OpenReq Issue Link Map: A Tool to Visualize Issue Links in Jira

Clara Marie Lüders
The Qt Company
Espoo, Finland
clara.luders@qt.io

Mikko Raatikainen
University of Helsinki
Helsinki, Finland
mikko.raatikainen@helsinki.fi

Joaquim Motger
Polytechnic University of Catalonia
Barcelona, Spain
jmotger@essi.upc.edu

Walid Maalej
University of Hamburg
Hamburg, Germany
maalej@informatik.uni-hamburg.de

Abstract—**Managing software projects gets more and more complicated with an increasing project and product size. To cope with this complexity, many organizations use issue tracking systems, where tasks, bugs, and requirements are stored as issues. Unfortunately, managing software projects might remain chaotic even when using issue trackers. Particularly for long lasting projects with a large number of issues and links between them, it is often hard to maintain an overview of the dependencies, especially when dozens of new issues get reported every day. We present a Jira plug-in that supports developers, project managers, and product owners in managing and overviewing issues and their dependencies. Our tool visualizes the issue links, helps to find missing or unknown links between issues, and detects inconsistencies.**

Index Terms—**Requirement Dependencies, Inconsistency Detection, Data-Driven Requirements, Issue Tracking Systems, Recommendation Systems, Release management, Similarity/Duplicate Detection**

I. Introduction

Many software companies keep track of their work in issue tracking systems, such as Jira, Bugzilla, or Github. Stakeholders can describe and store requirements—called issues or tickets in these systems—in the form of epics, user stories, tasks, bugs, feature requests or other defined types. Issues have multiple attributes such as title, description, status, resolution, priority, assignee, release version, or links to other issues. Additionally, stakeholders can comment on issues and track their life-cycle. Depending on the issue tracking system and the users, there can be a variety of different link types. Common types are parent-child, duplicate, dependency, similarity, and work breakdown. In this work, we refer to dependencies between issues as *links* adhering to the terminology in Jira.

Keeping all information regarding issues visible for the user is hard in large issue tracking systems. Especially in open-source projects where a community submits hundreds of tickets weekly [4]. Organizing new issues and planning releases becomes a demanding, time-consuming task: a frustrating situation for developers, project managers, and product owners, as described by Fucci et al. [2].

The Qt Company develops a cross-platform application framework under both commercial and open source licenses. It uses a Jira instance that everybody can submit tickets to as long as an account is created beforehand. Qt's Jira contains bugs, as well as requirements. Qt's public Jira contains over 111,959 issues, out of which 27,462 have at least one outgoing link, totaling 24,857 links in the system as of May 2019.

In Jira's view issue pages, users only see a list of all direct links of a selected issue and not further beyond. For example, in the view issue page of QTBUG-55604 in Qt's Jira, only three issues are visible to users. However, there are 19 issues linked transitively to the issue. Such a map of links can become complex. For instance, there is a link map of size 6755 in which the longest shortest distance is 51 links across multiple Jira projects. Abad et al. presented a systematic literature review about visualization in requirements engineering [1]. The described problem can be elevated to an extent with visualization techniques. Due to the size of Qt's Jira, this alone is not sufficient to solve the problem as there are large networks. Strategies to deal with this information overload need to be developed.

Links can be unknown. A simple example is that a bug might get reported by two different users and the issues are not linked as duplicates. As links may pose constraints to release plans. It is thus important for users to be aware of relevant links as they might point to inconsistencies in future releases. Recommender systems can facilitate this task by looking for comments of users pointing out a link, using NLP to predict links between a pair of issues, or checking the consistency of an issue map concerning the release plan.

Existing Jira plug-ins with visualization for links in Jira do not offer detection for dependencies or checking consistency for releases. Combining these features with visualization such that users can make informed choices is novel in our tool.

With this tool, we aim to ease the described problem by combining visualization with recommender systems. The visualization (see Figure 1) helps understand existing issues and links. It also supports decision making when rejecting or accepting a recommended link from a recommender system. The tool is part of the European Horizon 2020 project OpenReq (https://openreq.eu/). One of the project's goal is to develop an open-source Jira plug-in that any organization can use to help manage its projects. We are evaluating the current implementation of the tool with The Qt Company in their Jira instance.
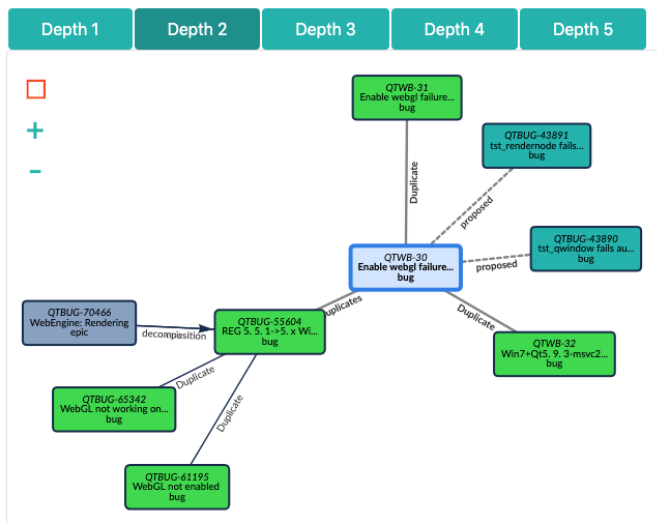
Fig. 1: Issue Map with depth 2 of QTBUG-30 in OpenReq Issue Link Map and Link Detection active

II. OVERALL ARCHITECTURE

The OpenReq Issue Link Map is currently a service-based tool that visualizes the link map of issues in Qt's Jira. We intend to have it as a Jira plug-in at the end of the project. So far, to get a better understanding of the whole picture, users have to explore the links one after another in Jira. With the Issue Link Map visualization users can see all linked issues at a glance. The front-end is a web-based interface while the back-end consists of several OpenReq microservices that work together in a choreographic manner.

III. BACKGROUND SERVICES

A. Graph of Links

Based on issues and their links, a graph of links facilitates the visualization of an issue link map. The graph is maintained as a separate process to allow for a real-time response of a link map. This is necessary for a synchronous issue retrieval and analysis in Jira directly, since many Jira instances contain a large number of issues.

B. Link Detection

Additional links are detected from the issue. To find duplicates, all issues of a single project are taken and their title and description are compared against each other to decide if they have similar content. Moreover, a cross-reference detection checks the comments of an issue for the mention of another issue, which can indicate a link. This happens in open-source settings: Everybody can create issues, but not everybody can create links between issues. To work around this, users often comment if they think a link exists. A maintainer then has to add the link manually.

C. Consistency Checker

The consistency checker [3] verifies that the release plan of an issue link map is consistent by applying constraint solving technologies: First, all child issues, which have the same or higher priority, must not be assigned to a later release. Second, any required issue must not have a later release or lower priority. Third, all links from a duplicated issue are inherited by the duplicate issue.

IV. USER INTERFACE

The user interface consists of two main parts, the visualization and the functionalities explained in III-B and III-C. As seen in Figure 1 the link map is shown as a graph on the left side where a user can select the depth of the map. *Depth* means the distance from the selected issue. The link map can be navigated by clicking any issue on the map. On the right-hand side, users can see general information about the issue from Jira. Above the information box, users can switch to the two features: link detection and consistency checker. When users call the link detection, the system presents a list of five recommended links that can be accepted or rejected. If they accept the link, they must select its type as well. When the users click the consistency checker they see if the issues contained in the issue link map and the corresponding releases are consistent. The different releases are also shown under the result. Additionally, users are able to filter the issues in the link map by different attributes.

V. FUTURE WORK

The tool is still under development and thus the capabilities of the link detection can be improved. For instance, the choice of the users to accept and reject links can be used to train and refine the system, as a human-in-the-loop approach. Additionally, it is planned to visualize what part makes a certain issue link map inconsistent with the option to automatically repair the inconsistency. The Qt trial started in January of 2018 and the tool is being developed and evaluated in this trial. We use AI technologies in this trial to enhance issue tracking systems and make them more manageable. In the long run, we plan to also analyze user acceptance of AI technologies.

REFERENCES

[1] Z. S. H. Abad, M. Noaeen, and G. Ruhe. Requirements engineering visualization: A systematic literature review. In *2016 IEEE 24th International Requirements Engineering Conference (RE)*, pages 6–15, Sep. 2016.

[2] D. Fucci, C. Palomares, X. Franch, D. Costal, M. Raatikainen, M. Stettinger, Z. Kurtanovic, T. Kojo, L. Koenig, A. Falkner, G. Schenner, F. Brasca, T. Männistö, A. Felfernig, and W. Maalej. Needs and challenges for a platform to support large-scale requirements engineering: A multiple-case study. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM '18, pages 19:1–19:10, New York, NY, USA, 2018. ACM.

[3] M. Raatikainen, J. Tiihonen, T. Männistö, A. Felfernig, M. Stettinger, and R. Samer. Using a feature model configurator for release planning. In *Systems and Software Product Line Conference - Vol. 2*, 2018.

[4] C. Stanik, L. Montgomery, D. Martens, D. Fucci, and W. Maalej. A simple nlp-based approach to support onboarding and retention in open source communities. In *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 172–182. IEEE, 2018.