A Content-Based Approach for Recommending UML Sequence Diagrams

Thaciana Cerqueira Leandro Marinho Franklin Ramalho

Department of Computer and Systems
Federal University of Campina Grande, Brazil
thaciana@copin.ufcg.edu.br
{lbmarinho,franklin}@dsc.ufcg.edu.br

Abstract

Software engineers usually have to face a large space of choices during the development process, including libraries/APIs, frameworks and UML models, which undermines their ability in finding the ones that best fit their needs. Recommender Systems appear as a solution to this problem since they have been applied successfully in other domains that suffer from similar issues. In this paper we propose to recommend UML Sequence Diagrams, a popular software artifact in many development processes, as an attempt to mitigate this problem. Our approach consists of: (i) a suitable representation of the users' information needs and sequence diagrams' content; and (ii) two content-based recommendation algorithms to recommend sequence diagrams that match the users' preferences. We performed a study with computer science subjects, where we generated recommendations with (ii) and measured the users' satisfaction upon these recommendations. Our preliminary results show that both algorithms are able to provide accurate recommendations.

1. Introduction

Created by OMG (Object Management Group) [1], the Unified Modeling Language (UML) describes a set of diagram types for describing different systems from various perspectives. Unfortunately, support for discovering and reusing diagrams is currently limited. From the many different types of media that can be retrieved through Web search engines (e.g. Google), software artifacts are not among them. For example, there is no easy way for a user express that she is searching for examples of real world UML Sequence Diagrams that use combined fragments and that asynchronous messages.

We propose to mitigate this kind of problem through recommender systems. Recommender Systems (RS) are great tools for filtering out and helping users to find relevant content. In some cases, they try to mimic the situation where the information needs of users are fulfilled by recommendations of like-minded or expert users [2]. Recommender Systems for Software Engineering (RSSEs), in particular, are software tools that can assist users in the activity of finding software artifacts. Moogle [3] is a search engine that uses metadata of software models for retrieving software artifacts. Our work is similar to this in the sense that we also exploit metadata information of software models (in this case UML Sequence Diagrams) to help users finding software artifacts of interest, but we do this by providing personalized recommendations based on the declared information needs of users.

RSSEs is a scarce area of research that has the potential to increase the quality and agility of software development [4]. While most of the related works in this area focus on source code recommendation (cf. Section 5), we investigate the recommendation of UML sequence diagrams, a yet unexplored problem in this field of research. This kind of recommendation has an important educational benefit for students and teachers since it allows one to find good examples of real world diagrams containing the features they want to learn/review.

Our approach consists in (i) a content-based recommender system's representation for sequence diagrams' features and (ii) the application of two classic content-based algorithms: a bag-of-words model borrowed from information retrieval [5], and another one based on our proposed representation [2]. We compare and evaluate both algorithms by means of a field experiment with computer science subjects well acquainted with UML diagrams and show that both approaches present reasonably accurate recommendations.

The remainder of this paper is organized as follows. Section 2 presents a brief background on RS and UML sequence diagrams. Section 3 introduces our approach. Section 4 presents the evaluation methodology and results. Section 5 presents related works and Section 6 concludes the paper with some final remarks.

DOI reference number: 10.18293/SEKE2016-147

2. Background

In this section, we present a brief summary of the main concepts related to this work: Recommender Systems, Bag of Words, and UML Sequence Diagrams.

2.1. Recommender Systems

RS are applications that aim to support users on their decision making process while interacting with large amounts of information. RS usually follow four main paradigms [6]: (i) **Collaborative-filtering**: The assumption is that users that shared similar interests in the past tend to share similar interests in the future. Collaborative-filtering algorithms usually rely on the historical data of users; (ii) **Content-based**: This type of recommendation is based on the content of the items being recommended; (iii) **Knowledge-based**: In this modality, the recommender algorithms exploit background knowledge about the recommendable items; (iv) **Hybrid**: Since each of the aforementioned approaches has advantages and disadvantages, a good combination of them would take advantage of the strengths and eliminate the weaknesses of each one.

2.2. Bag-of-Words

Many information retrieval systems represent queries and textual documents as a multiset of words [5]. This multiset, B, can be described as $B = \{(w_i, f(w_i))|1 \leq i \leq j\}$, where j is the amount of words of B, and f is a function that returns the number of occurrences for the word w_i in the current document. As an example, if we get a document composed by the sentence "sequence diagrams are UML diagrams", we could represent B as

$$B = \{(sequence, 1), (diagrams, 2), (are, 1), (UML, 1)\}$$

This representation does not store the semantics, since there is only the number of occurrences for each word and the order it occurs does not matter for the model. For example, the text fragments *sequence diagrams are UML diagrams* and *UML diagrams are sequence diagrams* have very different semantic meanings, but have the same bag-of-words representation [7].

2.3. UML Sequence Diagram

The UML enables software developers to represent system models through different views. Each one of these views is modelled with abstractions called UML diagrams, which can represent structural and behavioural characteristics of the object-oriented paradigm. UML defines thirteen diagrams [1], where each one is responsible for representing one or more features of the software being built. In order to represent the system behaviour, interaction diagrams allow the specification steps of interaction between objects and actors of the system. Below we describe some important elements that may compose a sequence diagram: (i) **Lifeline**: Represents system objects that participate in the exchange of messages. Figure 1 depicts two lifelines: *Web Customer* and *Bookshop*; (ii) **Message**: Represents the exchange of information between entities of an OO system. Figure 1 shows asynchronous and return messages. The message *1:search inventory*, sent from *Web Customer* and received by *Bookshop*, is an asynchronous message where a line is drawn with a stick arrowhead; (iii) **Combined Fragments**: In its 2.0 version, combined fragments were added to sequence diagrams, *i.e.* new elements capable of changing the normal flow of execution. The combined fragments have 12 types (operators) and each one of them has a different meaning. Figure 1 shows the *loop* and *opt* combined fragments, where *opt* is nested with *loop*. While *opt* defines a fragment that must be executed only if the analysis of the expression on the guard returns a true value, the *loop* describes behaviours that need to execute the same sequence of commands iteratively; (iv) **Interaction Use**: Allows reusing other sequence diagrams. In Figure 1 the interaction use is represented by *ref*.

Figure 1 shows two objects that interact by means of messages to perform a purchase of books online. While the customer is buying, the interactive combined fragment *loop* is being executed. Within the interaction, the client may optionally view the description of the chosen book and purchase a book. When a customer completes a purchase, it must end the session according to the behavior specified in the sequence diagram checkout.

3. Our Approach

Our approach consists of: (i) suitable representations for the sequence diagrams and the users (Section 3.1); and (ii) algorithms that operate under the representation defined in (i) (Section 3.2). As a proof of concept, we compared two of such algorithms: Bag-of-Words (BoW) model borrowed from the information retrieval field and a Content-Based (CB) filtering algorithm. The reason for choosing recommendation algorithms that relay on the content of sequence diagrams instead of, e.g. collaborative filtering, is due to the fact that we did not find any publicly available repository of interaction data between users and sequence diagrams.

3.1. Sequence Diagram Features

Before generating recommendations, we need to choose a suitable representation for users' and items' profiles. To

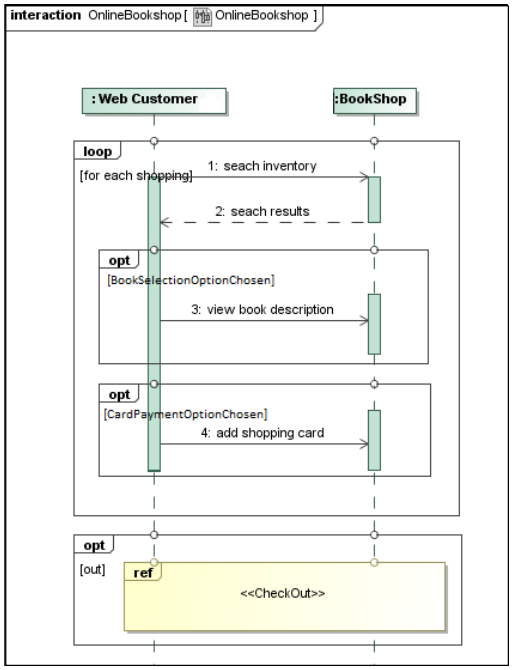

Figure 1: Example of a Sequence Diagram

this end, we have chosen to represent users and items as vectors in a space of features extracted from sequence diagrams. Below we describe the features we used for representing them. Follows the established representation: Presence of Lifelines (NL): We divided the number of lifelines into three groups, namely: small (from 1 to 3 lifelines), medium (from 4 to 6 lifelines) and large (over 6 lifelines) diagrams. Presence of Messages: Asynchronous(PAM); Return (PRM); Create (PCM); Delete (PDM); Presence of Combined Fragment: Conditional (PCoCF); Iterative (PItCF); Break (PBCF); Concurrent (PCuCF); Weak (PWCF); Strict (PSCF); Negation (PNCF); Critical (PCrCF); Ignore (PIgCF); Consider (PCsCF); Assertion (PACF); Presence of Interaction Use (PIU); Presence of Actors (PAc); Presence of State Invariant (PSI).

In the CB approach the user and item profiles are defined by binary vectors, as described above, of the form

$$\vec{p} = (\text{NL}, \text{PAM}, \text{PRM}, \ldots, \text{PACF}, \text{PIU}, \text{PAc}, \text{PSI})$$

In the BoW model, users and items are represented as vectors of strings extracted from the sequence diagrams XMI file. Each string represents a sequence diagram feature, e.g. “uml:Lifeline” and “uml:Actor”, as can be seen in column ”BoW” shown in Table 1. Each string is weighted with the well known tf-idf (term frequency - inverse document frequency) scheme where the string weights are directly proportional to their frequency in the sequence diagram and inversely proportional to their appearance over all sequence diagrams in the repository. In information retrieval terms, the user is the query and the sequence diagrams are the documents.

3.2. Recommender Algorithm

CB algorithm: A RS can be formally described as a function where U and I are the set of users and items (sequence diagrams in our case), and s is a function that estimates the utility of $i \in I$ to $u \in U$.

$$s : U \times I \rightarrow \mathbb{R} \tag{1}$$

The recommendation list is computed in two steps. First the similarities between the target user and the item profiles are calculated. Next, the n nearest sequence diagrams are recommended (aka top-n recommendation) to the user, according to Section 3.1. For the similarity computation we have used the well known and widely used cosine similarity measure. The cosine similarity receives two vectors as input and returns 1 if they have maximum similarity and 0 otherwise. More formally, for two m-dimensional profile vectors $\vec{x}$ and $\vec{y}$ the cosine similarity is computed by

$$sim(\vec{x}, \vec{y}) = \frac{\sum_{i=1}^{m} x_i \cdot y_i}{\sqrt{\sum_{i=1}^{m} x_i^2} \cdot \sqrt{\sum_{i=1}^{m} y_i^2}} \tag{2}$$

where x_i and y_i are the i-th component of vectors $\vec{x}$ and $\vec{y}$ respectively. Now, the top-n items for the target user $u \in U$ are computed as follows:

$$\text{top-}n(u) := \underset{i \in I}{\text{argmax}}^{n}\ sim(\vec{u}, \vec{i}) \tag{3}$$

BoW algorithm: A BoW can be formally described as a set of documents $C = \{D_1, \ldots, D_l\}$, where l is the amount of documents, and each document $D_p \in C$ is represented as a multiset $D_p = \{(w_{p_q}, f(w_{p_q})) | 1 \leq q \leq m\}$, where m is the amount of words of D_p and $f(w_{p_q})$ is a function that returns the number of occurrences for the word w_{p_q}. Each word w_{p_q} is extracted from the text file representation for D_p. Considering the diagrams at XMI format as text files (since the format is XML-based), base in an information retrieval algorithm [7].

Each user Y_r is represented as a set $Y_r = \{t_1, \ldots, t_w\}$ of search terms. For each user, the algorithm calculates an score (Equation 4) for every document $D_p \in C$.

$$score(Y_r, D_p) = \sum_{s \in Y_r} \text{tf-idf}_{s, D_p} \tag{4}$$

Lastly, the top-n items for the user $Y_r \in V$ are computed as follows (Equation 5):

$$\text{top-}n(Y_r) := \underset{z \in C}{\text{argmax}}^{n}\ score(Y_r, z) \tag{5}$$

4. Evaluation

In this section we present the experimental protocol used, results and threats to validity. In order to establish a proof of concept to the proposed profiles and also discover which algorithm makes the best recommendations, we have performed an experiment for comparing the proposed UML Sequence Diagrams recommenders.

4.1. Planning:

For evaluating the recommendations, the user and item profiles are created according to the preferences indicated by participants and the diagrams parsed by the system, respectively. Next, for each recommendation algorithm, the recommendations are generated and displayed to the participants that in turn must accept or not the recommended sequence diagrams displayed in a top-5 list. Finally, before the end of the experiment, the participants must judge the experiment as good suggestions or not, using the following levels of satisfaction: {Very Satisfied, Satisfied, Indifferent, Dissatisfied, Very Dissatisfied} (aka likert scale). [8].

4.2. Collecting and Formatting Data:

Given that we did not find any publicly available repository of sequence diagrams, we decided to construct our own experimental database. For that, we used the *Magic Draw* tool[1] to reverse engineer source code from the *FindBugs*[2] project. This tool also allowed us to recover and parse the XMI file generated from sequence diagrams. By parsing the XMI file, we identified all elements composing the sequence diagram. From this process, we were able to build 24 diagrams. Additionally, we generated 20 diagrams manually. These diagrams were generated because some features (e.g. PBCF, PWCF) were missing in the diagrams generated by Magic Draw. They were based on examples of literature, totaling 20 examples. For formatting the user profiles, each subject from the experiment must answer the questionnaire described in Table 1, column "*Are you interested in viewing...*". Each question relates to a specific features vector and the user should inform which features are of his/her interest. After constructing the user profiles, we calculated the cosine similarity between the target user and the item profiles vector and computed top-5 recommendation lists for BoW and CB approach.

Table 1 contains some examples about the user profiles.

[1] `http://www.nomagic.com/products/magicdraw.html`
[2] http://findbugs.sourceforge.net

Table 1: The example form of interest: Content-based mapping (CB), Bag-of-words mapping (BoW)

Id	Are you interested in...	CB	BoW
1	diagrams of which size? (Small or Medium or Large)	NL	"uml:Lifeline"
	messages...		
2	↪creation message? (Yes or No)	PCM	"uml:Message", "message-Sort=createMessage"
	combined fragment...		
3	↪combined fragments of type assertion?? (Yes or No)	PACF	"uml:CombinedFragment", "interactionOperator=assert"
	sequence diagrams with...		
4	↪interactions of use ("req")? (Yes or No)	PIU	"uml:InteractionUse"
5	↪actors ("actor")? (Yes or No)	PAc	"uml:Actor"
6	↪invariant state? (Yes or No)	PSI	"uml:StateInvariant"

4.3. Selection of Subjects:

The experiments were conducted with volunteer students the software design discipline in the computer science course from the Federal University of Campina Grande. The experiment included 26 participants. The approach was presented to the group in a workshop where instructions for using the tool were explained.

4.4. Experimental Design:

We had an unpaired comparative experiment where, for each participant, we randomly selected the recommendation algorithm, being transparent to the user which method was used. Thus, half of the subjects received recommendations from the BoW model and the other half from the CB approach. The effectiveness of the system is related to its ability to perform good recommendations. Thus, the central question of this research was to investigate the accuracy and the level of satisfaction of the users considering the recommendations of the two approaches investigated. As evaluation metrics we used *precision* (Equation 6) which is a well known and widely adopted metric in the information retrieval and recommender systems literature [9]. The precision for a given user $u \in U$ is defined as follows:

$$precision = \frac{|\{relevant\ items\} \cap \{recov.\ items\}|}{|\{recov.\ items\}|} \quad (6)$$

4.5. Questions and Hypothesis Formulation:

Our experiment addresses the research questions, using the null hypotheses respectively. RQ1: Given the same user profile, does the content-based and bag-of-words differ in

precision? and $H_{1\text{-}0}$: The precision of the two approaches is equal. RQ2: From the point of view of user satisfaction, which of the two approaches is better? and $H_{2\text{-}0}$: The satisfaction for both approaches is the same.

4.6. Results

We used the *Wilcoxon-test* on the precision values computed for each subject, which reached a *p-value* of 0.3543, indicating that there is no statistical difference between the CB and BoW approaches. Thus, it was not possible to refute $H_{1\text{-}0}$ presented in Section 4.5. As a reinforcement, we calculated the *Cohen d* value. Cohen [10] suggests a scale able to identify the impact of the intended effect that, in this experiment, is the difference of the precision between the algorithms. On this scale, a value of up to 0.2 is considered of small effect, from this one up to 0.5 is a medium effect, 0.8 and above represents a large effect. Our results for this calculation is 0.4303, which represents almost no effect.

From the evaluation results, the answer to RQ1: *Given the same user profile, the content-based and bag-of-words differ in precision?* No. Figure 2 shows the boxplot of the precision for both approaches. The CB approach has the highest precision in terms of the median, and this result may provide a positive evidence of the relevance of this algorithm, that will be further analyzed.

Figure 2: Boxplot - Comparison of Approaches

From a total of 44 sequence diagrams from the database, the experiment recommended 24 sequence diagrams from the total of 26 searches. Figure 3 shows the distribution of responses by recommended and accepted, summarizing the number of recommendations and sequence diagrams accepted for each of the approaches analyzed. This result demonstrates that the acceptance of the approaches is relevant and that users accepted more sequence diagrams recommended by the CB algorithm.

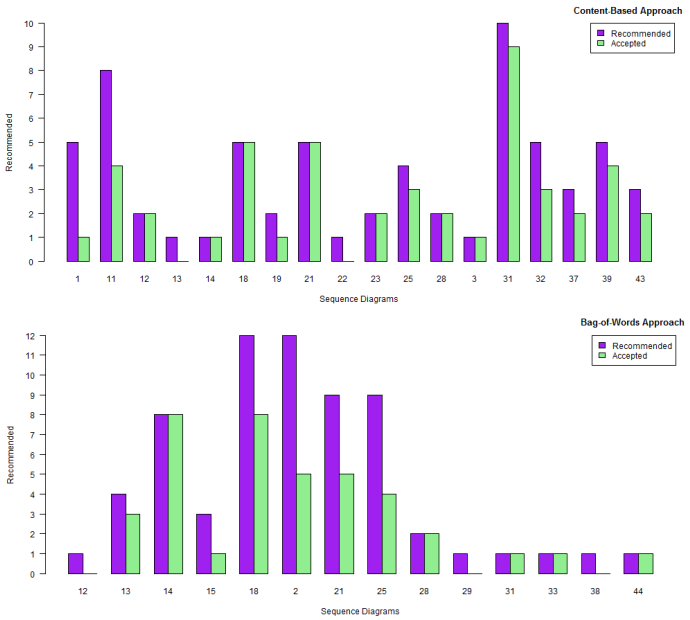

Figure 3: Recommendations Accepted by Approaches

Considering the sequence diagram characteristics displayed in Table 1, we can see that some characteristics were searched, selected and recommended more often than others by the users, such as asynchronous and return, in addition to combined fragment of the conditional (*opt* and *alt*) and iterative (*loop*) type. The most frequently chosen characteristics are also the most frequently recommended ones by the approaches, as we can see in Figure 4. One possible interpretation of this result is that both approaches are recommending sequence diagrams that match the characteristics requested by the users, but also that the subjects had a bias towards features they are already familiar with.

Regarding the answer to RQ2: *From the point of view of user satisfaction, which of the two approaches is better?* Two approaches achieve equal performance. The *Kruskal-Wallis chi-squared* indicates that there is a no statistical difference between the CB and BoW as concerning satisfaction, because *p-value* of 0.2987 and *Vargha-Delaney A measure* return *A measure* exactly 0.5.

4.7. Threats to Validity

In this section we describe the main threats to the validity of this work. **Internal Validity**: We had a low number of

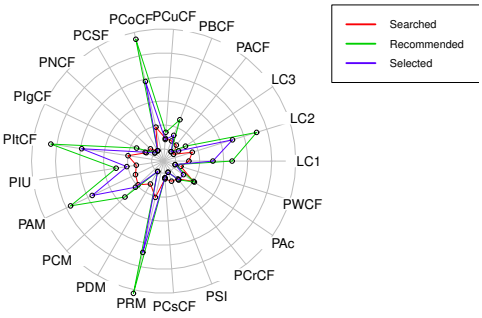

Figure 4: Item Profile Characteristics

sequence diagrams, which may result in cases where there is no sequence diagram covering the features of interest of the users. To address this threat, we manually created sequence diagrams to increase the number of characteristics available. **External Validity**: We have a low number of participants, which may lead to problems related to the statistical significance of the results.

5. Related Work

Most works aim to increase the reusability, providing ease of maintenance, improving productivity and making suggestions according to the preferences of the developer. Cubranic et al. [11] exploits recommender systems for bug fixes. Ye et al. [12], Lozano et al. [13], and McCarey [14] propose to recommend classes and methods based on the current class being used by the developer. Other works go beyond recommendation methods and indicate artifacts based on the bug fix process or recommend design patterns. The aforementioned works are important and represent an emerging area where information retrieval and recommender system techniques are used for searching and recommending software engineering artifacts. This paper complements these works by being one of the first research efforts (according to the reviewed literature) on using sequence diagrams recommendations.

6. Conclusion and Future Work

In this paper we proposed content-based recommender systems for recommending sequence diagrams. We have compared two recommendation models: a bag-of-words model borrowed from the information retrieval field and a classic CB algorithm. We conducted an experiment where we did not find statistical difference between the approaches considered. However, users accepted more sequence diagrams recommended by the CB algorithm. It is important to emphasize that when filling the user profile by the users the most frequently chosen characteristics are also the most frequently features present in the diagrams recommended by the approaches. Hence, our study can serve as basis for future works aimed at identifying the main behavioral features of interest to developers when specifying UML design. As an ongoing work, we are currently investigating several other recommendation algorithms to be incorporated in our approach. In order to address current threats and reach more precise results, we intend to design and perform a new experiment with a larger database of user profiles and sequence diagrams.

7. Acknowledgments

This work was partially funded by the EU-BR BigSea project (MCTI/RNP 3rd Coordinated Call).

References

[1] OMG. Introduction to OMGś unified modeling language (UML). [Online]. Available: http://www.omg.org/gettingstarted/what_is_uml.htm

[2] X. Amatriain, A. Jaimes, N. Oliver, J. M. Pujol, F. Ricci, L. Rokach, and B. Shapira, *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds. Springer US, 2011.

[3] D. Lucrédio, R. de M. Fortes, and J. Whittle, "Moogle: a metamodel-based model search engine," *Software & Systems Modeling*, vol. 11, no. 2, pp. 183–208, 2012.

[4] M. Robillard, R. Walker, and T. Zimmermann, "Recommendation systems for software engineering," *Software, IEEE*, vol. 27, no. 4, pp. 80–86, July 2010.

[5] D. Metzler, "Beyond bags of words: Effectively modeling dependence and features in information retrieval," *SIGIR Forum*, vol. 42, no. 1, Jun. 2008.

[6] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich, *Recommender Systems: An Introduction*. Cambridge University Press, 2010.

[7] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*, 2008.

[8] Q. Li, "A novel likert scale based on fuzzy sets theory," *Expert Syst. Appl.*, vol. 40, no. 5, pp. 1609–1618, Apr. 2013.

[9] P. Resnick and H. R. Varian, "Recommender systems," *Commun. ACM*, vol. 40, no. 3, pp. 56–58, Mar. 1997.

[10] S. Piasta and L. Justice, *Cohen's d statistic*. Thousand Oaks, CA: SAGE Publications, Inc., vol. In N. Salkind (Ed.), Encyclopedia of research design.

[11] D. Cubranic, G. Murphy, J. Singer, and K. Booth, "Hipikat: a project memory for software development," *Software Engineering, IEEE Transactions on*, no. 6, pp. 446–465, June 2005.

[12] Y. Ye and G. Fischer, "Reuse-conducive development environments," *Automated Software Eng.*, vol. 12, no. 2, pp. 199–235, Apr. 2005.

[13] A. Lozano, A. Kellens, and K. Mens, "Mendel: Source code recommendation based on a genetic metaphor," in *ASE, 2011 26th IEEE/ACM International Conference on*, Nov 2011, pp. 384–387.

[14] F. McCarey, "Agile software reuse recommender," in *ICSE 2005. Proceedings. 27th International Conference on*, May 2005, pp. 652–.