Managing Software Sourcing with Alternative Workforces: A Holistic Overview and Research Agenda

Klaas-Jan Stol

Lero—the Irish Software Research Centre
University of Limerick
Limerick, Ireland
klaas-jan.stol@lero.ie

Working Paper – March 2016

Abstract. Trends such as the Internet of Things (IoT), smart devices, and software defined * (where '*' can represent networking, infrastructure, enterprise) cause a dramatic and ever-increasing need for additional software to leverage these advances. To illustrate: if the source code contained in a 1995 Mercedes S-class, had been printed the paper stack would be 3m high; by 2005, the stack was already 50 metres high, and in 2020, it is expected to reach a height of 830 metres, as high as the Burj Khalifa, the world's tallest building [Schneider 2015]. It is becoming increasingly clear that software organizations cannot deliver high-quality software at such a rapid pace while also delivering innovative and creative solutions. Consequently, organizations are looking at alternative software approaches, such as open source, inner source (adopting the open source development model inside organizations), and crowdsourcing. Expertise and solutions to software development problems is increasingly sourced from developer-specific social networks such as StackOverflow. However, these new collaboration models introduce new and significant challenges. In managing these alternative workforces, issues such as planning, quality control, and governance are far more challenging than in traditional settings.

Copyright © 2016 Klaas-Jan Stol

1 INTRODUCTION

Trends such as the Internet of Things (IoT), smart devices, and software defined * (where '*' can represent networking, infrastructure, enterprise) cause a dramatic and ever-increasing need for additional software to leverage these advances. To illustrate: if the source code contained in a 1995 Mercedes S-class, had been printed the paper stack would be 3m high; by 2005, the stack was already 50 meters high, and in 2020, it is expected to reach a height of 830 meters, as high as the Burj Khalifa, the world's tallest building [Schneider 2015]. It is becoming increasingly clear that software organizations cannot deliver high-quality software at such a rapid pace while also delivering innovative and creative solutions. Consequently, organizations are looking at alternative software approaches, such as open source, inner source (adopting the open source development model inside organizations), and crowdsourcing. Expertise and solutions to software development problems is increasingly sourced from developer-specific social networks such as StackOverflow. However, these new collaboration models introduce new and significant challenges. In managing these alternative workforces, issues such as planning, quality control, and governance are far more challenging than in traditional settings. This research program will allow organizations to use these alternative workforces more effectively and efficiently. Studies with both Irish and global collaborators will result in analytical frameworks, models and metrics to enable organizations to make better-informed decisions and build long-lasting sustainable relationships with alternative workforces which in turn can offer innovation and creativity.

2 SOFTWARE DEVELOPMENT COLLABORATIONS

Software engineering increasingly takes place in organizations and communities involving many people [Tamburri et al. 2013; Yan & Wang 2013]. In addition to traditional approaches such as in-house software development (insourcing), there is an increasing trend towards globalization with a focus on collaborations with and within communities, which may be known or anonymous. Open Source Software (OSS) in particular has had a dramatic impact on the software industry. OSS was initially approached with much skepticism and fear, and disregarded as a commercially viable alternative [Fitzgerald 2006]. Today, many organizations adopt OSS in multiple ways and increasingly rely on OSS communities for a steady stream of updates for open source products. For example, in the 1990s Microsoft compared OSS development to communism in an attempt to discredit this emerging phenomenon [Feller & Fitzgerald 2002].

However, in the past few months, Microsoft announced its first Linux-based operating system Azure Cloud Switch. Open-source-inspired strategies such as crowdsourcing [Stol & Fitzgerald 2014] and innersourcing [Stol et al. 2014] are also gaining considerable attention and are becoming viable approaches [Yan & Wang 2013]. Software ecosystems such as Google's Android platform (and third-party apps) are widespread [Jansen et al. 2013].

Fig. 1 presents our analysis of the various sourcing strategies from a customer's perspective. We position these in a circumplex based on two dimensions: control of the product offering and the extent to which a workforce is known. One popular definition of control in the literature characterizes it as any attempt to align behavior with organizational objectives [Kirsch 1996]. Quadrant I contains traditional approaches to software sourcing: insourcing is in-house software development with a clearly defined workforce, and 'traditional' outsourcing involves a workforce that can initially be characterized as 'unknown' since outsourcing suppliers are often a black-box for customers. Of course, outsourcing workforces are more known than in, for example, open source, as a contract must be signed with a known entity, and, given sufficient time a relationship and trust can develop if an outsourcing supplier is used for an extensive duration. In both Quadrant I strategies customers have a considerable degree of control.

Quadrant II contains single-vendor open source [Riehle 2011]; these are OSS projects whereby one organization owns and controls an OSS product. Examples of this are MySQL and Eclipse. Also in Quadrant II is inner-sourcing, a scenario in which an organization adopts OSS development principles for its internal development [Stol et al. 2014]. This approach is gaining considerable interest from companies such as Allstate, PayPal, Rolls-Royce, Samsung and Sony Mobile [Stol et al. 2014]. Inner source facilitates ad-hoc collaborations between organizational units that otherwise would not collaborate. Because inner source relies on motivated individuals and self-selection of tasks, an organization has limited control (by design) over the software being developed—the purpose of inner source is to create a culture of transparency and collaborations; management's role is that of empowerment.

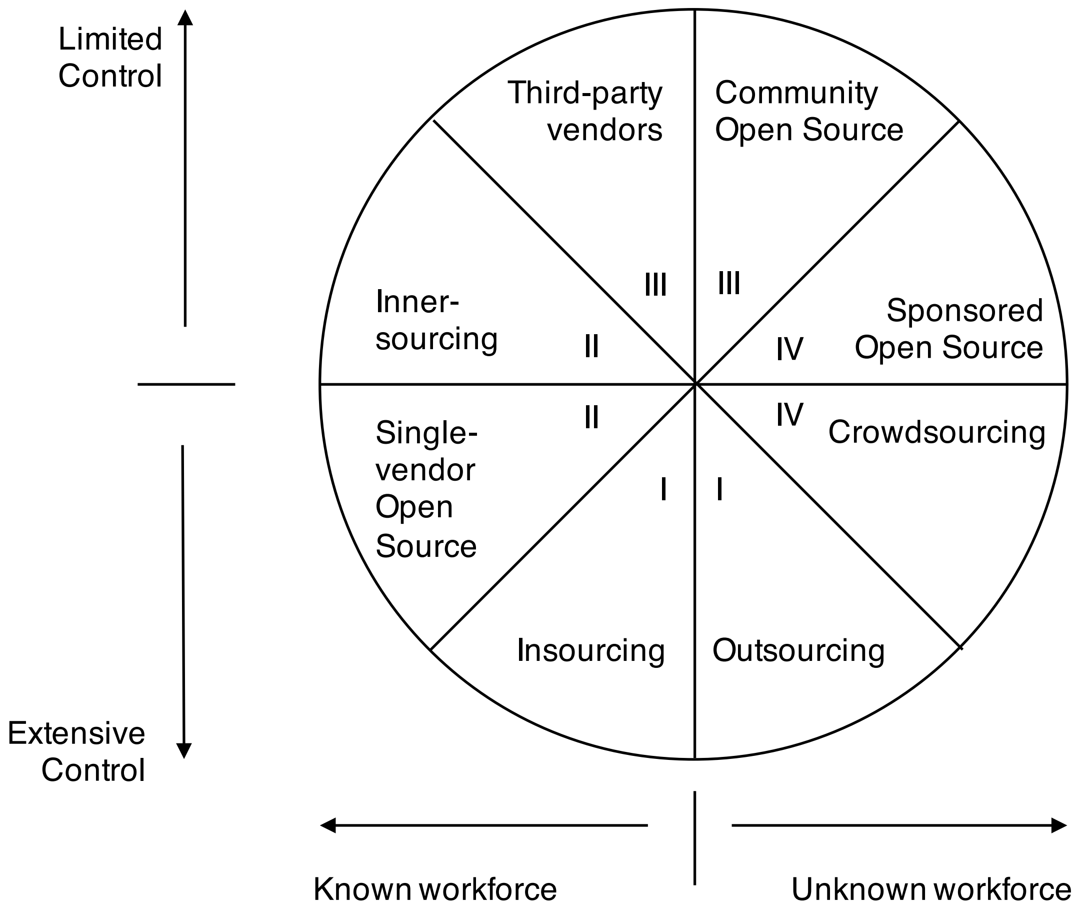

Figure 1. Sourcing strategies for software development

Quadrant III contains third-party vendors and community OSS. The former happens in software ecosystems [Jansen et al. 2013], whereby independent parties offer extensions or new functionality (Apps). Platform providers have limited control over the software developed; banning offerings to a platform (e.g. through an 'app store') is one way to exert such control. Such vendors are necessarily known as they usually advertise their offerings. Community OSS refers to 'traditional' open source [Riehle 2011], that is, OSS projects without any formal participation of firms (or non-profits) that can exert control over what is being developed. The workforce is very much unknown since developers are commonly using pseudonyms and little is known about specific individuals. The Debian Project (a Linux distribution) is one example of this which has a strong emphasis on the free/libre philosophy without corporate involvement [Michlmayr et al. 2015].

Sponsored OSS (quadrant IV) is similar to the single-vendor open source strategy, with the exception that an organization is merely involved as a co-developing party, and has no exclusive ownership, and therefore has limited control over the project as a whole. An example of this strategy is the Linux kernel—one study suggests that over 80% of all kernel development is done by paid developers [Corbet et al. 2013]. Next in quadrant IV is crowdsourcing, which is also inspired by open source [Stol & Fitzgerald 2014]. In crowdsourcing there is also an unknown workforce, at least up to the point that any post-delivery payments are made to the 'winner' of a crowdsourcing competition—even after payment, a customer will learn very little about a 'supplier.' In such a case a crowdsourcing organization has a significant level of control in terms of required features in a delivered software. Another form of crowdsourcing is bounty-sourcing, whereby a sponsor offers a bounty to implement or fix a specific feature in an OSS project. Internal crowdsourcing is another variant [Zuchowski et al. 2016].

In practice, an organization may face a mix of several strategies to develop software. For example, the OpenStack project (offering software for managing cloud infrastructure), involving several global companies such as EMC, HP and Intel, is a sponsored OSS project [Gonzalez-Barahona et al. 2013]; together these companies have a considerable level of (collective) control over the project, similar to a single-vendor OSS project.

There is a considerable body of knowledge on traditional approaches (in-/outsourcing), hence this research programme focuses on the remaining strategies in quadrants II-IV. A recent book on collaborative software development [Mistrik et al. 2010] presents a snapshot of the advances made in recent years, but most studies on collaborations tend to be among teams whose mutual relationships are well defined (i.e. not unknown and with defined control mechanisms). Organizations are increasingly engaging with such alternative developer communities [e.g., Fitzgerald 2006; Yan & Wang 2013, Teixeira 2014], on which they can exert different levels of control, and may or may not know anything about. These two dimensions, control, and extent to which a workforce is known, are guiding in this research.

Most of the research on collaborative software development tends to focus on collaborations within teams, between teams and among organizations [Mistrik et al. 2010]. In each of these scenarios, developers are employed, and are thus known and 'controllable' by their respective organizations. The proposed research focuses on what we call alternative workforces, which vary in much more dramatic ways than the more traditional workforces described above. Some but not all developers may be paid, developers may not be aware of each other (e.g. in a competition-based crowdsourcing setting, but also in open source) and the motivation and goals of developers may vary widely as well.

Alternative modes of software development such as open source, inner source and crowdsourcing offer several benefits. For example, large communities may benefit from the fact that many developers are able to review the code—this is often referred to as Linus's Law ('many eyeballs make all bugs shallow') [Fitzgerald 2006]. Furthermore, inner source can significantly help organizations in ensuring timely delivery of their products to the market; business units that find critical defects in a shared component shortly before a major release, can now fix issues themselves (as inner source offers access to all source code), rather than being dependent on the owner of that component [Stol & Fitzgerald 2015]. All three main modes listed above have the potential for creative, innovative or quality-improving solutions [Stol & Fitzgerald 2015].

Much research on open source and derived initiatives (i.e. inner source, crowdsourcing) focuses on initial adoption, but there is a paucity of research on sustainability of these initiatives. Key questions are: How can sourcing strategies be sustained if an organization has little influence on external workforces? And how can organizations build up sustainable relationships with unknown workforces?

The control dimension raises issues such as: governance approaches; ownership of innovation and IP; mechanisms to exert control such as payments; reputation of an actor in community-based development; conflict control and resolution; leadership and power-shifts. For example, employees who contribute to OSS projects on behalf of their employer build relationships and reputations with those communities that are partly 'personal'. If an employee leaves his/her current employer for a different one, the firm will partially lose this investment in these OSS projects [Henkel 2008]. OSS communities may also suffer from internal disagreements about the future of a project, which could cause 'forks' of projects, which greatly affects a project's sustainability [Gamalielson & Lundell 2014] because forking of a project may split a community of developers, jeopardizing a project's sustainability. Conflict negotiation has also been studied by Scacchi and colleagues [Elliott & Scacchi 2003; Jensen & Scacchi 2005]. Organizations start inner source initiatives to emulate the successes of open source communities internally—and such programmes require a lack of control and instead rely on empowerment of an internal workforce to self-select those tasks that they deem most useful. However, it is unclear how an organization's product strategies (driven by market trends and demands) can be sustained while relying on such an 'uncontrollable' model of software development.

The extent of to which a workforce is known or unknown raises issues such as: understanding goals of workforces, their motivations, beliefs, expectations, awareness, and norms of the workforce (as a heterogeneous group, i.e., these issues may vary per individual) versus those of a customer seeking to 'source' software; and the ability to retain knowledge and intellectual resources. One example of how these issues can disturb relationships between a 'customer' and 'supplier' is a misalignment of goals or motivation; OSS projects may be started by altruistic individuals, not to offer a fully functional and supported high-quality software solution. Organizations may have different expectations and assumptions. In a crowdsourcing scenario, the fleeting relationship with 'crowd' developers is a major concern from a knowledge management perspective [Stol & Fitzgerald 2014]. Thus, interacting and collaborating with an unknown workforce raises significant challenges for organizations whose aim it is to deliver commercial software products to a market or their clients.

While research exists on firm involvement in open source, inner source and crowdsourcing, this area is still in its nascent phase, and there is no integral research program that addresses the issues identified above.

3 ACKNOWLEDGMENTS

This work was supported, in part, by Science Foundation Ireland grant 13/RC/2094 and co-funded under the European Regional Development Fund through the Southern & Eastern Regional Operational Programme to Lero—the Irish Software Research Centre (www.lero.ie), Enterprise Ireland under grant IR/2013/0021 to ITEA2-SCALARE (www.scalare.org), and the Irish Research Council (www.research.ie).

4 REFERENCES

PJ Agerfalk, B Fitzgerald, K Stol. 2015. Software Sourcing in the Age of Open, Springer

A Capiluppi, K Stol, C Boldyreff. 2012. Exploring the Role of Commercial Stakeholders in Open Source Software Evolution. Proc. International Conference on Open Source Systems. Springer.

J Corbet, G Kroah-Hartman, A McPherson. 2013. Linux Kernel Development: How Fast It is Going, Who is Doing It, What They are Doing, and Who is Sponsoring It.

M Elliott, W Scacchi. 2002. Communicating and Mitigating Conflict in Open Source Software Development Projects. Institute for Software Research, UC, Irvine

J Feller & B Fitzgerald 2002. Understanding Open Source Software Development. Addison-Wesley B Fitzgerald. 2006. The Transformation of Open Source Software. MIS Quarterly 30(3)

J Gamalielsson, B Lundell. 2014. Sustainability of Open Source software communities beyond a fork: How and why has the LibreOffice project evolved? J Sys Soft 89, 128-145

JM Gonzalez-Barahona et al. 2013. Understanding How Companies Interact with Free Software Communities. IEEE Software 30(5)

VK Gurbani et al. 2006. A Case Study of a Corporate Open Source Development Model. Proc. ICSE J Henkel 2008. Champions of revealing—the role of open source developers in commercial firms. Industrial and Corporate Change, 18(3), 435–471

S Jansen, S Brinkkemper, M Cusumano (Eds.) 2013. Software Ecosystems: Analyzing and Managing

Business Networks in the Software Industry. Edward Elgar Press.

C Jensen, W Scacchi. 2005. Collaboration, Leadership, Control, and Conflict Negotiation and the Netbeans.org Open Source Software Development Community. Proc. HICSS 2005

E Kalliamvakou, D Damian, K Blincoe, L Singer and D German. 2014. Open Source-Style Collaborative Development Practices in Commercial Projects Using GitHub. Proc. ICSE

R Kazman, HM Chen. 2009. The metropolis model a new Logic for Development of crowdsourced systems. Communications of the ACM 52(7)

L Kirsch. 1996. The Management of Complex Tasks in Organizations. Organization Science 7(1)

S Krishnamurthy, S Ou, AK Tripathi. 2014. Acceptance of monetary rewards in open source software development. Research Policy. 43.

K Lakhani, RG Wolf. 2003. Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects. MIT Sloan Working Paper

TD LaToza et al. 2015. Borrowing from the Crowd: A Study of Recombination in Software Design Competitions. Proc. International Conference on Software Engineering, 551-562

M Michlmayr, B Fitzgerald, K Stol. 2015. Why and How Open Source Projects should adopt Time-Based Releases. IEEE Software 32(2).

I Mistrík, J Grundy, A van der Hoek, J Whitehead (Eds.). 2010. Collaborative Software Engineering, Springer, New York, pp. 307–328.

D Riehle. 2011. Controlling and Steering Open Source Projects. Computer 44(7)

W Scacchi. 2004. Free and open source development practices in the game community. IEEE Software 21(1)

M Schaarschmidt et al. 2015. How do firms influence open source software communities? Inf&Org 25 J Schneider 2015. Software-Innovations as key driver for the Green, Connected and Autonomous mobility. ARTEMIS-IA/ITEA-Co-Summit. https://itea3.org/co-summit-2015/presentations-1.html

K Stol et al. 2014. Key Factors for Adopting Inner Source. ACM Transactions on Software Engineering and Methodology 23(2)

K Stol, B Fitzgerald. 2014. Two’s Company, Three’s a Crowd: A Case Study of Crowdsourcing Software Development. Proc. ICSE’14, Hyderabad, India.

D Tamburri, P Lago, H van Vliet. 2013. Organizational Social Structures for Software Engineering. ACM Computing Surveys 46(1)

J Teixeira. 2014. Understanding Coopetition in the Open-Source Arena: The Cases of WebKit and OpenStack Proc. OpenSym'14, Berlin, Germany.

J Wesselius. 2008. The Bazaar inside the Cathedral. IEEE Software 25(3)

J Yan & X Wang. 2013. From Open Source to Commercial Software Development. ICIS 2013.

O Zuchowski et al. 2016. Internal Crowdsourcing: Conceptual Framework, Structured Review and Research Agenda. Journal of Information Technology. Forthcoming.