, or new features are already planned to be incorporated into the software. In these situations, the change request is closed, and the form is updated with the reason for closing [15].

For valid change requests, the next step is to assess the impact of the change. Generally, this task is the responsibility of the development team. The team must identify all affected components, the risks involved, and the estimated cost and time of making the change to evaluate the impact of the change [1][15].

After this analysis, the Change Control Board (CCB) decides whether, from a business perspective, the change should be made. This group is responsible for reviewing and approving change requests before they are implemented [1][15].

Andrew et al. [3] analyzed 43 articles related to challenges in RCM, with 32 on general problems and 11 specifics to Global Software Development (GSD). Then, they identified the main issues commonly faced by professionals. Finally, for each one, the authors calculated its frequency of occurrence in the selected papers. Given that this study does not focus on GSD, we only consider the 10 general challenges in RCM and their frequency of occurrence.

Andrew et al. [3] concluded that impact analysis is the most cited challenge (67%) and requires a thorough analysis of the new state of the system, consistency with existing business goals, and impact on other operational constraints. Cost and time estimates are critical aspects of project management mentioned in 25% and 12% of the papers, respectively. Another crucial challenge in the RCM process is the management of artifact documentation, which is mentioned in 25% of the documents. Requirement traceability and dependency are cited by 22% and 16% of the works, respectively. Change conflicts with existing requirements are noted in 12% of primary studies. The authors mentioned other challenges, such as change prioritization (6%), user involvement (6%), and system destabilizing (3%) [3].

2.2 Case-Based Reasoning

CBR is a paradigm in the AI area for reasoning and learning from past experiences. A reasoner retrieves and reuses similar past experiences to solve a new problem. This way, they can interpret it and discover ways of solving it. This technique is usually associated with how the human brain works when information stored in our mind is retrieved to create an answer to a problem [17].

In CBR, an experience is described as a case and stored in a base that contains many others. A case necessarily has two pieces of information about experience that are important for reasoning: the description of a problem and the solution adopted for its solution. A solution is not always successful, but it should be stored [18].

2.3 Knowledge Discovery in Databases

Knowledge Discovery in Databases (KDD) is a process of extracting information from databases. This process covers the essential steps for retrieving relevant data in mining, from preprocessing data to analyzing and interpreting the results. The initial preprocessing step consists of preparing the data for mining. There are usually null or inconsistent values that can generate incorrect results and need adjusting by the programmer. Therefore, only the relevant attributes are selected at this stage. After preprocessing, mining methods can be effectively applied to the data depending on the developer's wants. One of the most common techniques is ARM [11].

ARM is the procedure of finding frequent relationships between items in large datasets. These relations are expressions of form $X \rightarrow Y$, where X is a premise and Y is a consequence of this premise. Such associations consider two key measures: confidence and support of a rule. Confidence in data mining is the value that indicates how often a given premise and a consequence are true, that is, the percentage of times a given rule has occurred. This measure is highly associated with support, which indicates the frequency of occurrences of an item in a dataset. Confidence and support define an association rule [19][20]. Once the chosen algorithm has mined the data, it goes through a postprocessing step to clarify the visualization to the user. Finally, the last phase of the KDD process consists of the interpretation and analysis of what was obtained in the previous step to extract knowledge about the data.

Using different AI techniques to complement CBR is becoming more common. New hybrid methodologies seek to reduce disadvantages and increase the positive aspects of applying the techniques separately, thereby bringing greater precision and efficiency to an intelligent system [21].

The AI techniques are adequate to assist the team in change management activities, as they seek to replicate human intelligence [7]. Similarly, activities in the area of software engineering deal mostly with knowledge [1][22]. Among the AI techniques, we consider that the use of CBR is adequate, as it is commonly used in decision support systems [23], providing the user with ways to identify and resolve adversities in a "problem-solution" format similar to reasoning used by the human mind [9]. Even so, its use may not guarantee a complete understanding of the information. Therefore, the association rules are an ideal complement to bring greater precision and knowledge of the results obtained.

Based on several readings in the transaction database, the Apriori algorithm [24] is one of the best-known algorithms for mining by association rules, being able to work with a large number of attributes. The algorithm employs depth-first search and generates sets of candidate items. The entire database is crawled, and frequent itemsets are obtained from candidate itemsets. Infrequent patterns are eliminated. In Abid et al. [25], the authors used the Apriori algorithm to generate association rules that link source code and interface-level metrics with the quality of service.

Corners and Matties [26] argued that project documentation represents a valuable source of knowledge in project-based organizations. However, reusing knowledge encoded in design documents in future projects is not easy. The authors in the literature review pointed out that efficient tools to assist in the reuse of coding knowledge in project documents are lacking; therefore, this process is costly and time-consuming.

3. Related work

Change management processes depend on tools and techniques to support their activities and make their execution more efficient. Currently, numerous researchers proposed different solutions to ease the management of a change request.

Ali, Iqbal, and Hafeez [27] proposed a framework for RCM based on the CBR technique in the GSD context. GSD refers to software development by teams in various parts of the world. By applying the CBR-based framework in the cloud, the authors noted that the communication and coordination of the global teams during change management, which was previously challenging, became more effective. In addition, the services demanded were always available on a single platform without time and space restrictions, unlike when the team used tools on different sites.

Naz et al. [28] proposed a model that integrates RCM with the CBR technique. To validate the work, the researchers presented the framework to specialists and, later, asked them about factors such as customer satisfaction, history maintenance, and analysis of the impact on cost and time. The framework can help improve the factors listed by researchers.

Other authors proposed using techniques other than CBR for RCM. Tomyin and Pohthong [29] proposed a model based on object-oriented software engineering and the Unified Modeling Language (UML). Alsanad, Chikh, and Mirza [30] created a framework based on a multilevel ontology. Satyarthi and Pandey [31] developed a framework for managing change using a traceability matrix.

This work differs from the others as we focused on defining an approach to evaluate requirements change requests based on integrating ARM and CBR and developing a tool to support it in software projects. Given the proposed approach, the purpose is to assist users in analyzing risks that can occur during the implementation of an approved change, improving the change impact evaluation and effort estimations from the reuse of past cases.

Ali et al. [27] and Funk and Xiang [12] used CBR but did not explicitly support the same goals. In Ali et al. [27], the proposed framework aims to ease communication and control in teams with members scattered globally. In Naz et al. [28], the framework supports impact assessment but not risk management. Finally, Kitchenham et al. [29], Alsanad et al. [30], and Pandey and Satyarthi [31] propose techniques to assist the requirements changes using UML models, ontologies, and traceability matrices, respectively. But, these works do not consider information from past projects, which can improve the accuracy of the estimations.

Our approach proposes the use of CBR together with association rules. CBR allows reuse experiences to aid decision-making while evaluating a new change request. The association rules have two well-defined goals: assisting in retrieving cases and creating solutions by the user. Regarding case retrieval, the system uses the association rules to define the weights of relevant attributes when calculating the similarity between cases. Association rules can also reveal attributes that tend to cause problems when associated. If similar past cases are not retrieved, the user can propose a strategy to evaluate the new request by analyzing this set of mined rules.

4. Research method

Our approach, shown in Fig. 1, was developed using a combination of the CBR cycle proposed by Aamodt and Plaza [32] with knowledge discovery proposed by Tan et al. [11]. Prentzas and Hatzilygeroudis [21] mentioned that the disadvantages or limitations of specific intelligent methods could be overcome or mitigated by their combination with other methods. CBR provides easy knowledge acquisition using past cases, modularity, incremental learning, and some explanatory facilities. ARM provides general and compact domain knowledge through rules and rule-based explanation facilities [21].

We have chosen to use ARM before starting the CBR cycle to verify if the attributes of the current CR are associated with attributes that generated problems in past cases. This verification is needed because not all change requests may have associated risks, and you do not need to run the CBR cycle. The attributes will be used to define the weights used in the CBR similarity calculation if such attributes that caused problems are associated with the current CR. ARM does

not prepare the solution automatically, but it helps the stakeholder understand the attributes that can cause problems and build a solution.

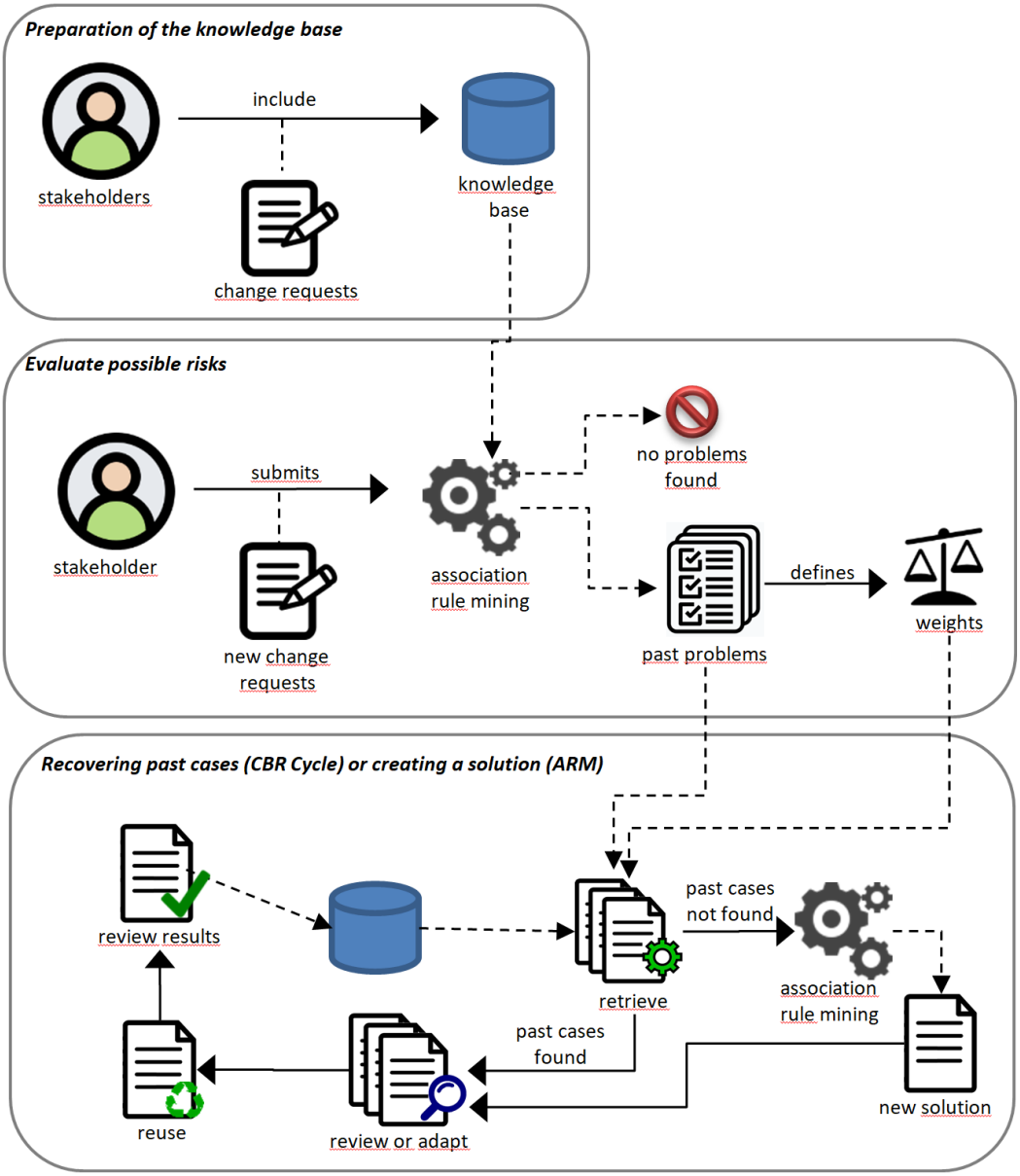


Fig. 1. Phases of the proposed approach

Therefore, our approach consists of the following phases:

1. Preparation of the knowledge base: change requests evaluated by the team are entered into the knowledge base. The data from these CR is prepared for use by the approach and is viewed as past cases;

2. Evaluating possible risks: possible risks of the current CR are evaluated by using the association of mining rules comparing the attributes of the CR with cases that generated problems in the past, extracting the data from the knowledge base using the Apriori algorithm;
3. Recovering past cases or creating a solution: if available in the knowledge base, then similar past cases are retrieved to be reused. The approach uses ARM to help the stakeholder create a solution if they are unavailable. After implementation, the stakeholder reviews the problems and solutions associated with the case and store them in the knowledge base.

4.1 Preparation of the knowledge base

The tool treats a change request as a case description created from a form containing the relevant attributes to be considered by the approach. A change request form has attributes and information for evaluating and implementing a change request. Explicit information and nonexplicit inferences should be considered. For example, project planning, the people involved, and the requirements affected impact the results of incorporating a change. Therefore, this data must also be available at the time of evaluation.

In ARM, this data is also necessary to identify the relationships that tend to generate consequences when they occur. The identified assumptions help the user assess whether a request has a low or high risk when implemented in the future based on the information entered in the form. Fig. 2 shows an example of a completed change request form and the class diagram used to represent the attributes and associations in the knowledge base.

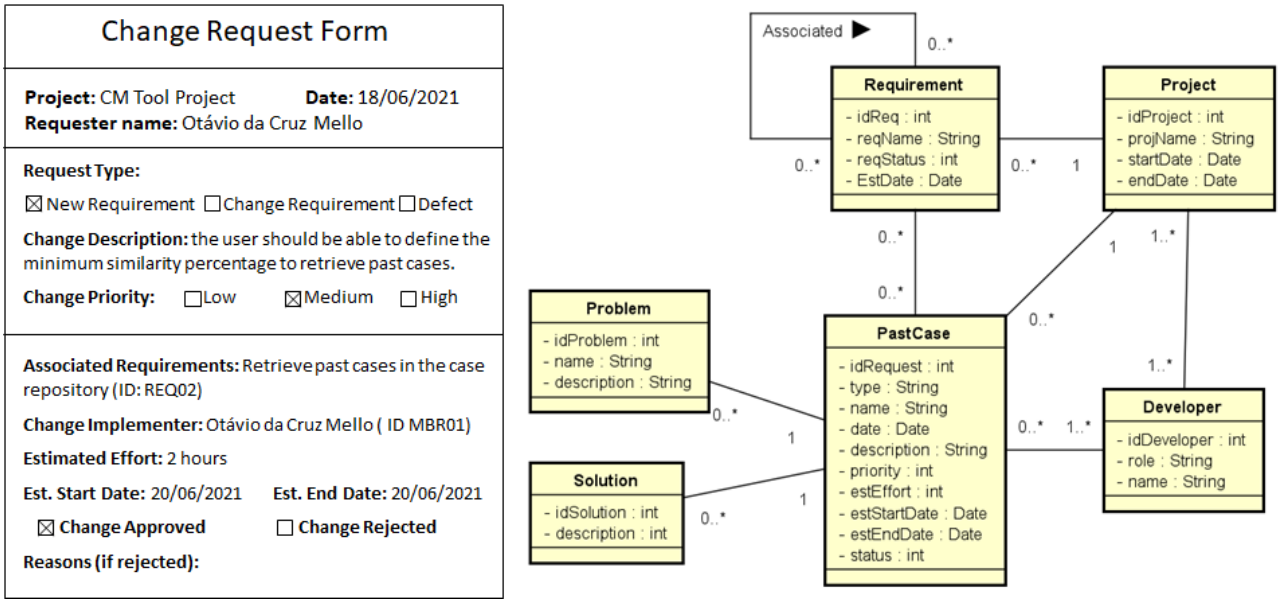


Fig. 2. A filled change request form example and Knowledge Base Class Diagram

4.2 Evaluating possible risks

The process begins when a stakeholder submits a new change request (Fig. 2). Then, the knowledge discovery process starts, retrieving associations of attributes of cases that have generated problems in the past by mining the data from the knowledge base using the Apriori algorithm. The retrieved premises are then compared with the data from the current CR to identify whether one or more rules apply to it. If at least one of the association attributes is not founded, then the approach considers that the rule is not related to the current CR. The full report of premises that could cause problems is

also returned to the stakeholder for analysis, and their confidence value is used to weigh attributes in the search for past cases. A rule is commonly composed of two essential parts: a premise - an attribute or an association of attributes - and your consequence. Furthermore, confidence indicates the number of times the premise-consequence statements are valid.

After submitting a new CR, the approach will mine the association rules in the knowledge base. The rules that result in problems and have a confidence level of at least 50% are filtered and displayed to the user. In addition, those in which the premises are not found in the new CR or are not relevant to the analysis are discarded.

If the approach does not return premises to the CR, then there will likely be no obstacles in implementing this change. When this occurs, there is no need to follow the CBR cycle for recovery from past cases. In this case, it will be evaluated after implementing the change.

The attributes have their weight in the similarity function defined by the respective confidence value of the filtered rules. If no rule with a premise matches any of the attributes of the request, then this will have its weight set to a default value. After the weights are defined, the flow advances to the recovery of the most similar past cases.

4.3 Recovering past cases or creating a solution

After that, the CBR cycle begins, consisting of five activities: case retrieval, review of suggested solutions, reuse of solutions, review of results, and case retention.

The approach initially predetermines the weight value of each attribute based on the association rules found on the knowledge base in the previous phase. Then, the weight measures the significance of each attribute and is used in the similarity functions to retrieve the most similar past cases.

We use the value equality similarity function to retrieve the *priority*, *request type*, *associated requirements*, and *implementer* attributes. This function determines if the attributes have the same value. For the *estimated effort*, we assign the function *distance between values*, which calculates the similarity of the values by distance. The *change description* attribute requires a specific similarity function that identifies the frequency of keywords in one text and compares it to the frequency of those same words in another. We have used the *keywords frequency* similarity function.

The "*k*" most similar cases are retrieved and sorted according to their similarity degree. The value of "*k*" can be changed, but five cases are returned by default. The stakeholder is able to decide which case best suits the current situation. The problems identified in past cases and their causes are made available in a report along with the solution that was taken at the time. A case has three parts: *attributes*, *problems*, and *solutions*. *Attributes* describe relevant items related to the change: priority, estimated effort, request type, associated requirements, implementers, and description. *Problems* describe the problems identified in past experiences, and *solutions* have been adopted to solve the problems.

However, the approach proposes a solution based on ARM if similar past cases are not found. With the rules explored by the Apriori algorithm, the user can also identify possible problems that may arise while implementing the change based on the association of attributes. Based on this analysis, a solution can be created to predict risks associated with CR based on its attributes, even without a similar past case. Once defined, the process will follow the CBR flow.

After selecting a similar past case or creating a solution using association rules, the stakeholder can follow the same procedures used to solve the past case or adapt them to the new context. After that, the team implements the change request as previously defined.

Following the implementation of the change, the stakeholder should review the findings to ensure that the solution has effectively reduced the risks identified in the change evaluation. If the risks are meaningless, then they should not be taken off the case. Moreover, previously unforeseen problems may occur. In this situation, the stakeholder should combine the problems and the solution adopted in the case. After the reviews, the case must be inserted into the knowledge base to complement it for future requests.

5. Case study

We chose to conduct a case study to evaluate our intelligent approach to assist teams in performing a risk analysis of new change requests in a project. Thus, we selected a project that aims to develop a virtual tactical simulator for training military personnel in military operations that use an Artillery Saturation Rocket System in SIS-ASTROS project.

A case study is carried out to examine a single entity or phenomenon in a given time frame. Over time, the researcher collects detailed information about an actual project and documents the results [33]. Case studies help to assess the benefits of processes and tools and provide a cost effective way to ensure that process changes predict desired outcomes [29].

Wohlin [33] posited that a case study is more suitable if the effect of a process change is widespread. However, the effect of change can only be assessed at a high level of abstraction because process change includes smaller and more detailed changes throughout the development process [29].

Based on the literature [29][33][34], we defined the following steps to carry out the case study:

1. Conception and design: we define the objectives and design the case study;

2. Preparation for data collection: we define procedures for data collection;

3. Collection: we execute the approach with the support of the tool and collect the data;

4. Analysis and reporting of collected data: we analyze the collected data and report the results.

5.1 Conception and design

First, we defined the project in which the case study would be carried out, the objectives, the research hypotheses, and how to evaluate the hypotheses and the results obtained.

We carried out a case study on a project that started in May 2021, which has an estimated duration of 4 years and a development team of 30 people. This case study is a research project between the Federal University of Santa Maria (UFSM) and the Brazilian Army that aims to develop a tactical virtual simulator for training soldiers in operations related to using an Artillery Saturation Rocket System. Two members of the project's CCB took part in the evaluation. This project continues another project that started in December 2015 and lasted 5 years. The cases stored in the case base come from this first project.

The main objective is to determine whether the proposed approach supported by the tool can minimize the main difficulties and problems faced by change management professionals, mitigate their occurrence, or help to solve them. For this validation, we considered the challenges proposed by Andrew et al. [3] (described in Section 2.1) with a percentage greater than 10%. Therefore, our goal is to verify if our proposal helps the teams meet the following challenges: impact analysis, cost estimation, artifact document management, requirements traceability, requirements dependency, change conflicts with existing requirements, and time estimation.

To reach this goal, we evaluate whether the proposed approach, supported by the tool, provides the necessary support to solve the main challenges related to configuration management, based on the hypotheses listed below.

H1. Our proposal can help the project team to carry out a change impact analysis more effectively than when performed on an *ad-hoc* basis.

H2. Based on the returned cases, the team can estimate realistic costs and time to make the change.

H3. By associating requirements with changes, the approach allows considering requirements traceability, requirements dependency, and change conflicts with existing requirements when analyzing a change by the team.

H4. The approach allows managing artifact documents while maintaining a history of requirements changes over time.

5.2 Preparation for data collection

We have implemented a change management tool using the Java object-oriented programming language, the Weka API and jColibri libraries, and the Java DB relational database management system (Derby) to support the defined approach.

We initially enter the already evaluated and finalized change requests into the knowledge base. Then, we register the risks and solutions described in CR with the support of professionals involved in the change management activities of the project. In this activity, we inserted 25 cases of the SIS-ASTROS project in the case base. Fig. 2 depicts the class diagram representing the classes implemented in the tool.

In addition, we have separated 10 change requests to be evaluated during the data collection phase to verify that the tool and its base can return to the user past cases with coherent data about previous problems and solutions.

5.3 Collection

Two project developers used the tool to enter 10 CR during tool validation. In the case study description, we called them "users" as they played the system user role. For the evaluation of each CR, the user followed the steps described below (Fig. 3).

Initially, the user enters the data available in the change request form, shown in Fig. 2, into the evaluation tool and complements this data by informing the requirements and project members of the request (Fig. 3.1). This data is needed to identify past cases. From this data, the tool performs ARM to identify assumptions that created problems in the past. Next, the user checks those present in the CR (Fig. 3.2). The tool automatically assigns the weight for the attributes to be considered based on the validation rules (Fig. 3.3). The user may modify these values considering his experience and intuition.

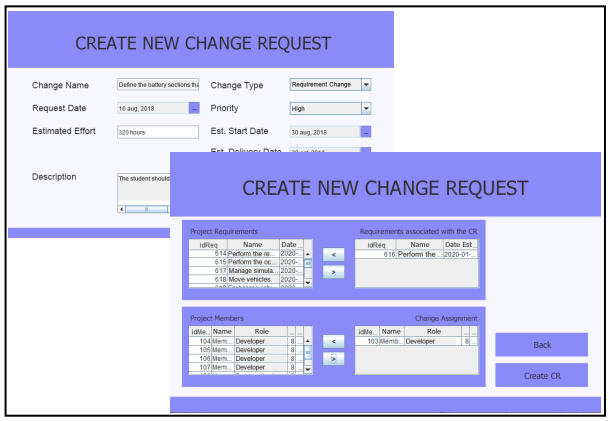
The tool returns to the user the "*k*" cases most similar to the current case and the percentages of similarity assigned from the weights (Fig. 3.4). The user analyzes and selects the case he wants to reuse. When the case is selected, the tool informs the associated problems and solutions. It temporarily associates these problems with the new request (Fig. 3.5). Finally, after implementing the CR, the user records information about the change implementation in the tool, reporting the problems and solutions (Fig. 3.6).

5.4 Analysis and reporting of collected data

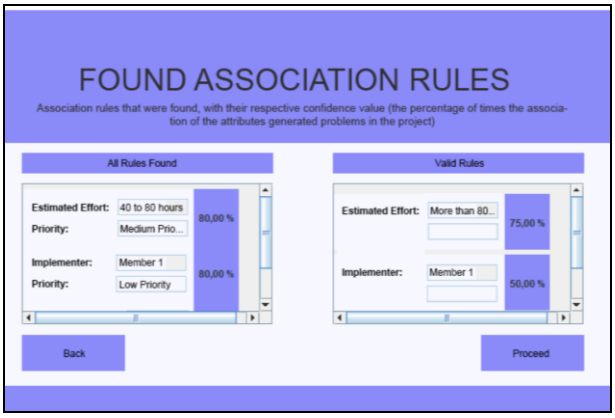
Runeson and Höst [34] noted that data collection through interviews is necessary for case studies. In this technique, the researcher asks a series of questions about the areas of interest in the case study. The dialog between the researcher and the interviewees is guided by a set of questions elaborated on the subjects of interest of the study [35]. In this case study, we used a semistructured interview. Initially, we prepared questions to verify whether the hypotheses were valid for the case. However, new questions emerged during the discussion and were included in the script.

During the validation, the project team developers considered the relevance of the results returned by the tool. Notably, the team had already implemented the CR used during validation. Therefore, the analysis can verify if the data returned by the tool were consistent with the actual implementation of the changes. The validation considered the relevance of the effort estimates, priority, problems, and the solutions documented in the past case to evaluate the current change request. Also notably, the problems encountered in past cases are significant sources of risk.

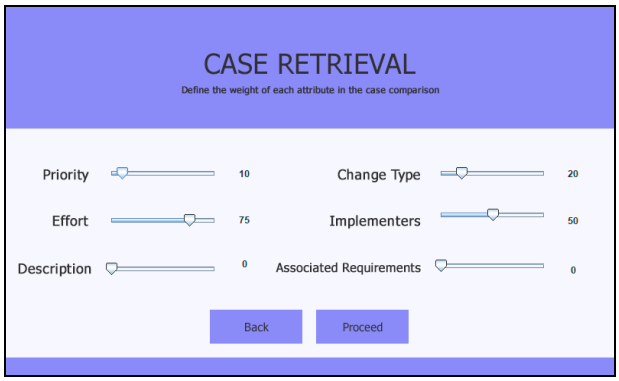
After entering all change requests, we interviewed the project team developers to verify if the hypotheses were valid for the case in question. Validation demonstrated that the approach helps in the evaluation of change requests. Many problems identified by retrieving past cases were also found during its implementation in the SIS-ASTROS project. We describe below some examples to demonstrate how the analysis was performed and how the results were obtained.



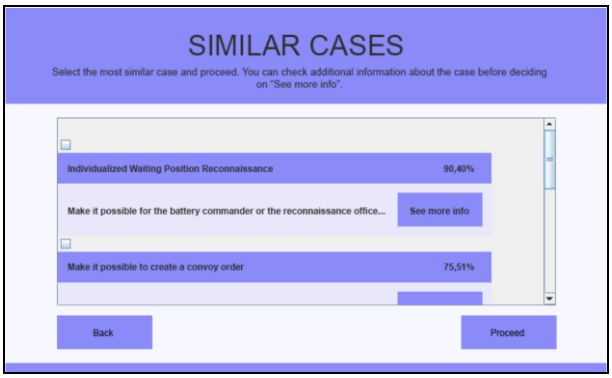
(3.1)



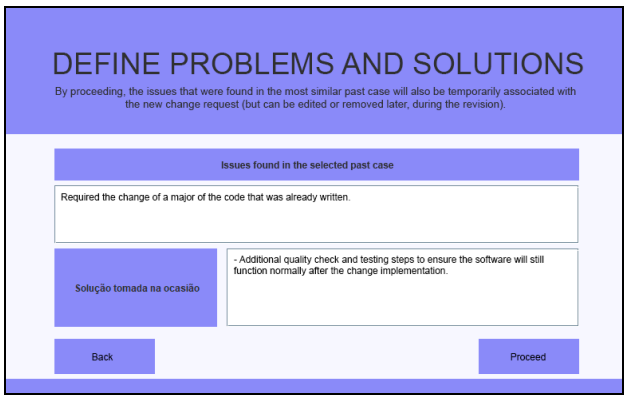
(3.2)



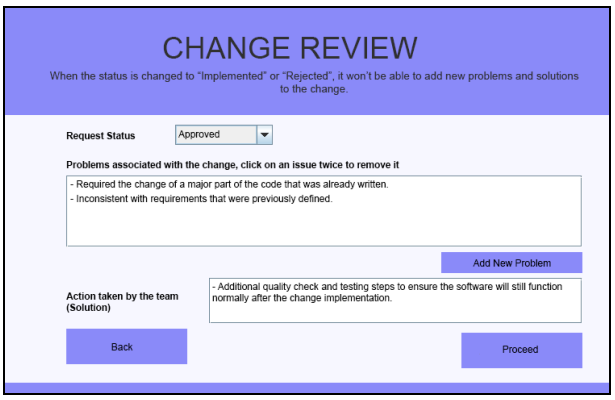
(3.3)



(3.4)



(3.5)



(3.6)

Fig. 3. Steps for evaluating a CR using the system

For the first change request used in the validation, the system retrieved the five most similar past cases, one where the similarity percentage exceeds 90% and others where it varies between 60% and 80%. The developers individually analyzed each case returned and selected the one that could be most useful for evaluating the current change request. In this example, the developers chose the most similar case because it has the same estimated effort as the new request, the same implementers, and the same change type (new requirement). In addition, the retrieved case description documented that much of the code had to be modified in implementing the change request.

Therefore, by comparing the information returned by the past case with that of the current change evaluation form, we saw that the CCB had identified this risk, which occurred in the change implementation. Therefore, the tool retrieved coherent and needed information for assessing the risks associated with the change. From another aspect, another problem was identified and described in the CR; it had inconsistencies with another previously defined requirement, requiring the team to acquire more information about the change from the customer. As this problem was not identified in the past case recovery, it had to be manually added. Situations such as these tend to occur in repositories that are still growing. As more past cases are stored, the information retrieved becomes more accurate and correct.

Another example was a change request to fix a defect related to viewing a simulator map. For this CR, similar past cases that, according to the team's developers, provided important information for evaluating the CR could also be retrieved. The most similar case reported difficulty in repetition as a problem. The tool returned two other less similar past cases that were not selected. The retry difficulty issue was associated with the new request. Upon further analysis, we confirmed that this problem occurred during the change implementation.

Past cases also helped to reject a change request. For example, in a change request, the customer requested a change in the teleport algorithm to allow teleporting vehicles to specific areas with dense vegetation. The past case retrieved reported the change's high complexity and that the algorithms had already been adjusted to allow for maximum vegetation density. Changes in these parameters could generate overlapping objects in the virtual world. Therefore, this information helps the team to reject a requested change without putting effort into its investigation.

Testing resulted in four possible scenarios: relevant cases were returned, no similar cases were returned, the problems returned did not satisfy the new request, or the most similar past case had no associated problems. Even with the small base, in only 10% of the cases tested, the system returned past cases that, according to the developers, were not relevant to the evaluation of the current request.

At the end of the validation, we interviewed the project developers to assess their perception of the validity of each hypothesis defined in the case study planning. Their considerations are described below.

Regarding H1, the developers agreed that the approach helps the project team to carry out a change impact analysis. They noted the importance of problems and solutions associated with the past case for identifying implementation risks and proposing responses. Developers could see that most of the issues reported in past cases occurred in implementing the change in the project.

In commenting on the impact analysis, the developers highlighted that the associations with the requirements help to identify the risks related to the requirement's complexity and source code quality. For example, complexity can lead to the need for additional time to understand current requirements, and poor source code quality can lead to refactoring (source code rewriting). When the project team evaluated the CRs, they did not identify the risk of requiring additional work owing to rewriting the source code. This association of the CR with the requirements allows for identifying this risk.

Regarding H2. The project developers agreed that the information from past cases returned by the tool helps estimate the time and cost of a change. According to the software engineering literature, using historical data is one of the most used techniques to perform estimation activities in software projects [36]. Past case retrieval considers the project member who made the change, the associated requirements, and the size of the change. These attributes are needed to improve the accuracy of estimates.

Regarding H3. The developers agreed that by associating requirements with change, the tool helps assess the impact of the change on other system requirements. In the tests, we can verify whether it was necessary to change other requirements not initially specified in the change request form if there are risks of implementing a change associated with a given requirement. Through past cases, developers could assess risks that requirements not described in the form but associated with the change may bring when implementing it. Traceability is also an important aspect to consider in the impact assessment, as noted in H1.

Regarding H4. The developers agreed that managing artifact documents is not the tool's purpose but maintaining a history of requirements changes over time. Most of the time, change request forms are not kept during the project and are discarded once the request is implemented. However, our approach stores these change requests in your case repository so that you can reuse them in the future in the CBR cycle. In this way, these documents will never be lost and can be viewed at any time by anyone, allowing the retrieval of the history of requirements changes.

Overall, the proposed approach based on CBR and ARM has the potential to solve seven of the 10 most frequent challenges encountered in managing change requirements.

5.5 Threats to validity

We performed a single case study for validation. We had to work with a limited set of change requests. Most of them, more specifically 25, were used to build an initial case base, and 10 additional cases were used during validation.

This number is relatively low, but only 10% of the time, the retrieved cases were inadequate to provide a solution for the new request. Therefore, this base needs to be supplemented with more past cases. In addition, the evaluation of the results obtained was carried out by two project developers and was a qualitative evaluation. In future works, we intend to improve our tool by adding more case studies on different projects.

One advantage of case studies is that they are easy to design. However, the disadvantages are that the results are difficult to generalize; it is possible to show the effects in typical situations evaluated.

6. Concluding remarks

As mentioned earlier, this study mainly aims to propose a change management approach that facilitates the risk analysis of a change request, providing information about past problems encountered in similar change requests and the solutions adopted. In this way, the approach helps teams to make better decisions and strategies when implementing a change.

Throughout this work, we present the selected techniques' basic concepts, the proposed approach, and the tool developed to support it. Finally, we validated the proposal to assess whether the approach and tool were adequate to solve challenges in change management.

The proposed approach was evaluated, considering the difficulties found in the literature. The approach had the potential to solve seven out of the 10 most frequent challenges of change management based on a deep analysis of each problem individually. From another aspect, the tool was evaluated by analyzing the results generated by a case base created using past change requests from the SIS-ASTROS project. This case base was built with the help of an expert from the project involved in change management activities.

The initial results showed that four different scenarios could occur: relevant cases were returned, no similar cases were returned, the problems returned did not satisfy the new request, or the most similar past case had no associated problems. However, only 10% of the cases failed to generate relevant results for the user.

After the tests, we interviewed the specialists to respond to hypotheses related to the change impact analysis, cost and time estimation, traceability, and dependency on previously defined requirements and documentation. They noted that the tool helped them to conduct change management activities and pointed out some considerations for improvement.

Based on the tests and the interview, the proposed approach can help to reduce the main problems in the area, and the support tool can provide logical results to identify possible implementation risks of the new change request.

We conclude from the results of this study that, although improvements are still needed, the usage of the proposed change management approach based on ARM and CBR can be beneficial. The risk analysis activity can be simplified for the user and consequently helps make decision-making easier.

Some limitations and difficulties were encountered during this research, particularly in its final stages. Acquiring data to create a case base for validation took considerable effort and time. As the project did not have many suitable change requests documented, we had to work with a relatively low number of change requests (25). Another issue we faced was the restricted number of professionals directly involved in change management activities in the project and their limited availability, causing the need to coordinate the testing and interview schedules.

Future works can improve this approach by proposing a solution to other issues in the change management area. In addition, future works can make changes to the support tool to bring more information, reports, and graphs to the user about the performance of the AI algorithms and improve the visual aspects of data representation to make their interpretation clearer when used.

This tool is also intended to be applied in other projects, and its performance can be analyzed over a longer time period with a more extensive case base. Thus, the results obtained can be evaluated in a practical environment, and therefore, implementation issues that may arise can be improved.

Acknowledgments

We thank the Brazilian Army for the financial support through the SIS-ASTROS (813782/2014) and SIS-ASTROS GMF (898347/2020) projects.

References

[1] Ian Sommerville, *Software Engineering*, 9th ed. São Paulo, Brasil: Pearson Prentice Hall, 2011.

[2] Standish Group International. (2014, 06). *The Standish Group Report CHAOS* [Online]. Available: https://www.projectsmart.co.uk/white-papers/chaos-report.pdf.

[3] S. Anwer, L. Wen, and Z. Wang, "A Systematic Approach for Identifying Requirement Change Management Challenges: Preliminary Results," in *EASE '19: Proceedings of the Evaluation and Assessment on Software Engineering. Association for Computing Machinery*, Copenhagen, Denmark, pp. 230-235, 2019.

[4] M. Harman, "The role of artificial intelligence in software engineering," in *RAISE '12: Proceedings of the First International Workshop on Realizing AI Synergies in Software Engineering*, Zurich, Switzerland, 2012, pp. 1-6.

[5] Information Resources Management Association, *Software Design and Development - Concepts, Methodologies, Tools, and Applications*, 1st ed. Hershey, PA, United States: IGI Global, 2013.

[6] A. P. Sage and W. B. Rouse, *Handbook of Systems Engineering and Management*, 2nd ed. Hoboken, NJ, United States: Wiley-Interscience, 2014.

[7] S. J. Russel and P. Norvig, *Artificial Intelligence - A Modern Approach*, 4th ed. Upper Saddle River, NJ, United States: Pearson Prentice Hall, 2020.

[8] A. Alsahli, H. Khan, and S. Alyahya, "Toward an Agile Approach to Managing the Effect of Requirements on Software Architecture during Global Software Development," *Scientific Programming*, vol. 2016, pp. 1–16, 2016.

[9] C.-H. Liu and H.-C. Chen, "A novel CBR system for numeric prediction," *Information Sciences*, vol. 185, no. 1, pp. 178-190, 2012.

[10] J. Ramos-González, D. López-Sánchez, J. A. Castellanos-Garzón, and J. F. De Paz, "A CBR framework with gradient boosting based feature selection for lung cancer subtype classification," *Computers in Biology and Medicine*, vol. 86, pp. 98-106, 2017.

[11] P.-N. Tan, M. Steinbach, A. Karpatne, and V. Kumar, *Introduction to Data Mining*, 2nd ed. Harlow, United Kingdom: Pearson, 2018.

[12] P. Funk and N. Xiong, "Case-based reasoning and knowledge discovery in medical applications with time series," *Computational Intelligence*, vol. 22, no. 3-4, pp. 238-253, 2006.

[13] A. V. Mote and M. Ingle, "Enriching Retrieval Process for Case Based Reasoning by using Vertical Association Knowledge with Correlation," *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 2, no. 12, pp. 4114–4117, 2014.

[14] C.e Ingram, "Using requirements and design information to predict volatility in software development," PhD. dissertation, School of Computing Science, Newcastle University, Newcastle, United Kingdom, 2011.

[14] I. Sommerville and G. Kotonya, *Requirements Engineering - Processes and Techniques*, 1st ed. Hoboken, NJ, United States: Wiley, 1998.

[15] R. S. Pressman and B. R. Maxim, *Software Engineering: A Practitioner's Approach,* 9th ed. New York, United States: McGraw-Hill Global Education Holdings, 2020.

[16] C. Iriarte and S. Bayona, "It projects success factors: A literature review," *International Journal of Information Systems and Project Management*, vol. 8, no. 2, pp. 49–78, 2020.

[17] J. Kolodner, *Case-Based Reasoning*, 1st ed. San Mateo, CA, United States: Morgan Kaufmann, 1993.

[18] M. M. Richter and R. O. Weber, *Case-Based Reasoning - A Textbook*, 1st ed. Berlin, Germany: Springer, 2013.

[19] C. C. Aggarwal, *Data Mining: The Textbook*,1st ed. Berlin, Germany: Springer, 2015.

[20] A.s Gregoriades and A. Christodoulides, "Traffic Accidents Analysis using Self-Organizing Maps and Association Rules for Improved Tourist Safety," in *19th International Conference on Enterprise Information Systems*, Porto, Portugal, 2017, pp. 452-459.

[21] J. Prentzas and I. Hatzilygeroudis, "Case-based Reasoning Integrations: Approaches and Applications," in *Case-Based Reasoning: Processes, Suitability and Applications*, Antonia M. Leeland, Eds., 1st ed. New York, United States: NOVA Science Publishers, 2011, pp. 1-28.

[22] E. F. N. Raza, "Artificial Intelligence Techniques in Software Engineering," *Lecture Notes in Engineering and Computer Science*, vol. 2174, 2009.

[23] P.-C. Lee, T.-P. Lo, M.-Y. Tian, and D. Long, "An Efficient Design Support System based on Automatic Rule Checking and Case based Reasoning," *KSCE Journal of Civil Engineering*, vol. 23, 2019, pp. 1952-1962.

[24] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *Proc. 20th Int. Conf. Very Large Data Bases*, San Francisco, United States, 1994, pp. 487-499.

[25] C. Abid, M. Kessentini, and H. Wang, "Early prediction of quality of service using interface-level metrics, code-level metrics, and antipatterns," *Information and Software Technology*, vol. 126, pp. 106313, 2020.

[26] A. Coners and B. Matthies, "Perspectives on Reusing Codified Project Knowledge: A Structured Literature Review," *International Journal of Information Systems and Project Management*, vol. 6, no. 2., pp. 25–43, 2018.

[27] S. Ali, N. Iqbal, and Y. Hafeez, "Towards Requirement Change Management for Global Software Development using Case Based Reasoning," *Mehran University Research Journal of Engineering and Technology*, vol. 37, no. 3, pp. 639-652, 2018.

[28] H. Naz, Y. Motla, S. Asghar, M. Abbas, and A. Khatoon, "Effective usage of AI technique for requirement change management practices," in *5th International Conference on Computer Science and Information Technology*, Amman, Jordan, 2013, pp. 121–125.

[29] B. Kitchenham, L. Pickard and S. L. Pfleeger, "Case Studies for Method and Tool Evaluation," *IEEE Software*, vol. 12, no. 4, pp. 52–62, 1995.

[30] A. A. Alsanad, A. Chikh, and A. Mirza, "Multilevel Ontology Framework for Improving Requirements Change Management in Global Software Development," *IEEE Access*, vol. 7, pp. 71804-71812, 2019.

[31] D. Pandey and S. Satyarthi, "Framework for Requirement Management using Requirement Traceability," *International Journal of Advanced Research in Computer Science*, vol. 8, pp. 904-908, 2017.

[32] A. Aamodt and E. Plaza, "Case-Based Reasoning: Foundational Issues, Methodological Variations, and SystemApproaches,", *AI Communications*, vol. 7, no. 1, pp. 39-59, 1994.

[33] C. Wohlin, M. Höst, and K. Henningsson, "Empirical Research Methods in Software Engineering," in *Empirical Methods and Studies in Software Engineering,* Conradi R., Wang A.I., Eds. Berlin, German: Springer, 2013, pp. 7-23.

[34] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, vol. 14, no. 2, pp. 131-164, 2009.

[35] P. Dilley, "Conducting Successful Interviews: Tips for Intrepid Research," *Theory Into Practice*, vol. 39, no.3, pp. 131-137, 2000.

[36] S. Charilaos, "Simulating Component-Based Agile Software Cost," M.S. thesis. School of Computing Scinece, Aristotle University of Thessaloniki, Thessaloniki, Greece, 2016.

Biographical notes

Otávio da Cruz Mello
Otávio da Cruz Mello is a Computer Science Master's student in the Federal University of Santa Maria and a Project Manager in PROCERGS, a data processing company in Brazil. Otávio holds a B.Sc. in Information Systems since 2021. His research focuses on Artificial Intelligence applied to several fields of Software Engineering, mainly Requirements Engineering.

Lisandra Manzoni Fontoura
Lisandra Manzoni Fontoura is an associate professor at the Department of Computer Applied, Federal University of Santa Maria. She has a Ph.D. and an M.Sc. in Computer Science from the Federal University of Rio Grande do Sul. Her research areas are Software Process, Project Management, and Game Development. She participates in simulator development projects for military training in partnership with the Brazilian Army.