Advances in Complex Systems
Vol. 17, Nos. 7 & 8 (2014) 1430001 (9 pages)

DOI: 10.1142/S0219525914300011

A COMPLEX NETWORKS PERSPECTIVE ON COLLABORATIVE SOFTWARE ENGINEERING

MARCELO CATALDO
EMC Corporation,
New York City, NY, USA
mcataldo@alumni.cmu.edu

INGO SCHOLTES
ETH Zurich, Chair of Systems Design
Weinbergstrasse 56/58, CH-8092 Zurich, Switzerland
ischoltes@ethz.ch

GIUSEPPE VALETTO
Fondazione Bruno Kessler,
Via Sommarive, 18, Trento, Italy
valetto@fbk.eu

Received 8 April 2015
Accepted 9 April 2015
Published 7 May 2015

Large collaborative software engineering projects are interesting examples for evolving complex systems. The complexity of these systems unfolds both in evolving software structures, as well as in the social dynamics and organization of development teams. Due to the adoption of Open Source practices and the increasing use of online support infrastructures, large-scale data sets covering both the social and technical dimension of collaborative software engineering processes are increasingly becoming available. In the analysis of these data, a growing number of studies employ a *network perspective*, using methods and abstractions from network science to generate insights about software engineering processes. Featuring a collection of inspiring works in this area, with this topical issue, we intend to give an overview of state-of-the-art research. We hope that this collection of articles will stimulate downstream applications of network-based data mining techniques in empirical software engineering.

1. Introduction

Large collaborative software engineering projects are interesting examples for evolving complex systems. The complexity of these systems unfolds in the complex code structures being developed, but also in the complex social structures emerging in teams of collaborating developers. Through the adoption of Open Source Software (OSS) practices and the widespread use of online support infrastructures and social coding platforms, the complex nature of software development can increasingly be

studied based on massive data sets. This has not only resulted in a surge of data-driven, quantitative studies in the field of *empirical software engineering*; it has also generated a huge interest in mining the wealth of *relational data* that can be extracted from those data sets on collaborative software engineering, and study it from a *complex networks* or *network science* perspective.

The topical issue at hand is devoted to such works, which address the complex technical, social and socio-technical aspects of team-based software development. It provides a thought-provoking overview of state-of-the-art research taking a network perspective to address problems in empirical software engineering.

In this editorial, we take the opportunity to briefly summarize some of the most significant challenges and opportunities of applying network science methods to data from software engineering processes, which are well-represented in the selection of works collected in this topical issue. Nevertheless, we can at most cover selected topics and examples that set the stage for the works contained in this topical issue. Therefore, our selection should not be mistaken for an exhaustive review of the much larger body of existing works in this area.

2. The Technical Dimension of Software Engineering

In the context of software engineering, methods from network science can first be applied to *technical aspects* of software projects. A particularly common approach is based on the extraction of data on the structure and evolution of software artifacts which are created by developers. This can be achieved by mining the development repositories in particular code and configuration management repositories such as SVN, Git etc.

Here, a network perspective can be applied to study evolving dependency structures that interconnect modular units of source code, such as methods, classes, packages or libraries. Such a perspective facilitates both the development of theoretical models of software evolution, as well as empirical analyses of software projects: Theoretical models for the growth dynamics of such dependency networks can for instance inform us about underlying growth mechanisms, the formation of network motifs, or sustainable regimes in the evolution of software [23, 24, 21]. A network perspective on dependency networks can further be used to evaluate software modularity, for instance to formulate models for the propagation of code changes. Such models can then help us to better understand which dependencies play a crucial role in the evolution of software [12].

Apart from such modeling approaches, numerous examples of network-based empirical analyses of software structures exist. A number of statistical analyses of dependency networks defined both at the class- and package-level of object-oriented source code have highlighted similarities as well as differences to complex networks emerging in other domains [14, 15, 13, 17]. Community structures in class dependency networks were studied for instance [22], highlighting their importance for the design of modular software structures. A similar approach was used [29]

to quantitatively assess the *congruence* between package structures designed by software engineers for the organization of code and the natural cluster structures emerging in dependency networks. The resulting network-analytic measure not only provides insights into the evolution of software projects, it also can be used to assist project managers and developers in refactoring efforts [32].

Besides such statistical analyses at the macroscopic level, a microscopic analysis of individual nodes can provide us with further insights about software. Applying centrality measures to package dependency networks can for instance help us to decide which OSS packages to use in a software project [13]. It has also been shown that node-level measures applied to class dependency networks can be used for the automated prediction of software defects [20, 1]. Similarly, a network perspective on dependencies between requirements has recently been proposed to predict integration errors in software projects [28].

Reflecting the broad set of activities in the field, this topical issue features several works which study the technical dimension of software engineering projects from a network perspective. In their article *Recode: Software Package Refactoring via Community Detection in Bipartite Software Networks*, Weifeng Pan and coauthors study dependency networks of software artifacts. They show how a community detection algorithm can be used to identify refactoring candidates that optimize the package structure of software projects. A statistical analysis of software dependency networks is also presented in the article *Node Mixing and Group Structure of Complex Software Networks* contributed by Lovro Šubelj and coauthors. The authors study clustering structures as well as correlations between the degrees of neighbouring nodes, showing that dependency networks differ significantly from complex networks found in other contexts. The important question how a network perspective can help us to identify the most important pieces of source code is addressed by Phil Meyer and coauthors in their article *Identifying Important Classes of an Evolving Software System Through K-core Decomposition*: their results from an analysis of three Java projects indicate that indeed network-analytic methods can be used to identify core classes.

3. The Social Dimension of Software Engineering

The works outlined above demonstrate that a network perspective on software artifacts can provide interesting insights into the output of collaborative software engineering processes and its characteristics. However, *social aspects* emerging in the communication, collaboration or coordination between developers and/or users are an important additional source of complexity in software projects. How do communication and coordination structures in development teams influence development productivity or code quality? And how do collaboration structures in OSS communities affect their resilience?

Again, such questions can be studied based on network representations of dyadic relations inferred from recorded interactions between developers or users. Data

on evolving collaboration structures of large Open Source Software communities have been studied in a number of works. Networks of OSS developers, which were assumed to be connected whenever they have been active in the same project, were studied [19]. The authors [19] again find that the resulting networks share statistical similarities with a number of social networks found in other contexts. Other researchers [9] studied 120 OSS projects on SourceForge, highlighting a significant variation of centralization across communities which indicates differences in their social organization.

Combining data from developer weblogs, mailing list archives and an online social network platform targeted at developers, the social network structure of OSS developers was studied [27]. The analysis of the resulting networks was used to calibrate an agent-based model for OSS projects, aiming at replicating how developers chose projects. E-mail communication [2] was used as the basis to construct the communication networks of OSS community members, again highlighting statistical similarities with the interaction networks found in other types of online communities. Again using data on E-mail communication [26, 25], models for the growth of social structures in OSS communities were studied. The models combine local and non-local network formation rules, thus highlighting a balance between hierarchical and distributed collective social mechanisms in OSS communities.

The authors [16] used commit logs to construct collaboration networks based on the co-editing of files in a number of OSS projects. Established measures from social network analyses were then applied to categorize OSS projects, and study the evolution of their collaboration structures. Similar macroscopic, network-analytic measures were used to investigate the evolving social organization of OSS communities [29]. The results highlight different organizational regimes which affect the performance and resilience of communities [31]. A microscopic analysis of the position of community members in collaboration networks was used to predict which bug reports will eventually be fixed, thus pointing at applications of social network analysis in the design of online support infrastructures [30].

Traditionally, works studying the social dimension of software engineering processes have focused on the important role played by the network structure of collaborations, communication or task allocation. Extending this notion, more recent works have started to additionally study the *content* of exchanges made within a particular network structure [11]. The results indicate that a combination of network-based methods with a study of content of communication exchanges allows us to better understand the performance of software development teams.

Two of the articles in this topical issue specifically address the social dimension of software engineering processes. In their article *Communication in Innovation Communities: An Analysis of 100 Open Source Projects*, Markus Geipel and collaborators take a network perspective on communication flows between users and developers in Open Source Software communities. Using a large-scale data set on 100 OSS projects, they find that users dominate the communication in the associated communities. Considering this important role of users, an interesting further

question is which of these users are likely to become involved in development tasks. This question is addressed by Qi Xuan and collaborators. In their article *Ranking Developer Candidates by Social Links* they study communication networks of OSS projects, applying different methods to predict which of the users eventually become members of the development team. The results suggest that well-known network-based ranking schemes can be used to identify developer candidates, thus highlighting that communication networks carry significant amounts of information about the motivation and skills of community members in OSS projects.

4. Socio-Technical Studies of Software Engineering

So far we have covered works that address either the technical or the social dimension in isolation. However, software projects are *socio-technical systems*. The combination of the two dimensions is thus a further source of complexity, and carries significant information. After all, it is a team of developers which shapes the architecture of a software. And similarly, this architecture affects which developers have to coordinate their work, thus shaping the organizational structures of the development team. The resulting intuition that social structures and software architectures co-evolve can be traced back more than 50 years to Melvin Edward Conway, thus often being paraphrased as "Conway's law". In more general terms, the related "mirroring hypothesis" states that the governance structures of an organization directly affect the modular structures of the products that they develop.

The availability of fine-grained data on both social interactions and software structures has recently allowed researchers to study this phenomenon from a quantitative perspective. Again the network perspective has proven to be valuable in this context. The authors [18] use a network-based approach to test the mirroring hypothesis both in commercial and Open Source software development. They find strong evidence for the fact that the modular organization of software structures is coupled with the structure of the social organizations by which they were developed. A number of works have thus utilized *two-mode networks* capturing both software dependencies and collaboration structures at the same time. Such an approach was used [8] to study *socio-technical congruence*, referring to the level of congruence between software dependencies and coordination patterns. The authors find that the degree of socio-technical congruence affects both software development performance and software quality [8, 7]. In particular, high levels of socio-technical congruence were found to significantly reduce the resolution time of modification requests as well as the rate of software failures.

Both the dependency network as well as a network capturing the assignment of tasks to developers have been studied [3], showing that a socio-technical perspective can help to predict software defects with higher accuracy. A similar socio-technical network perspective was used [10] to analyze and visualize the different organizational patterns in projects. Apart from serving as interesting empirical studies, a

socio-technical perspective on software engineering can be fruitfully applied to facilitate coordination in development teams. Building on the idea of socio-technical congruence, recent works have thus studied how fine-grained data on dependency structures and the association between developers and software constructs can be used to identify coordination needs in real-time [5, 4, 6].

Representing a particularly challenging problem, we are happy that two works in this topical issue have addressed socio-technical aspects in collaborative software engineering. In their article *Modeling Distributed Collaboration on GitHub*, Nora McDonald and coauthors study data on five OSS projects from GitHub. They particularly address the question how the use of collaboration mechanisms offered by this popular online collaboration platform affects both the social organization and the success of software projects. As such, their work addresses a socio-technical dimension, highlighting how design decisions in the development of collaboration platforms influence the emerging social structures in software projects. A large-scale data set featuring 360,000 OSS projects hosted on SourceForge is analyzed by Frank Schweitzer and coauthors in their work *How do OSS projects change in number and size? A large-scale analysis to test a model of project growth*. The authors show that the growth rate of collaborative projects can be modeled by an established statistical model of firm growth, which balances two antagonistic forces of developers joining existing projects versus founding new projects. As such, this work highlights the potential of interdisciplinary research in the modeling of socio-technical aspects of software engineering.

5. Conclusion

The works in this topical issue impressively demonstrate the various ways in which network-based methods can be utilized to study research questions relevant to empirical software engineering. However, with more and more massive data sets from social coding platforms like SourceForge or GitHub becoming available, they can necessarily only mark the beginning of a fruitful field of research. Much remains to be discovered and we hope that this topical issue stimulates further studies which provide all of us with interesting insights into the technical and social dimension of complex software engineering processes.

Acknowledgements

The guest editors would like to express their gratitude to all authors as well as to the anonymous reviewers. Without your much appreciated contributions, this topical issue would not have been possible.

References

[1] Bhattacharya, P., Iliofotou, M., Neamtiu, I., and Faloutsos, M., Graph-based analysis and prediction for software evolution, in *Proceedings of the 34th International Conference on Software Engineering*, ICSE '12 (IEEE Press, Piscataway,

NJ, USA, 2012), ISBN 978-1-4673-1067-3, pp. 419–429, http://dl.acm.org/citation.cfm?id=2337223.2337273.

[2] Bird, C., Gourley, A., Devanbu, P., Gertz, M., and Swaminathan, A., Mining email social networks, in *Proceedings of the 2006 International Workshop on Mining Software Repositories*, MSR '06 (ACM, New York, NY, USA, 2006), ISBN 1-59593-397-2, pp. 137–143, doi:10.1145/1137983.1138016, http://doi.acm.org/10.1145/1137983.1138016.

[3] Bird, C., Nagappan, N., Gall, H., Murphy, B., and Devanbu, P., Putting it all together: using socio-technical networks to predict failures, in *Software Reliability Engineering, 2009. ISSRE '09. 20th International Symposium on* (2009), ISSN 1071-9458, pp. 109–119, doi:10.1109/ISSRE.2009.17.

[4] Blincoe, K., Valetto, G., and Damian, D., Do all task dependencies require coordination? the role of task properties in identifying critical coordination needs in software projects, in *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2013 (ACM, New York, NY, USA, 2013), ISBN 978-1-4503-2237-9, pp. 213–223, doi:10.1145/2491411.2491440, http://doi.acm.org/10.1145/2491411.2491440.

[5] Blincoe, K., Valetto, G., and Goggins, S., Proximity: a measure to quantify the need for developers' coordination, in *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, CSCW '12 (ACM, New York, NY, USA, 2012), ISBN 978-1-4503-1086-4, pp. 1351–1360, doi:10.1145/ 2145204.2145406, http://doi.acm.org/10.1145/2145204.2145406.

[6] Blincoe, K. C., *Timely and Efficient Facilitation of Coordination of Software Developers' Activities*, Ph.D. thesis, Drexel University, Philadelphia, PA, USA (2014), aAI3613734.

[7] Cataldo, M. and Herbsleb, J., Coordination breakdowns and their impact on development productivity and software failures, *Software Engineering, IEEE Transactions on* **39** (2013) 343–360.

[8] Cataldo, M., Herbsleb, J. D., and Carley, K. M., Socio-technical congruence: a framework for assessing the impact of technical and work dependencies on software development productivity, in *Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM '08 (ACM, New York, NY, USA, 2008), ISBN 978-1-59593-971-5, pp. 2–11, doi:10.1145/1414004.1414008, http://doi.acm.org/10.1145/1414004.1414008.

[9] Crowston, K. and Howison, J., The social structure of free and open source software development, *First Monday* **10** (2005).

[10] De Souza, C., Froehlich, J., and Dourish, P., Seeking the source: software source code as a social and technical artifact, in *Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work* (ACM, 2005), pp. 197–206.

[11] Ehrlich, K. and Cataldo, M., The communication patterns of technical leaders: impact on product development team performance, in *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing* (ACM, 2014), pp. 733–744.

[12] Geipel, M. M. and Schweitzer, F., Software change dynamics: evidence from 35 Java projects, in *Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering* (ACM, 2009), pp. 269–272.

[13] Kohring, G., Complex dependencies in large software systems, *Advances in Complex Systems* **12** (2009) 565–581.

[14] LaBelle, N. and Wallingford, E., Inter-Package Dependency Networks in Open-Source Software, *eprint arXiv:cs/0411096* (2004).

[15] Li, D., Han, Y., and Hu, J., Complex network thinking in software engineering, in *2008 International Conference on Computer Science and Software Engineering*, Vol. 1 (2008), pp. 264–268, doi:10.1109/CSSE.2008.689.

[16] Lopez-Fernandez, L., Robles, G., Gonzalez-Barahona, J. M., *et al.*, Applying social network analysis to the information in CVS repositories, in *International Workshop on Mining Software Repositories* (IET, 2004), pp. 101–105.

[17] Louridas, P., Spinellis, D., and Vlachos, V., Power laws in software, *ACM Trans. Softw. Eng. Methodol.* **18** (2008) 2:1–2:26.

[18] MacCormack, A., Baldwin, C., and Rusnak, J., Exploring the duality between product and organizational architectures: A test of the mirroring hypothesis, *Research Policy* **41** (2012) 1309–1324.

[19] Madey, G., Freeh, V., and Tynan, R., The open source software development phenomenon: An analysis based on social network theory, *AMCIS 2002 Proceedings* (2002) 247.

[20] Nguyen, T., Adams, B., and Hassan, A., Studying the impact of dependency network measures on software quality, in *Software Maintenance (ICSM), 2010 IEEE International Conference on* (2010), ISSN 1063-6773, pp. 1–10, doi:10.1109/ICSM.2010.5609560.

[21] Tessone, C. J., Geipel, M. M., and Schweitzer, F., Sustainable growth in complex networks, *EPL (Europhys. Lett.)* **96** (2011) 58005.

[22] ubelj, L. and Bajec, M., Community structure of complex software systems: Analysis and applications, *Physica A: Statistical Mechanics and its Applications* **390** (2011) 2968–2975.

[23] Valverde, S. and Sol, R. V., Logarithmic growth dynamics in software networks, *EPL (Europhys. Lett.)* **72** (2005) 858.

[24] Valverde, S. and Solé, R. V., Network motifs in computational graphs: A case study in software architecture, *Phys. Rev. E* **72** (2005) 026107.

[25] Valverde, S. and Solé, R. V., Self-organization versus hierarchy in open-source social networks, *Physical Review E* **76** (2007) 046118.

[26] Valverde, S., Theraulaz, G., Gautrais, J., Fourcassié, V., and Solé, R. V., Self-organization patterns in wasp and open source communities, *Intelligent Systems, IEEE* **21** (2006) 36–40.

[27] Wagstrom, P., Herbsleb, J., and Carley, K., A social network approach to free/open source software simulation, in *Proceedings First International Conference on Open Source Systems* (2005), pp. 16–23.

[28] Wang, J., Li, J., Wang, Q., Yang, D., Zhang, H., and Li, M., Can requirements dependency network be used as early indicator of software integration bugs?, in *Requirements Engineering Conference (RE), 2013 21st IEEE International* (2013), pp. 185–194, doi:10.1109/RE.2013.6636718.

[29] Zanetti, M. S., Sarigöl, E., Scholtes, I., Tessone, C. J., and Schweitzer, F., A quantitative study of social organisation in open source software communities, in *2012 Imperial College Computing Student Workshop, ICCSW 2012, September 27–28, 2012, London, United Kingdom* (2012), pp. 116–122, doi:10.4230/OASIcs.ICCSW.2012.116, http://dx.doi.org/10.4230/OASIcs.ICCSW.2012.116.

[30] Zanetti, M. S., Scholtes, I., Tessone, C. J., and Schweitzer, F., Categorizing bugs with social networks: a case study on four open source software communities, in *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13

(IEEE Press, Piscataway, NJ, USA, 2013), ISBN 978-1-4673-3076-3, pp. 1032–1041, http://dl.acm.org/citation.cfm?id=2486788.2486930.

[31] Zanetti, M. S., Scholtes, I., Tessone, C. J., and Schweitzer, F., The rise and fall of a central contributor: dynamics of social organization and performance in the gentoo community, in *Cooperative and Human Aspects of Software Engineering (CHASE), 2013 6th International Workshop on* (IEEE, 2013), pp. 49–56.

[32] Zanetti, M. S., Tessone, C. J., Scholtes, I., and Schweitzer, F., Automated software remodularization based on move refactoring: a complex systems approach, in *Proceedings of the 13th International Conference on Modularity*, Modularity '14 (ACM, New York, NY, USA, 2014), ISBN 978-1-4503-2772-5, pp. 73–84, doi:10.1145/2577080.2577097, http://doi.acm.org/10.1145/2577080.2577097.