Time-Constrained Requirements Elicitation: Reusing GitHub Content

Roxana Lisette Quintanilla Portugal,
Julio Cesar Sampaio do Prado Leite
Dept. de Informática PUC-Rio
Rio de Janeiro, Brasil
rportugal@inf.puc-rio.br, julio@inf.puc-rio.br

Eduardo Almenero
Dept. de Matemática
UFRRJ
Rio de Janeiro, Brasil
almenero@ufrj.br

Abstract—Requirements elicitation is the activity of identifying facts that compose the system requirements. One of the steps of this activity is the identification of information sources, which is a time-consuming task. Text documents are typically an important and abundant information source. However, their analysis to gather useful information is also time consuming and hard to automate. Because of its characteristics, the identification of information sources and analysis of text documents are critical in time-constrained projects, which are typically addressed through agile approaches. This paper presents a strategy for time-constrained elicitation, which is based on mining GitHub content. The strategy aims the identification of information sources (similar projects) and the automatic analysis of textual documents (projects content) through text mining techniques. Furthermore, it maintains the traceability between the data mined and its sources, boosting the reuse of existing information. A tool is being created to support the strategy.

Index Terms—Requirements elicitation, reuse of information, text-mining, just-in-time requirements, time-constrained requirements.

I. INTRODUCTION

Discovering of requirements, as referred by Meth [1], has two main differentiated processes: requirements identification and requirements transformation, which can be performed manually or with the aid of a tool. The first one comprises the activity of separating texts describing requirements from texts that are not relevant from the requirement point of view. In this sense, several authors have made efforts to discover or mine texts from a corpus of documents for the following purposes: disambiguation [2], abstractions for problem understanding [3], domain understanding [4], and non-functional requirements classification [5] among others.

To perform requirements elicitation when the source of information are text documents we usually use reading techniques to analyze the content. However, before performing this task it is necessary to find effective sources of information, which is a time-consuming task in traditional development settings [6], hence, just-in-time requirements usually takes place. This lightweight approach is very usual in agile projects [7] and also practiced in software startups which, inspired by the lean startup approach [8], build only what is needed as they have to deal with time-to-market constraints. In this scenario, professionals may miss out on techniques in requirements engineering [9], particularly in requirements elicitation.

To address this issue, we propose an approach to reuse information from GitHub, which is widely used by the software development community for the reuse of code. It owns a model that enforces the use of its artifacts to perform a social network interaction. Artifacts such as issues, commits, comments, and readme among others, have descriptions that range from simple notes to even requirements [10].

Our experience tracking requirements in GitHub [11] and related literature [10][12][13] made us to see it from another perspective: the one of a requirements engineer. From this perspective, GitHub can serves as a low-cost information source to perform elicitation through the identification of similar projects. For instance, usually the readme of a project has a description of its main functionalities, thus it is an interesting information source for similar projects. However, information from GitHub artifacts are not structured for reuse from the requirements point of view, therefore we must organize this information to support reuse. Our proposal to organize this content is based on text-mining techniques.

The mining of relevant content will assist in learning about the problem domain, the benchmark of similar project features, and thus boosting the reuse of contents (artifacts) that could seem valuable for the requirements engineer. In this regard, GitHub serves for a just-in-time requirement elicitation.

This paper describes the work in progress in mining content from GitHub projects. Chapter II relates similar works taking advantage of GitHub metadata. Chapter III details a process to achieve our proposal and the steps achieved since we started. Chapter IV shows some scenarios of usage and the envisioned tool. Chapter V presents some comparisons with related works. Finally, we conclude, identifying future work.

II. RELATED WORKS USING GITHUB

Since GitHub metadata allows inference of knowledge according to the reader's intention, it is currently being researched by different approaches. This is the case of Gila tool [12], which make inferences according to the contributions made by a developer, allowing to know which developers are experienced in certain topics. Another tool, Reviewerrec [13] recommends the users who have the best profile to accept pull re-
quests (changes pushed to a project) by evaluating their pull requests in other projects, thereby they make effective use of crowdsourcing (contributions from a community).

III. PROPOSED STRATEGY

A. Example of Simple Text Mining on GitHub

To demonstrate how a minimal variation in text mining configuration can lead to different results, a Real Estate (i.e. the business of selling land and buildings) example was used. The goal here was to find texts that can provide knowledge in the existing projects related to the Real Estate concept, thus helping understanding the domain. To achieve this, we created a dataset with only readmes of 1519 projects using the GitHub’s API searching for the words <real> and <estate>, and the best match option selected. However, it is important to understand that the GitHub does not allow for phrase searching, so using "real estate" will return an <and> search result with the attention message saying: "Your query contains a character that is ignored".

From the dataset, we applied text-mining techniques (searching by phrases) using the R language [14], and below we show parts of text retrieved followed by the information (trace) of where they were found.

1) Using: “Real Estate”

```
#Miami-Real-Estate-Agent-Your-Guiding-Star-to-Buy-Best-Apartment-of-your-Choice. If you are looking for a beneficial and stable real-estate agent who can help you in buying a ref=’http://www.luxreale
 estate.com/’>b>C>Luxury Apartments in Miami</b>
</a> then make sure to select the one that is reliable and trustworthy.
You would need the help of real estate agent while buying, marketing or rental a real estate, even if you are searching for the resi-dential or commercial...you would be able to find numerous websites that would take you to the names of best realtors that are available in Miami. Trace: 0630..10...user:RichardJames12..repo:Miami-Real-Estate-Agent-Your-Guiding-Star-to-Buy-Best-Apartment-of-your-Choice..txt

2) Combining: “Real Estate” and “Feature”

* #List #Feature properties
* #Display full details of a property
* #Search Form with autocomplete for location
* Currently there are reserve page name these are:
* #Property - display the single or full details of property.
* #Search Properties - Display the result of the searched properties.
Trace:1502..5..user:wp-plugins..repo:wp-real-estate-property-listing-crm..txt

3) Combining: “Real Estate”, “Feature” and “Issues”

* #PRO Added RSS feature for listings.
* #Fixed an issue on open house feature.
* #Fixed some issues on rooms feature.
* #Added mailto feature for agent info activity.
Trace:1477..8..user:wp-plugins..repo:real-estate-listing-realtyna-wpl..txt.

Nonetheless, we have also found that the term “real estate” is used as an analogy in the context of screens design. As such, generating false positives, as shown below.

```
An option map of additional drawing parameters. I lowered it to 63dpi so it wouldn’t take up so much screen real estate.
Trace: 0953..1..user:Engelberg..repo:instaparse..md.txt
```

In item 1, the mining was performed searching the phrase “real estate”. In the item 2, the mining was performed using the phrases “real estate” and “feature”. It is possible to note that by adding the word “feature”, the search returned results that are more relevant from a requirements specification point of view. Furthermore, in item 3, we added the word “issue” to the searching phrases. The intention here was to find information related to common problems in the domain and, as we can see in the result, the information obtained is more focused on problems.

On the other hand, there are results, where the “real estate” phrase is related to the “screen” domain that is not what we want, thus bringing up an ambiguity problem. It is important to observe that the traceability between the source and the result was preserved (the last line in each item, marked by the word “trace”).

B. Proposing a Process

To systematize our strategy, we have defined the scope of a preliminary process using the SADT modeling technique (Fig. 1. Fig. 2. Fig. 3. ). The macro process activities are: Collate, Distill and Collocate. This process is based on the compilation of text mining techniques used in the environment for statistical computing R Project [14] and the phases approach for mining in [15]. In the first activity, the raw information is collected from GitHub repository. This raw information is refined in the second activity, where the useful information is separated from the useless one. The last activity organizes the information, providing means to its reuse by the requirements engineer.

![SADT Model for mining GitHub](image)

To date, only the first of the macro process activities (Fig. 3. ) was detailed. With the aim to achieve this activity (Collate), a retrieval task was needed. It was realized that GitHub possesses artifacts which are mostly in natural language, among them we selected: issues and readmes. The results up to date for the Retrieve activity, is a script that collects all the projects readmes, given one or more terms for searching. This script keep the recall’ [16] of what would be obtained in a conventional search through the GitHub website. Moreover, the GitHub API settings allowed us to create a strategy to collect repositories using its metadata for sorting by: forks, stars; and ordering by: asc or desc, leading us to obtain results that a user would obtain from a sequence of five searches.

---

1 If a human would perform the task of searching in GitHub. As such, we depart from the same recall.
With the aim to take advantage of GitHub, we plan to text mine making use of requirements engineer metadata to specify requirements, e.g. user stories syntax use modal verbs such as want, be able, allows, among others. We defined some categories (TABLE I.) for semantic filtering as in the approach defined by Sawyer et al. [4].

**TABLE I. PREDEFINED CATEGORIES**

<table>
<thead>
<tr>
<th>Categories</th>
<th>Motive</th>
<th>e.g. terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modal verbs</td>
<td>Expressing possible</td>
<td>Must, wants, allows</td>
</tr>
<tr>
<td></td>
<td>requirements or features</td>
<td></td>
</tr>
<tr>
<td>Concrete Nouns</td>
<td>Expressing possible</td>
<td>User, client, developer</td>
</tr>
<tr>
<td></td>
<td>stakeholders: Place, person,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>organizations</td>
<td></td>
</tr>
<tr>
<td>Quality nouns</td>
<td>Expressing possible</td>
<td>Security, precision, usability</td>
</tr>
<tr>
<td></td>
<td>NFR requirements</td>
<td></td>
</tr>
</tbody>
</table>

IV. SCENARIOS FOR USAGE

Given that natural language tend to be ambiguous, our mining have to take in consideration, that manual analysis and discrimination are best performed by human experts. This section describe some scenarios that our mining strategy could achieve in order to reduce the time of searching and reading contents by humans.

A. Scenarios

1) Discarding the Reading of Projects Without Real Estate Phrase Embedded: GitHub default criteria to search terms does not allow searching phrases. As such, the data set resulted from the GitHub search is pruned by the use of R, which allows for phrase search.

2) Discarding the Reading of Readmes With no Content Inside: GitHub requires a readme artifact for each project created. It is common for users to create dummy readmes just to fulfill this rule. Therefore, the search for these artifacts can return useless results.

3) Finding Live Samples of Projects: The reuse of software is improved when a project inserts a visual example. As such, images or links to the product interface can be found in a readme. These visual clues improves the understanding of the project, giving better background when deciding what to reuse.

4) Boosting Creativity: As stated by Goldin et al. [17]. “The granularity of reuse can vary from reusing a single artifact, such as a component, document, or test case, to reusing a whole product”. Through the learning and comparison of several existing projects, our strategy may be a way of supporting creativity, by facilitating the opportunity for the assembling different parts of projects to create innovative ones.

B. Envisioned Tool

Our final intention is to assist the user in identifying the projects with better opportunities for reuse in the context of fact-finding, instead of the conventional manual search of GitHub. As a result, our strategy would help in the path of approaches such as [17] when time to market is a priority.

Our tool metaphor envision a presentation for the data mined, simplifying the work by filtering results, for instance. One possibility is filtering to help in disambiguation of the dataset of readmes through a cloud of terms classified by nouns referring to the domain of discourse; thus, the user will be able to choose manually the desired one (Fig. 4.). Another possibility is filtering based in predefined categories (such as patterns, as in TABLE I.) by displaying a semantic network (Fig. 5).

V. COMPARISONS WITH RELATED WORKS

The approaches that have been studied have shown the challenges and risks when handling documents in natural language. The AbstFinder tool [3] differs from ours in the dataset they worked, which is a corpus of documents relating to requirements specifications. Our research challenge is the vagueness of the retrieved texts, which are not systematized specifications, but explain what a project does in most of the cases. A second work [4] is more alike, it is intended to use some strate-
gies presented, but taking into consideration the noise present in the texts that can overlap some abstractions when mining. In the tool of Sawyer et al. [4], the corpus of documents has a defined domain, which relates to Air Traffic Control descriptions, interviews, among others. In contrast, our proposal has the challenge to find the documents that belong to a certain domain, i.e. Real Estate was identified to also be part of the Screen Design lexicon. Another related work [15] tackles disambiguation by using LSA techniques applied to treatment of polysemous words, which may help us in our attempt to disambiguate domains.

![Semantic Network of mined information](image)

**VI. CONCLUSIONS AND FUTURE WORK**

The results obtained so far indicate that using GitHub as a repository to boost requirements elicitation can help in the just-in-time requirements problem. Therefore, we are motivated to detail the process activities, using more sophisticated text mining techniques. The intention of the proposed tool is to assist the requirements engineer or any user with the intention to elicit information to reuse it for the construction of requirements. As mentioned in the reviewed literature, we agree that a tool who assist in the manual work of searching sources and reading documents its better than a complete automatized one. As such, we aim to build a tool giving the option to backward to the sources at any time, letting free the navigation of contents according the own user criteria. As a future work, we will explore both terminology lists and ontologies to create more specific queries (like "real estate mortgage" or "real estate agent" on the Real Estate example), as well as search for domain independent patterns (as in **TABLE I.**).

**ACKNOWLEDGMENT**


**REFERENCES**


