A Method to Recommend Artifacts to New Tasks in Software Projects

Edson M. Lucas[a,b], Toacy C. Oliveira[a], Paulo S.C. Alencar[c]
[a]*PESC/COPPE, Federal University of Rio de Janeiro, Rio de Janeiro, Brazil*
[b]*Polytechnic Institute (IPRJ/UERJ), Nova Friburgo, Brazil*
[c]*David Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada*

edmlucas@cos.ufrj.br, toacy@cos.ufrj.br, palencar@uwaterloo.ca

Abstract— **The software development workflow typically involves developers executing tasks and manipulating artifacts. When developers receive a new task they typically envision a task context with the artifacts they intend to manipulate based on their past experiences. Given software projects may last several months, accumulating a vast amount of tasks, artifacts and developers, envisioning this initial task context may be difficult and error-prone. Developers have to walk-through months of past experiences or examine the experience of other developers, select similar tasks and then define the initial context. This paper introduces a method that helps developers defining the initial task context by combining interaction information over artifacts with text information of tasks. First, the Method uses the Clustering technique to organize project tasks into similar groups by interaction in artifacts. Then, the Method uses the Natural Language Processing technique to associate a new task with groups of similar tasks by interaction. The evaluation shows that the clustering of similar tasks by interaction produces similar tasks assigned with artifacts that will be edited by new tasks. The association of new tasks with similar groups by interaction indicates correlation between textual similarity and interaction similarity.**

Keywords-component; interaction; task context; recommendation

I. INTRODUCTION

Software development projects last for months or years, where developers interact with each other and manipulate many artifacts. A typical task context in software development is formed by a set of artifacts that the developer uses to perform a task [1], [2]. Another widely explored context variable is the identification of task experts [3], as well as by experts in similar tasks to the new task [4].

When the developer receives a task to correct an error or add new functionality to the software, the developer usually starts by searching for artifacts that he needs to change to accomplish the task. This search is based on the developer's experience and the human capacity to remember of the artifacts already used in previous similar tasks [5]. In addition, the search is time-consuming, even when using the traditional navigation and textual search tools available in development environments, e.g., Eclipse Project Explorer.

In this scenario, recommending an initial task context can help developers in performing new tasks. The recommendation tools in Software Engineering aim to reduce the uncertainties of the future through the analysis of project history, and in a more specific way, according to Robillard et al. [6], these tools are software applications that provides information items estimated as valuable to a software engineering task in a given context.

Aiming at inferring new task context, some works focus on the recommendation of artifacts, i.e., source code, files, classes, packages [1], [7]–[11], others in the recommendation of artifact experts [12], [13]. Proposals [14], [15] and [16] recommend experts to new tasks. Malheiros et al. [17] and Ashok et al. [18] recommend similar tasks to new task, while Wang et al. [4] and Ashok et al. [18] also recommend experts on similar tasks to new task.

This paper presents part of the Tacin method (**Ta**sk **C**ontext based on **In**teractions) to recommend a new task context in software development projects. First, we defined that two tasks are similar by interaction if the developers edited at least one common artifact while performing the tasks. The weight of similarity is equal to the number of artifacts in common. After, the Method uses the Clustering technique to organize project tasks into similar groups by interaction in artifacts [19]. Then, the Method uses the Natural Language Processing technique to associate a new task with groups of similar tasks by interaction [20]. In this article, the edit interaction values are captured by Mylyn, an interaction model [28].

In the evaluation, the clustering of similar tasks by interaction indicated the production of task groups that had only 6% of the artifacts available in the project, but containing 77% of the artifacts that will be edited by new tasks. The association of new tasks with similar groups by interaction reduced by 90% the available artifacts and produced a recall of 52%. Therefore, the evaluation indicates success in Clustering and need for improvement in Natural Language Processing technique, i.e., improve the recall to a percentage closer to 77%.

This paper is organized as follows. Section II defines Clustering and Natural Language Processing. Section III presents the Method to recommend artifacts to new tasks in software projects. Section IV assesses the Method based on

DOI reference number: 10.18293/SEKE2019-100

average hits on recommending artifacts to new tasks. Section VI reviews related work and Section VI concludes our paper with a brief description of future work.

II. Background

Developers interact with artifacts and collaborate among them to perfom tasks in software development projects. The task objective is generally expressed by text. For example, the Mylyn Docs project uses a *short description* in natural language (English) to express an error. So, developer uses this textual information to search artifacts from the project's artifact database to correct it.

Table 1. Short description of task 245759 from the Mylyn Docs project.

	TaskId 245759[a]
Short Description	cannot run the HtmlViewer or MarkupViewer in a stand-alone GUI

a. Available in https://bugs.eclipse.org/bugs/show_bug.cgi?id=245759.

The project task history is large since the projects last months, but for each performed task is possible to have the artifacts that were edited and by whom. Our method uses Clustering technique to organize the software project history database in clusters using edit interaction information. After, our method uses Natural Language Processing technique to identify the cluster that best fits the new task using textual information [20]. In this section we briefly introduce these two techniques.

Clustering is a computational technique for organizing data objects (elements) into groups (clusters) in order to provide an organization to support the human being in understanding information. Most similar elements tend to stay in the same group, while less similar or non-similar elements tend to stay in different groups [21], [22]. Similar groups do not have an identification according to the content of the groups, so this technique is also known as unsupervised learning.

The Natural Language Processing (NLP) is formed by a set of computational techniques motivated by theory for the automatic analysis and representation of human language [20]. The techniques and models designed for one language are not easily generalized to other languages in most cases [23]. According to Cambria and White [20], NLP research began with the word analysis paradigm, evolved to the analysis of concepts, and it is expected that models can understand narratives in the future.

III. Method to Recommend Artifacts to New Tasks

The Tacin method recommends a new task context defined as: 1- artifacts that will probably be edited by developers to perform a new task; 2- performed tasks similar to the new task; 3- experts in the new task. This paper presents the part of Tacin to recommends artifacts to new tasks in software projects.

In software projects, developers create or change artifacts to perform tasks. So, Tacin defines that the similarity weight between two tasks is equal to the number of artifacts edited in common between them. First, Tacin organizes task history into groups of similar tasks by edit interaction. Figure 1 shows the task representation of the Mylyn Docs project. Tasks are represented by green triangles; artifacts by blue squares; interactions by black lines linking tasks to their artifacts. A black line can represent one or more edit interactions on artifact.

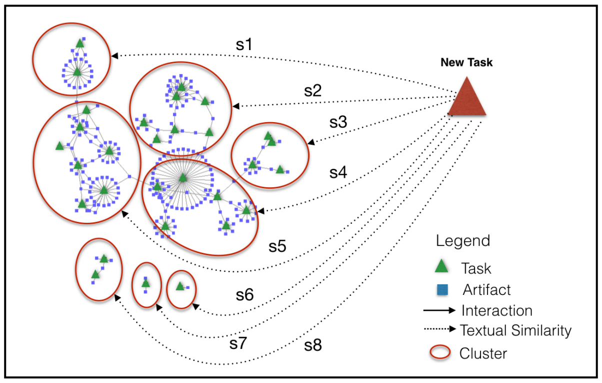

Figure 1. Illustration of the method to recommend artifacts to compose the context of a new task in software development project.

The automatic determination of similar groups from a dataset is extensively studied in the literature [21]. Tacin uses LNS_SMC (Large Neighborhood Search - Software Module Clustering) heuristic based on the large neighborhood search metaheuristic. The LNS_SMC algorithm was chosen because presented a good efficiency using the Modularization Quality (MQ) measure applied to the clustering problem of software modules [20].

Figure 1 illustrates the association of a new task with the project task history organized in clusters. Each cluster can be seen as a single task where its *short description* is defined as the concatenation of the short descriptions of all tasks belonging to the cluster. Thus, the textual similarity between the new task and the clusters can be calculated.

Figure 1 also illustrates the recommendation of artifacts with Tacin. The first three groups (clusters) that present the greatest textual similarity to the new task are selected for the recommendation. Tacin recommends artifacts that belong to the group(s) considering three options: 1 - only the group most similar textually to the new task; 2 - the two groups most similar textually to the new task; 3 - and the three groups most similar textually to the new task. Tacin chooses a option accoording with the effectiveness of past recommendations. For this, the harmonic average between Reduction and Recall (Rc-measure) for three options is calculated for each new task, equations defined in (1-3). The option that has the highest Rc-measure receives 1 point. Then Tacin can recommend artifacts to developers using the highest scoring option. If there is a tie, Tacin recommends the option of a smaller number of selected groups.

$$\text{Reduction} = 1 - \frac{|\text{Recommended Artifacts}|}{|\text{Project Artifacts}|} \qquad (1)$$

$$\text{Recall} = \frac{|\text{Recommended Artifacts} \cap \text{Relevant Artifacts}|}{|\text{Relevant Artifacts}|} \qquad (2)$$

$$\text{Rc-measure} = 2 \times \frac{(\text{Recall} \times \text{Reduction})}{(\text{Recall} + \text{Reduction})} \qquad (3)$$

IV. Assessment

The planning of the study includes objective, case, research questions and method according to the guide to conduct and report case study in Software Engineering [24]. The objective is described in the format indicated by GQM [25]: *Analyze* the Tacin method to recommend artifacts *for the purpose of* evaluating their effectiveness *from the perspective of the* developer *in the context of* a software development project. According to this objective following the RQ1 research question: Is the Tacin method effective in recommending artifacts at the beginning of a new software development task?

The effectiveness of Tacin depends on the effectiveness of task clustering by edit interaction and the correlation between textual similarity and similarity by edit interaction between tasks. Then two research sub-questions were defined: RQ1.A: Is task clustering by edit interaction effective for recommending artifacts? RQ1.B: Is there evidence on the existence of correlation between textual similarity and similarity by edit interaction among tasks?

Tacin reduces the number of artifacts available at the beginning of a new task, trying not to omit the artifacts that will be edited by developers. Equation 1 present Reduction measure. Equation 2 show Recall measure. So, Equation 3 combines these two measures in a harmonic way. Therefore, the Rc-measure metric was chosen to measure the efficacy of the Method.

The Mylyn Docs project was the case selected to evaluate the artifacts recommendation because it has the textual information of the tasks and developer interactions on the artifacts. The data collected from Mylyn Docs project was generated by the Mylyn, Git and Bugzilla tools. The Mylyn plugin logs developer's interactions about artifacts as interaction event with kind='edit'. The parameters for the query were Classification = Mylyn and Product = "Mylyn Docs" and Component = EPUB or Framework or HtmlText or Wikitext and Status = RESOLVE and Resolution = FIXED and Match ALL of the following separately → Attachment Description → contains the string → mylyn/context/zip. The query was executed in the site https://bugs.eclipse.org/bugs/query.cgi at April 05, 2017 and returned 334 tasks with 49906 edit interactions performed by 6 developers over 1538 artifacts. We have identified many performed tasks in the Mylyn Docs project that do not have Mylyn logging. So, committing actions were collected to infer editing actions, usually, a committing action submits edited files to code repository. Commits extraction resulted in 918 commits performed by 33 developers over 5407 artifacts.

The study generated artifact recommendations to new task in 20 trials, simulating a new task on each first day of the month from 09/01/2008 to 04/01/2010 to evaluate a long time period. The artifacts that were associated with tasks up to the date of the trial and were also associated with the new task after their completion form the set of relevant artifacts of the new task. Tacin tries find these relevant artifacts using only edit interactions finished before the start of each new task. Each trial generates groups of similar tasks by edit interaction. For each trial, 30 trials using the LNS_SMC were performed and the cluster with the highest MQ was chosen to be used to make recommendation.

Tacin calculates all textual similarities between the new task and the calculated clusters. The *short description* field of the new task was used with concatenation of all the *short description* fields of the tasks in each cluster. RapidMiner Studio was used to calculate textual similarity. The text processing used the functions Replace Tokens, Transform Cases, Tokenize, Filter Stopwords, Filter Tokens (by Length) and Stem (Porter). Then the texts were represented in vectors using the occurrence of terms. Finally, the similarities between the vectors (texts) were calculated using the cosine similarity function.

The evaluation contemplates three options of recommendation according to textual similarity among new task and tasks clustered by edit interaction. The first option evaluates the recommendation of the artifacts to the group that presents greater similarity. The second recommends the task artifacts of the two groups that present the greatest similarities. Likewise, the third one recommends the task artifacts of the first 3 groups. These 3 options are rated according to Recall, Reduction and Rc-measure for each trial.

A. Results

The average Recall of the first option was 37%, with Reduction equal to 97% and Rc-measure of 41%. The second option presented Recall equal to 45%, Reduction of 93% and 48% of Rc-measure. The third one obtained average Recall of 52%, Reduction of 90% and 57% of Rc-measure. Accordingly, we observed that the third option is the best according to the average values of Rc-measure.

The result of this study shows that clustering of similar tasks by interaction (RQ1.A) built at least one cluster that had only 6% (94% of Reduction) of the artifacts available in the project, but that had 77% of relevant artifacts (Recall) for a new task. The conclusion is that the answer is yes to RQ1.A when combined LNS_SMC with MQ. In 8 rounds, the first cluster more similar textually with the new task also presented the highest Recall. However, in 10 trials, the 3 clusters most similar textually to the new task did not present maximum Recall, among these, in 4 trials (20%) there are no correlation between similarity by interaction and textual. The conclusion is that the answer is yes also for RQ1.B, the study showed that there is evidence of correlation between textual similarity and similarity by interaction in software development tasks.

V. Related Work

Hipikat uses a large number of documents available in the project such as source code, documentation, communications between developers (e-mail, discussion forums), error reporting and test plans. The Hipikat evaluation presented an average Recall of 65% [1]. Antunes et al. [8] proposed a recommendation system to recommend a list of relevant artifacts. The System uses developer interactions in real time and artifact access time to list the most relevant artifacts. Then it uses the relations of the language structure (Java) and textual associations to order the recommended artifacts in a decreasing way of relevance. The evaluation showed an average Recall of 42,7%.

The proposal of Ye et al. [9] lists a classification of relevant artifacts (top 10) to correct an error in the software considering the information of the project history. The evaluation indicates that the proposal can recommend relevant artifacts in 70% of the recommendations. CodeRAnts is a recommendation method for recommending artifacts. This Method is based on the repetition of the textual searches performed by the programmers and on the metaphor of the ant colony [10]. The evaluation was performed in a simulated environment, in this environment CodeRAnts obtained an average Recall of 71%. Almhana et al. [11] have shown that to recommend relevant artifacts to support the developer in the correction of an error can be mitigated as a multi-objective problem, maximizing the relevance of the recommended artifacts and minimizing the number of recommended artifacts. The evaluation showed strong evidence that the proposal may recommend lists with relevant artifacts: Recall @ 5 = 72%; Recall 10 = 81%; Recall 15 = 87%; Recall 20 = 94%.

VI. Conclusion

This paper presents part of Tacin method to recommend artifacts to compose the context of new tasks in software projects. Tacin makes use of Clustering and Natural Language Processing techniques. The clustering of task history in similar tasks by interaction presented a Recall of 77%. The next study will consider the degree of relevance between artifact and task. The textual association of the new task with the similar task groups to produce artifacts recommendation with 52% of Recall. This study uses only task *short description* field. The next study should consider other text information from task history, e.g., comments from developers while performing tasks. In addition, other techniques such as dw-cosine [26], an extension of the cosine of similarity, and also techniques for small texts with grammatical errors need to be evaluated.

Acknowledgment

This work was partially supported by the Brazilian funding agencies CAPES and CNPq and Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] D. Čubranić, G. C. Murphy, J. Singer, and K. S. Booth, "Hipikat: a project memory for software development," *IEEE Transactions on Software Engineering*, vol. 31, no. 6, pp. 446–465, Jun. 2005.

[2] M. Kersten, "Focusing knowledge work with task context," University of British Columbia, 2007.

[3] D. W. McDonald and M. S. Ackerman, "Expertise Recommender: A Flexible Recommendation System and Architecture," in *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*, New York, NY, USA, 2000, pp. 231–240.

[4] Z. Wang, H. Sun, Y. Fu, and L. Ye, "Recommending crowdsourced software developers in consideration of skill improvement," presented at the ASE 2017 - Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering, 2017, pp. 717–722.

[5] I. Roediger Henry L., "Relativity of Remembering: Why the Laws of Memory Vanished," *Annu. Rev. Psychol.*, vol. 59, no. 1, pp. 225–254, Dec. 2007.

[6] M. P. Robillard, W. Maalej, R. J. Walker, and T. Zimmermann, Eds., *Recommendation Systems in Software Engineering*. Berlin Heidelberg: Springer-Verlag, 2014.

[7] M. Andric, W. Hall, and L. Carr, "Assisting artifact retrieval in software engineering projects," presented at the Proceedings of the 2004 ACM Symposium on Document Engineering, 2004, pp. 48–50.

[8] B. Antunes, J. Cordeiro, and P. Gomes, "An Approach to Context-based Recommendation in Software Development," in *Proceedings of the Sixth ACM Conference on Recommender Systems*, New York, NY, USA, 2012, pp. 171–178.

[9] X. Ye, R. Bunescu, and C. Liu, "Learning to rank relevant files for bug reports using domain knowledge," presented at the Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering, 2014, vol. 16-21-November-2014, pp. 689–699.

[10] I. Caicedo-Castro and H. Duarte-Amaya, "CodeRAnts : A recommendation method based on collaborative searching and ant colonies , applied to reusing of open source code," 2015.

[11] R. Almhana, W. Mkaouer, M. Kessentini, and A. Ouni, "Recommending relevant classes for bug reports using multi-objective search," presented at the ASE 2016 - Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering, 2016, pp. 286–295.

[12] A. Moraes, E. Silva, C. da Trindade, Y. Barbosa, and S. Meira, "Recommending Experts Using Communication History," in *Proceedings of the 2Nd International Workshop on Recommendation Systems for Software Engineering*, New York, NY, USA, 2010, pp. 41–45.

[13] T. Fritz, G. C. Murphy, E. Murphy-Hill, J. Ou, and E. Hill, "Degree-of-knowledge: Modeling a developer's knowledge of code," *ACM Transactions on Software Engineering and Methodology*, vol. 23, no. 2, 2014.

[14] X. Xie, W. Zhang, Y. Yang, and Q. Wang, "DRETOM: Developer Recommendation Based on Topic Models for Bug Resolution," in *Proceedings of the 8th International Conference on Predictive Models in Software Engineering*, New York, NY, USA, 2012, pp. 19–28.

[15] X. Xia, D. Lo, X. Wang, and B. Zhou, "Accurate developer recommendation for bug resolution," presented at the Proceedings - Working Conference on Reverse Engineering, WCRE, 2013, pp. 72–81.

[16] J. Zhu, B. Shen, and F. Hu, "A learning to rank framework for developer recommendation in software crowdsourcing," presented at the Proceedings - Asia-Pacific Software Engineering Conference, APSEC, 2016, vol. 2016-May, pp. 285–292.

[17] Y. Malheiros, A. Moraes, C. Trindade, and S. Meira, "A Source Code Recommender System to Support Newcomers," in *2012 IEEE 36th Annual Computer Software and Applications Conference*, 2012, pp. 19–24.

[18] B. Ashok, J. Joy, H. Liang, S. K. Rajamani, G. Srinivasa, and V. Vangala, "DebugAdvisor: A recommender system for debugging," presented at the ESEC-FSE'09 - Proceedings of the Joint 12th European Software Engineering Conference and 17th ACM SIGSOFT Symposium on the Foundations of Software Engineering, 2009, pp. 373–382.

[19] M. C. Monçores, A. C. F. Alvim, and M. O. Barros, "Large Neighborhood Search applied to the Software Module Clustering problem," *Computers & Operations Research*, vol. 91, pp. 92–111, Mar. 2018.

[20] E. Cambria and B. White, "Jumping NLP Curves: A Review of Natural Language Processing Research [Review Article]," *IEEE Computational Intelligence Magazine*, vol. 9, no. 2, pp. 48–57, May 2014.

[21] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data Clustering: A Review," *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264–323, Sep. 1999.

[22] J. Han, M. Kamber, and J. Pei, "10 - Cluster Analysis: Basic Concepts and Methods," in *Data Mining (Third Edition)*, Third Edition., J. Han, M. Kamber, and J. Pei, Eds. Boston: Morgan Kaufmann, 2012, pp. 443–495.

[23] R. Levy and C. Manning, "Is It Harder to Parse Chinese, or the Chinese Treebank?," in *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, Stroudsburg, PA, USA, 2003, pp. 439–446.

[24] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, vol. 14, no. 2, p. 131, Dec. 2008.

[25] V. R. Basili, "Software Modeling and Measurement: The Goal/Question/Metric Paradigm," University of Maryland at College Park, College Park, MD, USA, 1992.

[26] B. Li and L. Han, "Distance Weighted Cosine Similarity Measure for Text Classification," in *Intelligent Data Engineering and Automated Learning – IDEAL 2013*, 2013, pp. 611–618.