MEASURING RESILIENCE OF SOCIAL ORGANIZATIONS USING ENTROPY—A TEMPORAL CASE-STUDY FROM OPEN SOURCE SOFTWARE DEVELOPMENT

Christian Zingg, *Giona Casiraghi, Frank Schweitzer*

SIAM Workshop on Network Science 2020
July 9–10 · Toronto

Summary

Members of social organizations such as communities of software developers can mitigate challenging situations by interacting with their peers. We introduce a measure to track the extent to which the interactions between the members are constrained. In a case study on different Open Source developer communities, our measure reproduces shocks such as the sudden leave of a core-developer.

Teams of Open Source Software Developers

Social organizations are ubiquitous in our everyday life, ranging from the project team we are working in to the special interest group we are contributing to, online. For our study, we focus on communities of software developers from Open Source Software projects such as the `Gentoo Linux` operating system, or the `igraph` programming library. The developers in such a community collaborate with the goal to develop and maintain their particular Open Source Software.

In order to survive, these communities have to mitigate issues such as software malfunctions, or losses of key-developers. To solve these issues, the developers can interact with their peers, for example, to distribute tasks, to discuss possible solutions, to write code together, etc. We argue that the more ways there are in which different developers can interact, the broader is the repertoire of possible solutions the community can come up with.

However, how the developers interact, is subject to constraints. For example, if two developers have to interact often to develop a particular software component together, they have less time available to interact with other peers. Our aim is to develop a method to assess how strongly the interactions among all developers in a community are restricted by such constraints.

Measuring Diversity in Interaction Preferences

The key element of our method is a measure of the **potentiality** of the organization [4]. This expresses the ability to recover from any kind of perturbation, or shock, that the organization experiences (individuals leaving, interactions interrupted, etc.). To calculate the potentiality, we use a representation of the organization as an **ensemble of networks**. In these networks, the developers correspond to nodes, and their interactions to edges. Suppose that at time t we observe a network g_t. In principle, we can treat g_t as a random realization, sampled from the set of all possible configurations of the organization that could have been observed under the constraints given. The lower the number of configurations, or the more similar the possible configurations, the lower the potentiality of the organization, because fewer options are available for the system to operate. The higher the number of possible configurations and the more diverse they are, the higher the potential of the organization to recover from a shock or to mitigate an issue, as more and more alternative ways of interacting are available for its members.

By appropriately fitting a generalised hypergeometric ensemble of random graphs (gHypEG) [1] to the observed interactions, we study the probability distribution of the possible configurations of the system. We thus estimate potentiality in terms of the probability distribution underlying the ensemble. A system has high potentiality if the distribution is broad and possible configurations are heterogeneous, and low potentiality if the distribution is narrow and possible configurations tend to be homogeneous. By computing the **Shannon entropy** of the gHypEG, we quantify precisely this property. Thus, the *potentiality of a social organization* is the Shannon entropy of the network ensemble characterizing the current state of the system.

From Interaction Data to Networks

Open Source developers leave traces of their interactions online, in issue trackers, in code repositories, or also in mailing lists. Here we focus on two sources of interactions, depending on which is available for a particular community.

The first source of interactions are code-repositories,

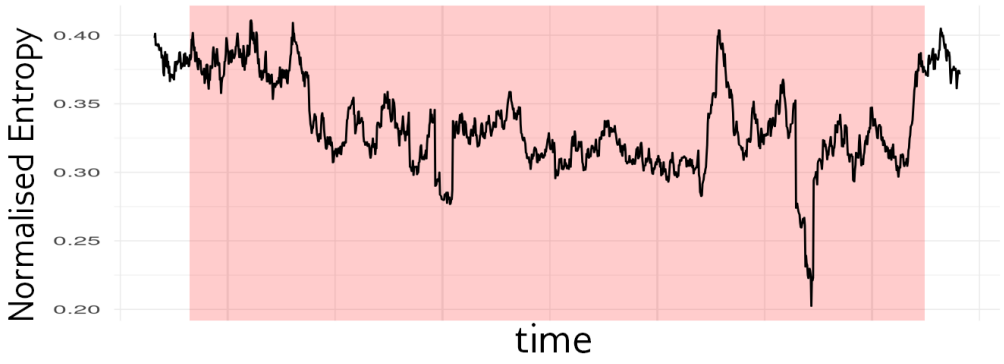

Figure 1: Evolution of potentiality in the `Gentoo Linux` developer community.

from which code-editing interactions can be extracted with the Python library `git2net` [2]. Such a code-editing interaction occurs when one developer changes existing code written by another developer. These interactions indicate that the editing developer gains knowledge about a functionality introduced by the edited developer. Now, to mitigate issues arising in this code, also the editing developer has knowledge to collaborate on possible solutions.

The second source of interactions are issue-trackers, which are online forums where the developers discuss how to fix a particular bug. This type of interaction reflects awareness of the other developers and their ideas on how to fix bugs. Hence, to fix a new bug, a developer interacting with many peers has access to a broader selection of feedback.

From either source, we can construct a network based on the interactions occurring in a specific time-interval. By sliding this time-interval through the observation period of the respective community, we can construct a time-series of networks to characterize the dynamics of the interactions between the developers. From this time-series of networks we compute a corresponding time-series of potentiality of the organization.

Case Studies

In a case study, we compute the changes in potentiality over time for 5 different Open Source Software communities: `Gentoo Linux`, `igraph`, `FFmpeg`, `libav`, and the `Linux Kernel`. For this extended abstract we focus on the `Gentoo Linux` software developer community and interaction data from its issue-tracker. By interviewing `Gentoo` developers, the authors of [3] found that this community exhibited a threatening evolution between the years 2004 and 2008. They found that in this period a particular developer in the organization, *Alice*, became the main responsible developer for distributing incoming tasks to the rest of the community. I.e., the other developers developed a tendency to receive tasks from *Alice* instead of acting on their own. Then, in March 2008, *Alice* left the community abruptly after an internal dispute, and the other developers had to handle the task distribution again among themselves.

Potentiality identifies this trend: It decreases between the years 2004 and 2008, when *Alice* distributed a large part of the tasks. Then, in 2008, it increases almost instantaneously again, when *Alice* left. The evolution of potentiality is displayed in Figure 1, with a red background highlighting the period when Alice was a member of the community.

References

[1] G. Casiraghi and V. Nanumyan. Generalised hypergeometric ensembles of random graphs: The configuration model as an urn problem. *arXiv:1810.06495*, 2018.

[2] C. Gote, I. Scholtes, and F. Schweitzer. Git2net: Mining time-stamped co-editing networks from large git repositories. In *Proceedings of the 16th International Conference on Mining Software Repositories*, MSR '19, page 433–444. IEEE Press, 2019.

[3] M. S. Zanetti, I. Scholtes, C. J. Tessone, and F. Schweitzer. The rise and fall of a central contributor: Dynamics of social organization and performance in the gentoo community. In *CHASE/ICSE '13 Proceedings of the 6th International Workshop on Cooperative and Human Aspects of Software Engineering*, pages 49–56, 2013.

[4] C. Zingg, G. Casiraghi, G. Vaccario, and F. Schweitzer. What is the entropy of a social organization? *Entropy*, 901(21), May 2019.