Bilkent University

Department of Computer Engineering

Senior Design Project

ReLink

Project Specification Report

Ayberk Yaşa, Cemhan Kaan Özaltan, Çağatay Şafak, Fatih Kaplama, Görkem Ayten

Supervisor: Eray Tüzün

Jury Members: Erhan Dolak, Tağmaç Topal,

Advisor: Eray Tüzün

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Table of Contents

1. Introduction

In our age of technology, bugs, improvements, and features of a software project are stored and tracked in project management tools that provide development teams with an issue tracker. The rise of agile results in large amounts of data about issues in these project management tools which require much more effort to inspect manually [1]. Moreover, commits and pull requests are fundamental artifacts used in the software development lifecycle (SDLC) [1], [2]. Whereas issues are stored in bug tracking and project management tools such as Jira, Bugzilla, and GitHub Issues, pull requests and commits are stored in servers utilizing a version control system (VCS) such as GitHub, GitLab, and Bitbucket. Such contemporary VCS options offer built-in features for manually linking issues with commits and pull requests through a predefined commit message syntax where the issue ID is explicitly given, allowing for a more structured and traceable project. However, despite the importance of traceability in software development, most commits and pull requests are not explicitly linked to issues by developers due to the lack of the fixed rules [3]. The information contained in the missing links is therefore lost, reducing efficiency in processes such as bug localization, bug prediction, and feature improvements [3]. Since it is highly possible for developers to forget establishing such links or fail tagging a commit due to a typo and the manual cost of recovering these links is very high, we propose an automation tool for tackling this issue.

The following sections of the report are structured as follows: Section 1.1 describes the tool we propose. Section 1.2 evaluates some possible constraints such as implementation, economic, and ethical. Section 1.3 discusses professional and ethical issues that should be considered during the SDLC. Section 2 includes requirements, specifically Section 2.1 explains functional requirements, Section 2.2 is related to non-functional requirements, and Section 2.3 contains pseudo ones. Section 3 provides a competitive analysis giving an insight into whether there is any tool similar to ours and to what extent this is related to ours. Finally, Section 4 contains the references utilized in the report.

1.1. Description

ReLink is a web application which serves as a tool for automating the issue-commit and issue-pull request linking process during software development. To be specific, ReLink detects missing links retrospectively and creates a link between issue-commit pairs or issue-pull request pairs. Moreover, ReLink determines whether a link is constructed between the commit and an issue at the commit stage. If the developers forget to tag the commit with an issue ID, this tool warns the developers and suggests possible issues that may be related to the corresponding commit. This process is also available for opening a pull request. All these are done through certain machine learning (ML) algorithms. ReLink also provides the visualization of the analysis of missed links among commits, pull requests, and issues by certain filters such as by developer, by time, and by reason. Finally, ReLink offers a graphical interface for visualizing data on the historical evolution and progress of a specific project's issues. To be specific, there is a historical graph visualization for links among commits, pull requests, and issues.

1.2. Types of Innovation

We are planning to implement the three types of innovation among the ten in our project. These are Network, Product System, and Service. The reason why we are planning to implement the Network is that we will work in integration with GitHub, Jira, and Azure DevOps. Our purpose is to improve our customers' projects on GitHub, Jira, and Azure DevOps through integrations. The reason for the Product System is that we are planning to implement the dashboard as well as the built-in tools in GitHub, Jira, and Azure DevOps. Thanks to this dashboard, our customers can track their project data. The reason for the Service is that our dashboard makes our project easy to use. Moreover, the innovation that we want to bring to the market is sustained, focusing on the design of a new digital product and service. Therefore, our digital business strategy is digital business transformation.

1.3. Constraints

1.3.1. Implementation Constraints

The project will be implemented as a web application. For the frontend, the React.js [4] framework will be used. For the backend, the Django [5] framework will be used. For data storage, we will use MongoDB [6]. For graph construction and content delivery, the MinIO [7] object storage will be used. PyTorch [8] will be used for training existing state-of-the-art models such as BERT [9], in addition to building any custom neural networks. The models will be trained using Google Colab [10] and external GPUs.

1.3.2. Economic Constraints

The main economic constraints of our project will be the fees of database hosting services, in addition to a web domain fee. The datasets we will fit our models to will be generated from the Jira, Azure DevOps, and GitHub repositories of open-source projects or private repositories to which we have access through an authorization token. Therefore, no expenses will be made for this.

1.3.3. Ethical Constraints

The generated datasets will not be published without authorization for the used private repositories and the ACM Code of Ethics [11] will be followed throughout development.

1.4. Professional and Ethical Issues

1.4.1. Professional Issues

Asynchronous communication through Google Drive to collaborate on documents are planned. Moreover, synchronous communications such as face to face or online weekly meetings are made. Meeting logs about topics discussed, decisions taken, sprint reviews, sprint retrospective, and sprint planning is recorded in the cloud service providers. Besides in-team meetings, biweekly meetings with our supervisor, Eray Tüzün, and several meetings with course instructors, Erhan Dolak and Tağmaç Topal, are made. Also, we will use GitHub as the version control system and Jira as the project management and bug tracking system.

1.4.2. Ethical Issues

We give importance to privacy by design which is a system design that takes into account the data protection at the beginning of the design and architecture of the application [12].

While our application fetches the data of the open source projects, we will need to access the commit, pull request, and issue data of the projects on GitHub and Jira. We will use this data to match pull requests or commits and issues. However, while doing this, we will use encryption algorithms like SHA256 to prevent any data leaks and protect the data of the projects.

2. Requirements

2.1. Functional Requirements

- Developers should be able to link all past PRs and issues manually through possible match suggestions.
  - Using code similarity and other metadata
  - Using commit messages and NLP
  - Using past correctly matched links

- Developers should be given recommendations for linking PRs to issues if they do not provide an explicit link in PR titles or descriptions through a specific convention, e.g., SA-123.
- Developers should be given recommendations for linking commits to issues if they do not provide an explicit link in commit titles or descriptions through a specific convention, e.g., SA-123.
- Developers should be able to receive a warning message when there is a typo or technical thing that prevents linking between a PR and an issue.
- Developers should be able to receive a warning message when there is a typo or technical things that prevent linking between commit and issue.
- Developers should be able to categorize the source code file for the sake of training to allow prediction with higher accuracy when the commit is not manually linked.
- Developers should be able to see the visualization of the analysis of missed links between commits and issues.
    - By developer
    - By time
    - By reason, e.g., a typo, missing, or technical things
- Developers should be able to see the visualization of the analysis of missed links between PRs and issues.
    - By developer
    - By time
    - By reason, e.g., a typo, missing, or technical things
- Developers should be able to see a historical graph visualization for links among commits, pull requests, and issues, where the links are constructed by an ML algorithm.
- Developers should be suggested a list of possible issues when clicking a commit button or a pull request creation button with an average of 95% recall.
- The accuracy of the automated traceability linking algorithm will be at least 60% accurate.
- The system should be able to support GitHub and AzureDevOps as a version control system and Jira, GitHub Issues, and Azure DevOps as a bug tracking system.

2.2. Non-Functional Requirements

2.2.1. Usability

- There should exist a navigational sidebar visible from all pages for allowing the user to easily navigate the app.

- The titles on the navigation bar, the titles on each page, and the labels on all buttons should be meaningful and self-explanatory so that users who do not read the user manual are able to understand how the website is used from the titles on the navigation bar, the titles on the screen and the labels on the buttons.
- All screens including pop-ups should be reached by being clicked at most twice.
- In order to be a user-friendly website, it is imperative to be consistent in the website layout and design by following a design pattern in which all screens accessed from the sidebar use the same main template.
- Except for the pop-ups, none of the screens on the website must be connected to each other, so users don't have to go backward on the website.
- Web Content Accessibility Guidelines [13] must be followed, which makes content in the website more accessible to users with disabilities.

2.2.2. Reliability

- Users must be able to access a historical graph visualization for links among commits, pull requests, and issues of their projects 90% of the time without failure.
- The data fetched from repositories must be used without changing.
- Users who close the browser without properly logging out of their account will be automatically logged out.

2.2.3. Performance

- The load time of each page must be less than 2 seconds.
- The average response time of each button must be about 2 seconds.
- The website must keep the above-mentioned times the same, up to 100 simultaneous users.

2.2.4. Supportability

- User feedback will be evaluated continuously and if there is any bug reported by users, it will be assigned to a developer within 24 hours.

2.2.5. Scalability

- When the daily traffic of this website, which has 1000 visitors per day, exceeds this number, the max bandwidth limit of this website that is allocated to the hosting plan should not be exceeded.

3. Competitive Analysis

There exist many articles inspecting the traceability and recovery of issue links from various perspectives [1], [2], [3]. In these studies, tools aiming to detect and recover concurrent and past issue links are proposed only for research purposes and a commercial tool with such features does not yet exist. As such, we did not include the proposals given in these studies in our competitive analysis. Moreover, certain GitHub Actions pipeline commands exist in GitHub marketplace, which enforces traceability by not allowing a pipeline to exist without all PR-commit pairs being linked [14], [15]. Since these are pre-existing tools only aimed toward Actions pipelines, we do not consider them in our analysis.

	ZenHub	**Boring Cyborg**	**Quantify**	**ReLink**
Manual Issue-PR Linking	✅	✅	✅	✅
Manual Issue-Commit Linking	✅	✅	✅	✅
Suggestion for possible (Issue-Commit and Issue-PR) links	❌	❌	❌	✅
Warning if link is missing	✅	✅	❌	✅
Enforcing linking at the commit and merge states	❌	❌	❌	✅
Detecting past missing links	❌	❌	❌	✅
Visualization of missing links	❌	❌	❌	✅
Visualization of links among commits, PRs, issues	❌	❌	❌	✅
Supported environments	GitHub	Jira GitHub	Jira GitHub	Jira GitHub Azure DevOps

4. Academic Analysis

In this section, a summary of the academic research conducted on this topic will be provided. Aung et al. explain that the recovery of lost artifact links is an important practice in change impact analysis (CIA) which is a method of change affect assessment during software evolution, and further explains several methods of link recovery using ML methods such as the usage of the random forest algorithm and recurrent neural networks (RNNs) [1]. Rath et al. further emphasize the importance of links between commits and issues for software system traceability, providing several methods for text similarity detection to be used with naive bayes, decision tree, and random forest classifiers. This paper also explains the importance of the model's recall, which measures the percentage of correctly detected true positives, is an important metric while recommending developers with possible issue IDs [2]. Lüders et al. investigate links between issues in a similar way, and introduce the BERT deep learning model which is trained on titles and descriptions of issues [3]. Even though we are currently not planning to include link recovery between issues in our project, the BERT model may prove useful during our implementation since it is used for a similar purpose. Finally, DeepLink is introduced by Ruan et al., which is an RNN implementation for recovering commit-issue links along with the usage of methods like word embedding, which is the most recognized existing academic implementation of such a tool, therefore being paramount for the development of our project [16].

References

[1] T. W. Aung, H. Huo, and Y. Sui, “A literature review of automatic traceability links recovery for software change impact analysis,” *Proceedings of the 28th International Conference on Program Comprehension*, 2020. [Online]. Available: https://doi.org/10.1145/3387904.3389251. [Accessed: 29-Sep-2022].

[2] M. Rath, J. Rendall, J. L. Guo, J. Cleland-Huang, and P. Mäder, “Traceability in the wild,” *Proceedings of the 40th International Conference on Software Engineering*, 2018. [Online]. Available: https://doi.org/10.1145/3180155.3180207. [Accessed: 29-Sep-2022].

[3] C. M. Lüders, T. Pietz, W. Maalej. “Automated Detection of Typed Links in Issue Trackers”. [Online]. Available: https://doi.org/10.48550/arXiv.2206.07182. [Accessed: 29-Sep-2022].

[4] “React – a JavaScript library for building user interfaces,” *React*. [Online]. Available: https://reactjs.org/. [Accessed: 08-Oct-2022].

[5] *Django*. [Online]. Available: https://www.djangoproject.com/. [Accessed: 08-Oct-2022].

[6] “The developer Data Platform,” *MongoDB*. [Online]. Available: https://www.mongodb.com/. [Accessed: 08-Oct-2022].

[7] I. MinIO, “High performance, kubernetes native object storage,” *MinIO*. [Online]. Available: https://min.io/. [Accessed: 08-Oct-2022].

[8] “Pytorch,” *PyTorch*. [Online]. Available: https://pytorch.org/. [Accessed: 08-Oct-2022].

[9] “Getting started with the built-in Bert Algorithm,” *Google*. [Online]. Available: https://cloud.google.com/ai-platform/training/docs/algorithms/bert-start. [Accessed: 08-Oct-2022].

[10] *Google Colab*. [Online]. Available: https://colab.research.google.com/. [Accessed: 08-Oct-2022].

[11] “ACM Code of Ethics and Professional Conduct,” *Code of Ethics*. [Online]. Available: https://www.acm.org/code-of-ethics. [Accessed: 08-Oct-2022].

[12] G. D. Law, “What is ‘Privacy by design’ (PBD)?,” *Medium*, 22-Sep-2021. [Online]. Available: https://medium.com/golden-data/what-is-privacy-by-design-pbd-9a3e4d96536a. [Accessed: 08-Oct-2022].

[13] “Web Content Accessibility Guidelines (WCAG) 2.1”. *W3schools*. [Online]. Available: https://www.w3.org/TR/WCAG21/. [Accessed: 29-Sep-2022].

[14] “Github Action Check Linked Issues,” *GitHub*. [Online]. Available: https://github.com/nearform/github-action-check-linked-issues. [Accessed: 08-Oct-2022].

[15] H. Shobokshi, “Verify Linked Issue Action,” *GitHub*. [Online]. Available: https://github.com/hattan/verify-linked-issue-action. [Accessed: 08-Oct-2022].

[16] H. Ruan, B. Chen, X. Peng, and W. Zhao, “DeepLink: Recovering issue-commit links based on Deep Learning,” *Journal of Systems and Software*, vol. 158, p. 110406, 2019. [Online]. Available: https://doi.org/10.1016/j.jss.2019.110406. [Accessed: 08-Oct-2022].