Centrality Change Proneness: an Early Indicator of Microservice Architectural Degradation [Online Appendix]

Anonymous

No Institute Given

Abstract. This is the Online Appendix for the paper "Centrality Change Proneness: an Early Indicator of Microservice Architectural Degradation", containing the mathematical background that was excluded from the main manuscript due to space constraints.

Keywords: Microservices · Centrality · Temporal networks · Architectural Smells

1 Background

In this section, we present the background notions necessary for our work.

1.1 Architectural Reconstruction

In this work, we aim to obtain the architecture of an MSS for several releases. For this reason, we leverage static analysis, which refers to the analysis of the source code (in certain cases - bytecode [6]) of a system to gain insight into the system's design and architecture, code quality, and presence of patterns/antipatterns/smells [2]. Bakhtin et al. [3] performed a Systematic Mapping Study of static analysis tools aiming in particular at reconstructing microservice architecture. Together with additional authors, they currently attempt to run the tools and consider their effectiveness in achieving this task [9]. Our study used the *Code2DFD* tool by Schneider et al. [10], which is identified in [3].

1.2 Temporal Networks

Most networks currently considered in Software Engineering are *static* networks [1], i.e., networks consisting of a set of nodes linked together, but no temporal evolution is considered. Conversely, TNs are most commonly networks that maintain a static set of nodes, but edges contain some kind of temporal information [4]. The simplest model is the *snapshot* network, where we consider edges at discrete time instances $t = 1, 2, 3, ...$ over a static set of nodes. We can apply TNs to the problem of MAD as follows: each timestamp is a particular version

of the system (release or a combination of co-existing releases of separate microservices), nodes are the microservices themselves, and edges are the possible reconstructed calls between the services.

Centrality algorithms assign a metric to each node in the network that measures its overall importance in the network. To consider the centrality in the temporal perspective, Taylor et al. [11] (from now on - Taylor algorithm) proposed to extend the eigenvector centrality measure of a static network into the temporal domain by constructing a *supra-centrality matrix* of a temporal network.

The supra-centrality matrix block matrix is constructed as follows: (1) for each snapshot of the network, its adjacency matrix is put on the diagonal block of the supra-centrality matrix; (2) identical nodes at different snapshots are connected by placing inter-layer coupling matrices to the off-diagonal blocks of the supra-centrality matrix. The adjacency matrices can be converted to centrality matrices by applying some kind of centrality transformation, such as Hubs or PageRank, but can also be kept as original adjacency matrices.

Different strategies for constructing inter-layer coupling matrices are possible [7]. Taylor et al. [11] placed identity matrices next to the diagonal of the supra-centrality matrix to connect each node to itself in the previous and following snapshot, while Liu et al. [8] (Liu), Yin et al. [12] (Yin) and Huang et al. [5] (Huang) proposed different strategies - Liu and Yin apply different similarity measures to adjacency matrices of consecutive snapshots to compute the inter-layer coupling while Huang fit a times-series ARMA model to the degree sequence of each node to compute the coupling of a given snapshot to all the previous ones. Moreover, Liu converts the adjacency matrices to centrality matrices by weighting each pair of nodes by the sum of the two nodes' degrees.

After the supra-centrality matrix is constructed, the temporal eigenvector centrality is obtained by computing the leading eigenvector of the matrix. The entries of this eigenvector are the joint centralities (JC) of each node in the network at each snapshot. Furthermore, additional metrics can be obtained: Marginal Node Centrality (MNC), which is the sum of each node's JC across snapshots, and Marginal Layer Centrality (MLC), which is the sum of all nodes' JC in each snapshot. Given the JC and MLC of the network, conditional centralities (CC) are defined as the Joint centralities in each snapshot divided by the corresponding MLC.

Moreover, Taylor solved analytically for the zero- and first-order power expansion of the eigenvector equation used to compute the JC. The nodes' metrics given by the zero- and first-order expansions are called Time-Averaged Centrality (TAC) and First-Order Mover score (FOM), respectively. They are not computed by snapshot but are only given once for each node for the entire TN. The TAC metric represents the centrality of the nodes without taking into account the temporal evolution, while FOM describes the possibility of a node changing its JC metric during the evolution of the TN.

Since the supra-centrality matrix can contain potentially unbounded values, the TAC and FOM scores are also unbounded, and only the relative values

of the nodes matter. For this reason, we propose to normalize the FOM score quadratically to force the values onto the $[0, 1]$ interval. For each FOM score m_i, the normalized FOM score m'_i is computed as $m'_i = \frac{m_i}{\sqrt{\sum_i m_i^2}}$.

We propose the **Centrality Change Proneness (CCP)** rank, which is obtained from the FOM score by assigning to each quartile of the FOM score a value on the nominal scale `LOW`, `MEDIUM-LOW`, `MEDIUM-HIGH`, `HIGH`. In this work, we aim to assess the applicability of the proposed CCP rank to trends in MAD.

References

1. Abufouda, M., Abukwaik, H.: On using network science in mining developers collaboration in software engineering: A systematic literature review. International Journal of Data Mining & Knowledge Management Process (IJDKP) **7**(5/6), 17–34 (November 2017)
2. Bakhtin, A., Al Maruf, A., Cerny, T., Taibi, D.: Survey on tools and techniques detecting microservice api patterns. In: 2022 IEEE International Conference on Services Computing (SCC). pp. 31–38. IEEE (2022)
3. Bakhtin, A., Li, X., Soldani, J., Brogi, A., Cerny, T., Taibi, D.: Tools reconstructing microservice architecture: A systematic mapping study. Agility with Microservices Programming, co-located with ECSA **2023** (2023)
4. Holme, P., Saramäki, J.: Temporal network theory, vol. 2. Springer (2019)
5. Huang, Q., Zhao, C., Zhang, X., Wang, X., Yi, D.: Centrality measures in temporal networks with time series analysis. EPL (Europhysics Letters) **118**(3), 36001 (may 2017)
6. Hutcheson, R., Blanchard, A., Lambaria, N., Hale, J., Kozak, D., Abdelfattah, A.S., Cerny, T.: Software architecture reconstruction for microservice systems using static analysis via graalvm native image. In: 2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER). pp. 12–22 (2024)
7. Kumar, T., Narayanan, M., Ravindran, B.: Effect of inter-layer coupling on multilayer network centrality measures. Journal of the Indian Institute of Science **99**(2), 237–246 (jun 2019)
8. Liu, R., Zhang, S., Zhang, D., Zhang, X., Bao, X.: Node importance identification for temporal networks based on optimized supra-adjacency matrix. Entropy **24**(10), 1391 (sep 2022)
9. Schneider, S., Bakhtin, A., Li, X., Soldani, J., Brogi, A., Cerny, T., Scandariato, R., Taibi, D.: Comparison of static analysis architecture recovery tools for microservice applications: Preprint. arXiv preprint arXiv:2412.08352 (2024)
10. Schneider, S., Scandariato, R.: Automatic extraction of security-rich dataflow diagrams for microservice applications written in java. Journal of Systems and Software **202**, 111722 (2023)
11. Taylor, D., Myers, S.A., Clauset, A., Porter, M.A., Mucha, P.J.: Eigenvector-based centrality measures for temporal networks. Multiscale Modeling & Simulation **15**(1), 537–574 (2017)
12. Yin, R.R., Guo, Q., Yang, J.N., Liu, J.G.: Inter-layer similarity-based eigenvector centrality measures for temporal networks. Physica A: Statistical Mechanics and its Applications **512**, 165–173 (2018)