Service Networks for Development Communities

Damian A. Tamburri
Department of Computer Science,
VU University Amsterdam,
The Netherlands
d.a.tamburri@vu.nl

Patricia Lago
Department of Computer Science,
VU University Amsterdam,
The Netherlands
p.lago@vu.nl

Hans van Vliet
Department of Computer Science,
VU University Amsterdam,
The Netherlands
j.c.van.vliet@vu.nl

***Abstract*—Communities of developers have rapidly become global, encompassing multiple timezones and cultures alike. In previous work we investigated the possible shapes of communities for software development. In addition, we explored mechanisms to uncover communities emerging during development. However, we barely scratched the surface. We found that development communities yield properties of dynamic change and organic evolution. Much work is still needed to support such communities with mechanisms able to proactively react to community dynamism. We argue that service-networks can be used to deliver this support. Service-networks are sets of people and information brought together by the internet. This paper is a first attempt at studying this research area by means of a real-life case-study in a large global software development organisation.**

I. Introduction

Software engineering has evolved from a process of few to an organisation of many [1], [2]. Where once stood a handful of developers, now stands a community of thousands of people, with multiple organisations, roles, cultures, in multiple locations [3]. This new angle on software processes, requires novel approaches for their representation and support. More in particular, *How can we support (global) social communities of developers?*

In previous work we obtained the state of the art in social communities [1] and produced a social community decision mechanism [4]. The missing keystone is to support social communities with an innovative and pro-active mechanism operating through services. The research hypothesis that drives the work in this paper is quite simple and equally intriguing: *social communities of developers can be supported by a global network of software and socio-technical services, spanning different organisations, sites, timezones and cultures.* The result is a service-network [5], [6] that blends the internet of services with large-scale, adaptable choreographies to deliver a powerful and scalable solution that adapts to the changes of a community. On one hand, software services are pieces of software operating under a service-dominant logic. These pieces of software collaborate together across the web using standard protocols, to deliver complex, adaptable functionality (e.g. cloud-based functionalities such as GoogleDocs). Much literature in service sciences provide ways to identify, monitor and adapt software services [7]. On the other hand, socio-technical services are hybrid human and software services, i.e. services that explicitly mediate the collaborative work of people within a social community, e.g. by fostering relevant community aspects [1], [4], or by increasing situation awareness of community members [8] or maintaining community's socio-technical congruence [9].

Supporting development communities with service-networks can be achieved by associating each social community emerging in an organisation with the set of software and socio-technical services needed to dynamically support its operations. Communication across communities would then become a service-assembly problem that we can study and support with state-of-the-art services technology. For example, suppose that project X in company Y involves two social communities. Both need explicit support. A single service-network can support project X blending services to support both social community types. A specific combination of services can be used to support X within its context. Ad-hoc adaptation dynamics can be integrated.

The benefits of this approach are manifold. Service-networks could be shaped using the state of the art in service sciences, to deliver a powerful and innovative solution for community-based software engineering [1], [2]. Moreover, a service-dominant logic would allow dynamic retrieval of services from the cloud, as needed to support communities. In addition, the approach could assist the emergence of Enterprise 2.0 technologies [10], blending social communities and services together.

This paper shows the potentials of this idea in practice, by reporting on its application to a real-life industrial scenario from a large global software development organisation [11]. We found that service-networks help pinpointing hazardous organisational barriers (i.e. impediments to the smooth cooperation within a community [12]) in software development. In addition, service-networks ease (semi-)automatic steering of development social communities. Our findings are encouraging but open up many intriguing research paths.

The rest of the paper is structured as follows: Section II explains related work; Section III applies the idea on a case-study; Section IV discusses the case-study while Section V concludes the paper.

II. Previous and Related Work

Our previous work offers theoretical premises to sustain the idea presented in this paper. In [13] we discussed the importance of studying social communities in (global) software engineering. In [1] we identified relevant social community

978-1-4673-3076-3/13 © 2013 IEEE

ICSE 2013, San Francisco, CA, USA
New Ideas and Emerging Results

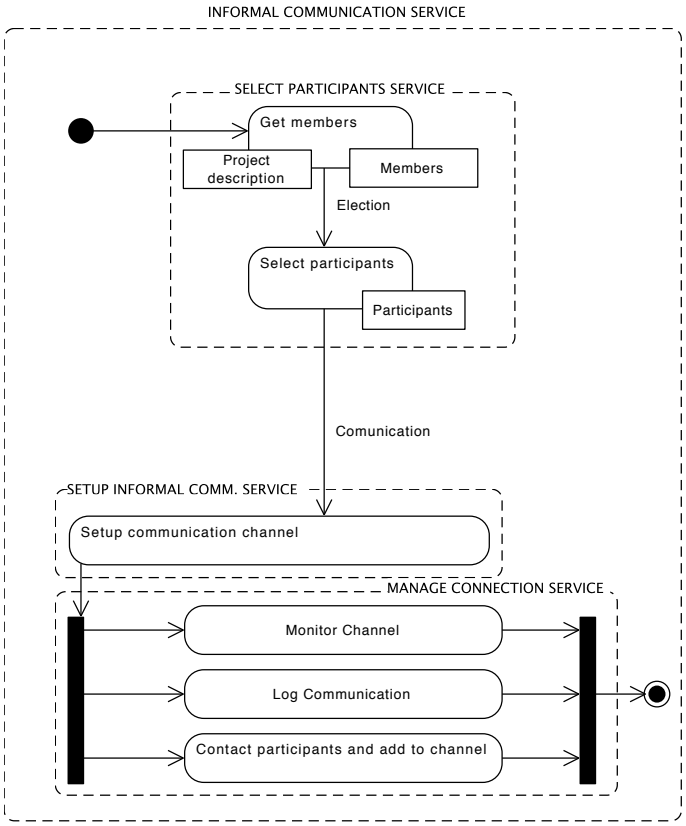

Fig. 1. Case-Study: informal communication service.

types from organisational and social-networks theory. The types we identified, can be used as core ingredients to express social community types observable in software engineering practice. The work in this paper offers a practical application of the idea on a real-life industrial scenario. We use empirical evidence and mechanisms from previous work [4] to apply service identification [14].

Related works such as [15], [16] present related and compounding ideas. In both works, the authors rise the abstraction level of service-based software solutions to introduce *design by units*. Units encompass human, infrastructure and hybrid services alike. Our idea of a service-network is indeed rotating around very similar concepts, i.e. socio-technical or software services to mediate human collaboration. Our work in this paper however, doesn't go as far as designing a complete solution, it shows services to support social communities in a real-life industrial scenario.

Similarly, works like [17] act towards the idea of using cloud-based solutions to aid global software development. Our idea could benefit from such mechanisms to deploy and adapt service-networks for (global) social communities.

III. Supporting Development Communities with Service Networks: A Case-Study

In pursuit of our goal, i.e. to support communities with service networks, we further studied the case contained in [11], [4]. The case entails two major corporations: A is a big software consulting and development company with over twelve thousand employees spanning four continents; B is a banking multi-national, serving over 85 million customers worldwide. In the case in question, company A (Netherlands Site) works for company B (Netherlands site) on a business-critical integration project. Company A works with an integrated Scrum model encompassing a partner development company from India. Figure 2 provides an overview of the case organisational structure. The integrated Scrum model adopted by A is realised with three development units (Ready, Done, Shippable) collaborating globally between two sites (India and Netherlands). In addition to the classic Scrum Model [18], the behavior in our case included a Wrapper sprint, in which the increment from the current sprint from company A, is "wrapped" inside systems at company B.

In previous work [4], we found that the three Scrum teams were working as: (a) a "Formal Network" in NL; (b) a "Community of Practice" in India; (c) finally, an "Informal Network" globally, across sites (see top part of Fig. 2). We focused on the global community, i.e. "Informal Networks", so that, for example, we could investigate support for time distance barrier of 4-hours. From [1] we learned that "informal communication" is key for "Informal Networks".

To identify software and socio-technical services to support "informal communication" in "Informal Networks " we applied the service identification method in [14]. We used available information in [1] and [6] as requirements. We elicited the behaviour of candidate services to support "informal communication" within "Informal Networks". Such behaviour (see Fig. 1) can be described in three stages, supported by one service each (see dashed boxes in Fig. 1). This is shown with the service-network in Fig. 2:

1) Service *"Select Participants"*, automates the process of participants election. We found that "informal interactions" take place between a number of participants, elected among members of the "Informal Network". In our case-study, Scrum Masters need informal lists of people-to-task allocations, to understand who is member of the "Informal Network" of developers. Moreover, from a set of members, a number of participants need to be selected according to current interaction needs. The "Select Participants Service" is a socio-technical service since it requires the input of humans and mediates their collaboration. For example, in our case-study, all teams ("Shippable","Ready" and "Done") are members of the "Informal Network", but during daily Scrum meetings only Teams "Ready" and "Done" are participants.
2) Service *"Setup Informal Communication Channel"*, automates channel creation and decommissioning. We found that "informal interactions" need an informal communication channel to be opened and deployed for participants. The "setup Communication Channel" is a software service, since it can be fully automated with state-of-the-art technology. For example, in our case, an

array of eye-catcher devices [1] were used to support daily Scrum meetings.

3) Service *"Manage Connection"*. We found that "informal interactions" need constant monitoring and logging services to adapt communication, e.g. to channel misbehaviour. In addition, interactions need to log exchanged information. Finally, unavailable but required participants need to be reached via other means and linked to the interaction channel. A "Connection Management" service can be offered to deliver these combined functions. The "Connection Management" service is a software service, since it can be fully automated with state-of-the-art technologies. For example, in our case, minutes of meeting were informally gathered and circulated among participants. In addition, informal calls and e-mails were used frequently to substitute eye-catcher interaction. More pro-active mechanisms (e.g. Enterprise Social Networks [10]) could be deployed to dynamically support communication needs.

The resulting service-network is illustrated in Fig. 2 where setup- and run-time services are differentiated and, for sake of clarity, the teams are repeated (see upper and lower part of the figure). Services in grey are needed to setup the service-network. Services in white are run-time services that support participants' interactions. More specifically, according to the Scrum process in our case-study, an "informal interaction" is needed many times during the agile process: (a) every day, for daily Scrum meetings where "Ready" and "Done" teams participate; (b) monthly, for Wrapper Sprints and Sprint Reviews, where all teams participate. In addition, Scrum masters are responsible to setup the service-network for "Informal Networks" by calling the "Informal Comm." service. To setup, the "Informal Comm." service, Scrum Masters call eligible participants by means of the "select Informal Comm. Participants" socio-technical service. The "Select Informal Comm. Participants" service provides a list of participants to the "Setup Informal Communication Channel" service that links participants' clients to the ad-hoc communication channel. Finally, once communication starts, the "Informal Comm." service deploys the "Manage Connection" service that manages the informal communication.

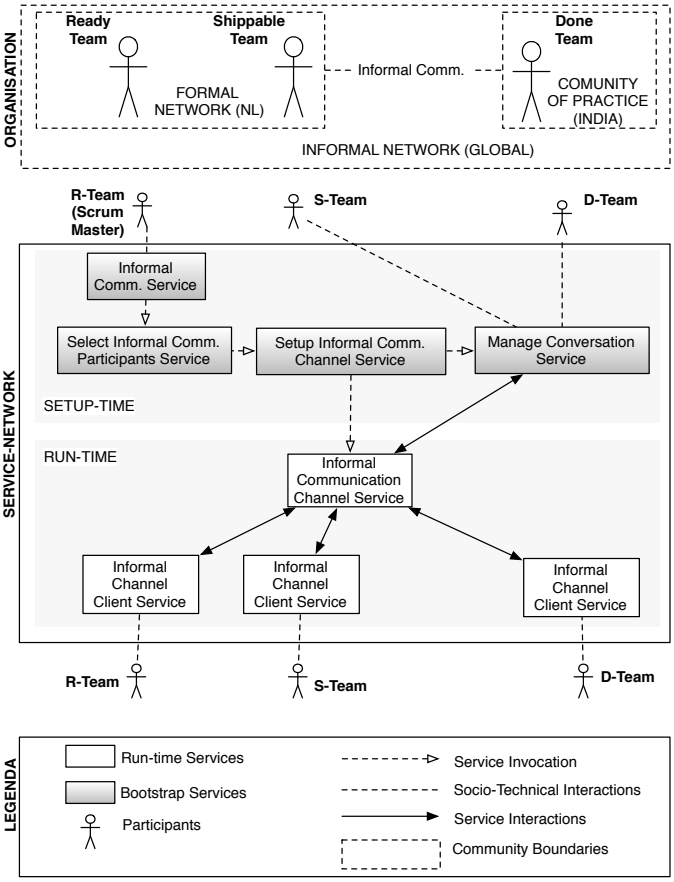

Fig. 2. Case-Study: three communities across two locations - during service-network setup, Scrum masters invoke the "Informal Comm. Service".

IV. Discussion

While the case-study illustrated in Fig. 2 specifically addresses the informal communication occurring globally across "Informal Network" participants, the work in [11] highlighted that major barriers are linked exactly to insufficient support of such informal communication. Comparing our solution supported by service-networks with the conclusions in [11], we made four key observations. In *italic* an interpretation.

First, we found that a service-networks help locating hazardous organisational barriers [12]. In our case, for example, due to EU policies some banking information could not be exchanged between sites. Using a service-network, we were able to narrow down the effect of this limitation. More specifically, we found that developers' "informal interactions" did not need to be concerned with the limitation since interactions did not discuss sensitive information. Rather, the limitation had to be taken into account every time the stakeholder (the Banking Corporation) came into play, since their "informal interaction" involved sensitive information. Therefore, information restrictions could be applied (e.g. through specific protocols) only during interactions that involved stakeholders (monthly Wrapper sprints in our case). *Representing software development communities with service-networks helps identifying and mitigating potential organisational barriers. As part of the work in [1] we found many organisational barriers. Further study should investigate which barriers occur during development and how these can be mitigated effectively, e.g. by adding services to the service-network dynamically.*

Second, we found that a service-networks help uncovering missing or implicit support. In the case-study reported in [11] none of the services identified were explicitly supported by software. For example, "informal interaction" service exists in the form of passive schedules and meeting dates, as regulated by Scrum masters. Conversely, the "informal interaction" service could be automated with state-of-the-art services. *Service-networks can be used to steer community operations in a*

[1] http://www.qconferencing.eu/product/eye-catcher/

(semi-)automated manner. This characteristic helps the emergence of Enterprise 2.0 technologies since it enables social communities to self-organise by means of software and based on their operational context. In perspective, more pro-active services could crawl scheduling information to dynamically and autonomously adapt meetings according to data currently available (e.g. applying evidence-based scheduling [19]).

Third, in previous work [4] three social community types were identified for the case presented in this paper, namely, "Informal Networks", "Community of Practice" and "Formal Networks". In this paper we found software/socio-technical services that support the key operation within an "Informal Network", i.e. "informal communication". We found that some of these services are needed for "Community of Practice" type as well. More specifically, the "informal interaction" service could be adapted and reused to support the collocated cooperation of people at the Indian site. *The existence of services that are already common to more than one community type suggests that service-networks could ease the work of co-operating communities. Multiple communities could blend in the same service-network. Each community becomes a services clique, cooperating with others through standard interfaces to deliver the much needed just-in-time retrieval of supporting service compositions. Within the resulting (adaptable) service compositions, both social and technical needs of the community meet and tune with the supporting service-network.* Fourth, we found that service-networks help identifying "accountability holes". In our case-study, for example, we found explicit responsibility allocation only for the "Select Informal Communication Participants" service, which is completely in the hands of Scrum masters. Other services did not have an accountable counterpart. *While this specific research topic is not addressed explicitly in our case, service-networks could indeed be used to allocate responsibility more dynamically. Allocation could aid developers' engagement, e.g. by means of reward mechanisms for highly-responsible employees.*

V. Conclusions and Future Work

In this paper we proposed the use of service-networks to support social communities of developers. We expanded this idea with a real-life industrial case-study. Although very humble, this work shows encouraging results. Service-networks help uncovering hazardous organisational barriers acting upon the community. More specifically, service-networks present the organisational structure in the form of services and allows a more granular analysis. In addition, service-networks help mitigating organisational barriers using service adaptation. Finally, service-networks help in (semi-)automatically steering the community operations. This complies with Enterprise 2.0 proactivity, openness and social-awareness.

Much work is still needed in this research. In the future we plan to elaborate the validation of the present work. In addition we plan to identify service patterns for all community types in [20], [1]. An analysis of community-support service patterns could reveal services that ease community interoperation. Finally, we plan to apply the identified patterns in additional real-life scenarios, to understand how communities (co-)operate successfully.

References

[1] D. A. Tamburri, P. Lago, and H. van Vliet, "Organizational social structures for software engineering," *to appear in ACM Computing Surveys*, pp. 1–35, 2012.

[2] J. Keyes, *Social software engineering*. Boca Raton, FL: Taylor & Francis, Auerbach Series, 2011.

[3] R. Sangwan, M. Bass, N. Mullick, D. J. Paulish, and J. Kazmeier, *Global Software Development Handbook (Auerbach Series on Applied Software Engineering Series)*. Boston, MA, USA: Auerbach Publications, 2006.

[4] D. Tamburri, P. Lago, and H. van Vliet, "Uncovering latent social communities in software development," *Software, IEEE*, vol. 30, no. 1, pp. 29 –36, jan.-feb. 2013.

[5] IfM and IBM, *Succeeding through Service Innovation: A Service Perspective for Education, Research, Business and Government*. Cambridge, United Kingdom: University of Cambridge Institute for Manufacturing, 2008. [Online]. Available: http://www.ifm.eng.cam.ac.uk/ssme/

[6] D. A. Tamburri and P. Lago, "Supporting communication and cooperation in global software development with agile service networks," *Proceedings of the European Conference on Software Architecture 2011*, vol. Springer, p. 8 pages, 2011.

[7] T. O. Group, "Soa source book," *http://www.opengroup.org/projects/soa-book/*.

[8] F. T. Durso and S. D. Gronlund, *Situation Awareness*. West Sussex, England: John Wiley & Sons, 1999, ch. 10, pp. 283–314.

[9] J. D. Herbsleb, "Global software engineering: The future of socio-technical coordination." in *FOSE*, L. C. Briand and A. L. Wolf, Eds., 2007, pp. 188–198.

[10] A. P. Mcafee, "Enterprise 2.0: The Dawn of Emergent Collaboration," *Management of Technology and Innovation*, vol. 47, no. 3, 2006.

[11] R. Noordeloos, C. Manteli, and H. van Vliet, "From RUP to Scrum in global software development: A case study," *International Conference on Global Software Engineering*, pp. 31 – 40, 2012.

[12] A. M. R. Correia, A. Paulos, and A. Mesquita, "Virtual communities of practice: Investigating motivations and constraints in the processes of knowledge creation and transfer." *Electronic Journal of Knowledge Management*, vol. 8, no. 1, pp. 11–20, jan 2010.

[13] D. A. Tamburri, E. di Nitto, P. Lago, and H. van Vliet, "On the nature of the GSE organizational social structure: an empirical study," *proceedings of the 7th IEEE International Conference on Global Software Engineering*, pp. 114–123, 2012.

[14] P. Lago and M. Razavian, "A pragmatic approach for analysis and design of service inventories," in *Proceedings of the 2011 international conference on Service-Oriented Computing*, ser. ICSOC'11. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 44–53.

[15] S. Tai, P. Leitner, and S. Dustdar, "Design by units: Abstractions for human and compute resources for elastic systems," *Internet Computing, IEEE*, vol. 16, no. 4, pp. 84 –88, july-aug. 2012.

[16] S. Dustdar and K. Bhattacharya, "The social compute unit," *IEEE Internet Computing*, pp. 64–69, 2011.

[17] P. Yara, R. Ramachandran, G. Balasubramanian, K. Muthuswamy, and D. Chandrasekar, *Global Software Development with Cloud Platforms*, 2009, p. 81.

[18] B. Gloger, "Scrum delivers or scrum and the toyota way," 2006.

[19] M. Gunderloy, *Painless project management with FogBugz*, 2nd ed., ser. Books for professionals by professionals. Berkeley, Calif.: Apress [u.a.], 2007.

[20] N. Milanovic, "Service engineering design patterns," in *Service-Oriented System Engineering, 2006. SOSE '06. Second IEEE International Workshop*, Oct. 2006, pp. 19–26.