Logical dependencies: applications in software clustering

1st Adelina Stana
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address

2nd Ioana Sora
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address

***Abstract*—This document is a model and instructions for LaTeX. This and the IEEEtran.cls file define the components of your paper [title, text, heads, etc.]. *CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

The software architecture helps developers in gaining a better understanding of the system and its expected behavior. Additionally, it is also of great help in change management. By knowing the existing system architecture, project managers can assess whether a requested change can be easily implemented or not.

Architecture reconstruction appears in contexts where a software system lacks documentation entirely, or where the documentation hasn't been updated to reflect changes in the system.

We intend to use connections obtained from the versioning system to enhance the results of clustering methods that previously relied solely on connections extracted from code.

II. Logical dependencies

About how LD are obtained; cite past papers

III. Related work

Talk about the results of others and then about what we intend to do.

IV. Method

Explain how the script works; insert diagram

A. Clustering algorithms: MST Clustering

Minimum Spanning Tree Clustering is a hierarchical clustering method based on the construction of a minimum spanning tree from the weighted graph formed by the data connections of the system [1].

B. Clustering algorithms: Louvain Clustering

Louvain Clustering is a community detection algorithm designed for finding clusters or communities in complex networks. The Louvain method involves a greedy algorithm that moves nodes between clusters to obtain clusters that are highly interconnected [2].

V. Results

TABLE I
Comparison of Louvain and MST clustering results for LD

Dataset	Entities Count	Cluster count		MQ metric	
		Louvain	MST	Louvain	MST
SD only	517	14	228	0.085	0.084
LD strength 10%	517	272	353	0.047	0.031
LD strength 20%	517	355	360	0.04	0.037
LD strength 30%	517	387	404	0.036	0.033
LD strength 40%	517	405	413	0.034	0.033
LD strength 50%	517	414	422	0.03	0.029
LD strength 60%	517	431	439	0.029	0.027
LD strength 70%	517	443	444	0.027	0.026
LD strength 80%	517	454	460	0.024	0.022
LD strength 90%	517	462	463	0.021	0.021
LD strength 100%	517	472	480	0.019	0.016

TABLE II
Comparison of Louvain and MST clustering results for LD combined with SD

Dataset	Entities Count	Cluster count		MQ metric	
		Louvain	MST	Louvain	MST
SD only	517	14	228	0.085	0.084
SD and LD strength 10%	517	15	74	0.087	0.054
SD and LD strength 20%	517	13	8	0.071	0.059
SD and LD strength 30%	517	13	14	0.071	0.083
SD and LD strength 40%	517	13	8	0.071	0.109
SD and LD strength 50%	517	13	8	0.071	0.109
SD and LD strength 60%	517	13	8	0.071	0.109
SD and LD strength 70%	517	13	10	0.071	0.155
SD and LD strength 80%	517	13	6	0.071	0.177
SD and LD strength 90%	517	13	2	0.071	0.129
SD and LD strength 100%	517	13	8	0.072	0.166

VI. Discussion

Based on the results from table II, we can observe that the combined approach of structural dependencies and logical dependencies gives a Modularity Quality (MQ) metric of 0.071, which is an improvement over the 0.085 MQ metric obtained when considering only structural dependencies.

Next, we will examine the clustering solutions obtained only from structural dependencies, in comparison to the clustering

solution derived from incorporating both structural and logical dependencies, filtered with a threshold of 20% for strength.

The entities listed below are placed in different clusters when analyzed based on structural dependencies alone, compared with the analysis using both logical and structural dependencies: taskdefs.Available$FileDir, taskdefs.Concat, taskdefs.Concat$1, taskdefs.Concat$MultiReader, taskdefs.Concat$TextElement, taskdefs.Javadoc$AccessType, util.WeakishReference, util.WeakishReference$HardReference.

The placement of the Concat class and its inner classes (Concat$1, Concat$MultiReader, Concat$TextElement) in a different cluster might be based on it's usage and purpose; according to the documentation : "This class contains the 'concat' task, used to concatenate a series of files into a single stream." [3].

REFERENCES

[1] M. Yu, A. Hillebrand, P. Tewarie, J. Meier, B. Dijk, P. Mieghem, and C. Stam, "Hierarchical clustering in minimum spanning trees," *Chaos (Woodbury, N.Y.)*, vol. 25, p. 023107, 02 2015.

[2] S. Harenberg, G. Bello, L. Gjeltema, S. Ranshous, J. Harlalka, R. Seay, K. Padmanabhan, and N. Samatova, "Community detection in large-scale networks: A survey and empirical evaluation," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 6, 11 2014.

[3] Apache Ant Project, "Apache Ant Concat Task Documentation," https://ant.apache.org/manual/api/org/apache/tools/ant/taskdefs/Concat.html, accessed on February 14, 2024.