Poster: Use of Trace Link Types in Issue Tracking Systems

Mihaela Todorova Tomova, Michael Rath, Patrick Mäder
Technische Universität Ilmenau, Ilmenau, Germany
{mihaela-todorova.tomova,michael.rath,patrick.maeder}@tu-ilmenau.de

ABSTRACT

Issue tracking systems (ITS) are widely used to describe and manage activities in software systems. Within the ITS, software developers create issues and establish typed links among these artifacts. A variety of different link types exists, e. g. that one issue clones another. The rational about choosing a specific link type is not always obvious. In this paper, we study link type selection and focus on the relationship of textual properties of connected issues to picked link type. We performed a study on seven open-source systems and quantified the usage of typed links. Further, we report preliminary results indicating, that depending on link type, a link mostly captures textual similarity of issues and thus may provide only limited additional information.

KEYWORDS

Traceability, Trace Link Evaluation, Trace Link Semantics

ACM Reference Format:
Mihaela Todorova Tomova, Michael Rath, Patrick Mäder. 2018. Poster: Use of Trace Link Types in Issue Tracking Systems. In *ICSE '18 Companion: 40th International Conference on Software Engineering Companion, May 27-June 3, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3183440.3195086

1 INTRODUCTION

The ability to interrelate uniquely identifiable development artifacts is the foundation for software traceability. Many studies have shown that established traceability is beneficial for developing well engineered software [2]. In open-source software (OSS), the developers utilize issue tracking systems (ITS) to organize their work. This includes creating bug reports about defects in the software, and defining features and improvements for future releases. Although not explicitly mandated, they spend lots of effort to manually create typed trace links among these artifacts. Thus, a large body of work [1, 2] investigates, how to automate this manual task termed trace link recovery. However, the link semantics and evaluation, i. e. why a specific link type is chosen, is less studied and requires further research [2]. Thompson et al. [4] investigated issue links and manually defined six categories to study work breakdown. They found, that links express steps of verification and constraints.

In this preliminary study we investigate whether the assigned link type is related to the textual properties of the connected artifacts. Therefore, we first quantify existing link types and their distribution retrieved from seven open-source systems utilizing popular Jira ITS containing 8,446 trace links. Afterwards, we present two scenarios each focusing on a particular link type. The first one deals with *clones*, because this link type has a defined meaning intended to connect identical copies of artifacts. The second scenario focuses on link type *relates*, which is by far the most commonly used type in the dataset. However, representing a basic connection, its link semantics are rather vague and thus need to be studied.

Table 1: Amount of issue interlinking and distribution of link types in the studied dataset.

Project		Linked Issues	Issue Links	Link Type: Blocks	Clones	Duplicates	Relates
DERBY	DE	46%	3,043	144	11	241	2,024
DROOLS	DR	9%	266	0	3	75	119
GROOVY	GR	13%	698	0	12	184	239
INFINISPAN	IN	26%	1,447	0	9	163	679
MAVEN	MA	36%	1,532	6	3	423	914
PIG	PI	21%	760	89	13	102	315
SEAM2	SE	20%	700	0	1	146	366

2 ISSUE MODEL & APPROACH

Issue Model. The basic artifact handled in Jira is termed *issue*. An issue contains multiple properties including a textual "summary", "description", and a "type". Jira offers a list of predefined issue types such as bug report, feature, and improvement. Furthermore, an issue can be connected to other issues using typed, directed *issue links*. Jira provides four predefined link types: *blocker*, *duplicates*, and the already introduced ones *clones*, and *relates*. The type *duplicates* is used, when one issue repeats another issue. *Blocker* models a dependency among two linked issues, such that the source issue I_{src} (i. e. the starting point of the link) needs to be resolved before the target issue I_{tgt} (i. e. the tip of the link) can be resolved. Table 1 shows the distribution of the four link types in our studied dataset [3]. The issue linking greatly varies among the projects, ranging from half of the issues are linked (DERBY) to only every tenth issue (DROOLS).

Approach. The following method is applied to every project in the dataset. First, we applied common IR text preprocessing steps to the combination of summary and description of every issue: stop word removal, lower casing, and stemming. The preprocessed texts serve as corpus and query for the IR search engine Apache Lucene. We configured Lucene with default settings and used BM25 as scoring method. We use two scenarios to study the semantics between linked issues $I_{src} \xrightarrow{?} I_{tgt}$. The first scenario "clones" focuses on issue pairs linked with type *clone* ($I_{src} \xrightarrow{clones} I_{tgt}$), and the second scenario focuses on issues linked with type *relates*. For each scenario, we used the source issue of every link pair as query against the whole corpus containing all issues and captured the ranked result list created by Lucene. We use standard IR metrics of

ICSE '18 Companion, May 27-June 3, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5663-3/18/05.
https://doi.org/10.1145/3183440.3195086

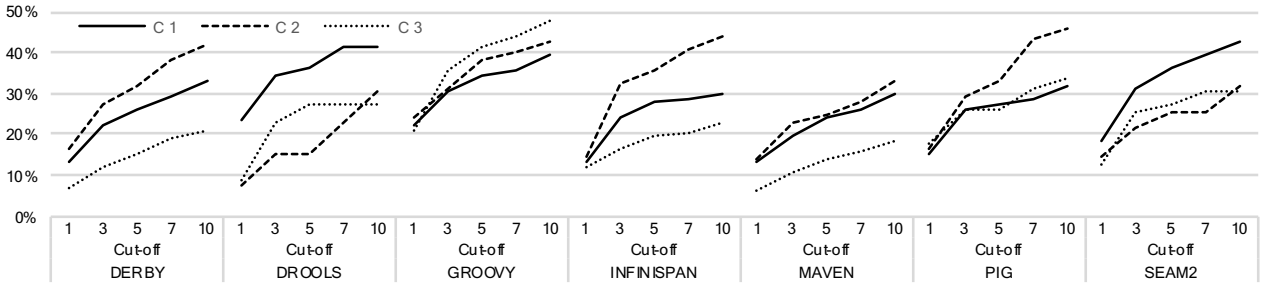

Figure 1: Recall@n for $I_{src} \xrightarrow{relates} I_{tgt}$ categorized by linked issue types: C1 bugs, C2 features/improvements, and C3 mixed.

Table 2: Precision@1 for retrieving I_{tgt}, if $I_{src} \xrightarrow{\textbf{clones}} I_{tgt}$.

	Project						
	DE	DR	GR	IN	MA	PI	SE
Precision@1	45%	67%	91%	100%	100%	46%	0%

precision and recall at certain cut-off points to measure retrieval performance. In scenario "clones", we focus on high precision, i. e. whether I_{tgt} is on top of the result list (precision@1). Since link type *clones* is supposed to link identical issues, I_{tgt} is expected to be the best match. In the second scenario "relates", the number of retrieved issues (ranked list length) until I_{tgt} is found is more important (recall@n). A short list indicates high textual similarity, whereas a long list indicates less or no similarity.

3 PRELIMINARY RESULTS

Scenario 1: "clones". Ideally, two issues I_{src} and I_{tgt} are linked with type *clone*, when the participating issue texts are highly similar or even identical. Therefore, when I_{src} is used as query, I_{tgt} should be on top of the retrieved list. However, this is not always the case, as shown in Table 2. For example, in project Infinispan this was true for all nine clone link pairs, whereas in project Derby only half the clones show high textual similarity. One counterexample is issue Derby-1288 cloning Derby-501 but significantly changes the text and even the issue type from bug to improvement. A manual examination of all 52 clone pairs in the dataset confirms, that indeed differences in issues texts exist. Thus we conclude, that the semantics of link type *clone* seems project dependent, and does not imply highest textual similarity of the connected issues. In this cases, the intentions of the developers are unclear or a mistake and requires further, detailed analysis.

Scenario 2: "relates". Contrasting link type *clones*, it is not unusual for link type *relates* to connect issues with different issues types. Since most issues in every project of the dataset are of type bug, there is a high chance that at least one endpoint of a relates link is a bug. To perform a systematic evaluation, we grouped $I_{src} \xrightarrow{\text{relates}} I_{tgt}$ link pairs as follows. The first issue set $\mathcal{S}_B$ consists of bug reports and the second issue set $\mathcal{S}_{F \cup I}$ consists of features and improvements. Based on the sets, we considered the following three cases. In the first case **C1**, both related issues are bugs $I_{src}, I_{tgt} \in \mathcal{S}_B$, whereby in the second case **C2**, both issues are either feature or improvement $I_{src}, I_{tgt} \in \mathcal{S}_{F \cup I}$. In the third case **C3**, issues of both sets are involved such that $I_{src} \in \mathcal{S}_B \wedge I_{tgt} \in \mathcal{S}_{F \cup I}$ or $I_{src} \in \mathcal{S}_{F \cup I} \wedge I_{tgt} \in \mathcal{S}_B$. Figure 1 shows the averaged recall values at different cut-off points for the three consider cases. The achieved recall values show, that there are differences among the projects as well as the defined cases. Overall, the cases **C1**, **C2**, i. e. when issues with the same issue type are linked *related*, achieve higher recall values, as for case **C3**. In **C3**, bugs are *related* to features and improvements or vice versa, and the compared artifact texts may be written differently. For example, bug reports often contain stack traces usually not found in feature descriptions. However, the overall achieved recall values are quite high. As explained earlier, every query with I_{src} is performed against the corpus of *all* issues in a project. With $\approx 6K$ candidate issues per project, an average achieved recall@10 of 39% e. g. for **C2** is considered high. This indicates, that link type *related* largely captures textual similarity of the connected issues. We argue, that text based link recovery algorithms are likely to propose these link pairs as well. However, in case **C3**, the rational of manually created *relates* links goes beyond textual similarity and may embody project knowledge of the creating developer. Therefore, these pairs are especially interesting and need to be further studied.

4 CONCLUSION & FUTURE WORK

In this paper, we investigated the relationship of issue texts and link types of linked issues in OSS. In our study, we focused on link types *clones* and *relates*. Our preliminary results show, that to a certain degree, the chosen link type also captures textual similarity of the connected issues. For *clones*, the similarity is usually high, as expected. For *relates*, the results vary across projects and connected issues types. Based on the promising initial results, we will extend our study and also include objective similarity of issues.

Acknowledgments We are funded by the BMBF grants: 01IS14026A, 01IS16003B, DFG grant: MA 5030/3-1 and by the EU EFRE/TAB grant: 2015FE9033.

REFERENCES

[1] Giuliano Antoniol, Gerardo Canfora, Gerardo Casazza, Andrea De Lucia, and Ettore Merlo. 2002. Recovering Traceability Links between Code and Documentation. *IEEE Trans. Software Eng.* (2002).
[2] Jane Cleland-Huang, Orlena Gotel, Jane Huffman Hayes, Patrick Mäder, and Andrea Zisman. 2014. Software traceability: trends and future directions. In *Proceedings of the on Future of Software Engineering*. ACM.
[3] Michael Rath, Patrick Rempel, and Patrick Mäder. 2017. The IlmSeven Dataset. In *25th IEEE International Requirements Engineering Conference, RE*.
[4] C. Albert Thompson, Gail C. Murphy, Marc Palyart, and Marko Gasparic. 2016. How software developers use work breakdown relationships in issue repositories. In *Proc. of the 13th Int. Conf. on Mining Software Repositories, MSR*. ACM.