Title Comes Here

Kadir Ersoy, Ecenur Sezer, Suzan Üsküdarli, Fatma Başak Aydemir
Boğaziçi University
Istanbul, Türkiye
{kadir.ersoy, ecenur.sezer, suzan.uskudarli, basak.aydemir}@boun.edu.tr

***Abstract*—Requirements traceability refers to the capability of following the life of a requirement in both forwards and backward directions, and to the ability to link requirements to other software artifacts through particular relationships. These traceability links enable stakeholders to monitor the development progress of requirements and assist system engineers in tracking the effort and workload associated with fulfilling those requirements. Traditional methods to recover traceability links necessitate significant human effort, making them unsuitable and inefficient, particularly as the project scale grows. In this work, we present a tool that establishes automated requirement traceability links between requirements written in natural language and software artifacts acquired from GitHub repositories. The tool implements three optional methods to trace the artifacts to the requirements, namely a keyword extraction pipeline, TF-IDF vector and word-vector. The captured traces are stored in a graph database and visualized. Additionally, an interactive dashboard featuring statistical data about the artifacts and their traces is implemented. By offering automated traceability and comprehensive visualization, this tool aims to enhance the management of requirements and to understand their quality in software development projects.**

***Index Terms*—Requirement traceability, NLP, Keyword extraction, Traceability graph**

I. Introduction

II. Problem and Motivation

- Requirements traceability definition
- Why do we do it, examples
  - It is too hard to do it manually
    - Cost of integration: Need trace link recovery,
    - Human effort: Do not force extra measures for traceability
  - Its role in project management
    - Monitoring the progress: Status of the project
    - Effort assessment: How much effort went to which features
    - Detecting problems for specific features
- showcasing trace data and other stats to user in a self-evident manner

III. Related Work

IV. Method

- Inputs-outputs
- Workflow + diagrams
- Heuristics:
  - Keyword extraction, more detailed than poster
  - Vector-based methods, more detailed than poster
- Implementation:
-
  - Trace graph + visuals
  - Dashboard
  - (Using same example for trace finding and trace graph)

V. Evaluation Plan

- Ground truth
- Experiment setup
- Results
- Evaluation of results

VI. Conclusions

- Observations
- Threads to validity
- Future Work

References