Bogdan Vasilescu

Social Aspects of Collaboration in Online Software Communities

Social Aspects of Collaboration in Online Software Communities

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de Technische Universiteit Eindhoven, op gezag van de rector magnificus prof.dr.ir. C.J. van Duijn, voor een commissie aangewezen door het College voor Promoties, in het openbaar te verdedigen op maandag 20 oktober 2014 om 16.00 uur door

Bogdan Nicolae Vasilescu

geboren te Ploiesti, Roemenië

Dit proefschrift is goedgekeurd door de promotoren en de samenstelling van de promotiecommissie is als volgt:

voorzitter:	prof.dr.ir. B. Koren
1^{e} promotor:	prof.dr. M.G.J. van den Brand
copromotor:	dr. A. Serebrenik
leden:	dr. V. Filkov (University of California, Davis)
	dr. P. Le Blanc
	prof.dr. T. Mens (University of Mons)
	prof.dr. B. Speckmann
	prof.dr. M.-A. Storey (University of Victoria)

Social Aspects of Collaboration in Online Software Communities

Bogdan Vasilescu

Promotor: prof.dr. M.G.J. van den Brand
(Eindhoven University of Technology)

Copromotor: dr. A. Serebrenik
(Eindhoven University of Technology)

Additional members of the reading committee:

dr. V. Filkov (University of California, Davis)
dr. P. Le Blanc (Eindhoven University of Technology)
prof.dr. T. Mens (University of Mons)
prof.dr. B. Speckmann (Eindhoven University of Technology)
prof.dr. M.-A. Storey (University of Victoria)

The work in this dissertation has been carried out under the auspices of the research school IPA (Institute for Programming research and Algorithmics).
IPA dissertation series 2014-11.

The work in this dissertation has been financially supported by The Netherlands Organisation for Scientific Research (NWO), under the project NWO 600.065.120.10N235.

A catalogue record is available from the Eindhoven University of Technology Library
ISBN: 978-90-386-3685-6

Cover design: Gabriella Sperotto
Reproduction: Ipskamp Drukkers, Enschede, The Netherlands

Acknowledgements

Grad school was undoubtedly the best time of my life! The freedom to choose interesting problems to work on, a stimulating atmosphere of intellectual inquiry, great mentorship, numerous opportunities for academic, professional, and personal development, the chance to see the world through conference travels and collaborations (for example, far away places in Canada, the east and west coasts of the USA, Hawaii, Japan, and almost India) are just few of the perks associated with my PhD life. For these and many others I am forever grateful to my promotors, Alexander Serebrenik and Mark van den Brand. Alexander, thank you for being the voice of passion during our collaboration, for your constant eagerness to explore new ideas, inexhaustible energy, and unconditional support. Mark, thank you for being the voice of reason, for your experience, ability to filter ideas and steer me in the right directions, and for teaching me that life is incomplete without good food. The combination of your two voices, of passion and reason, created a perfect mentorship environment!

During my graduate studies I was fortunate to collaborate and coauthor papers with many different people. They have all contributed significantly to my academic development, and some even to this dissertation, as coauthors on publications included here. Besides Alexander and Mark, thank you Andrea Capiluppi, Prem Devanbu, Vladimir Filkov, Mohammad Gharehyazie, Mathieu Goeminne, Georgios Gousios, Erik Kouters, David Lo, Markus Lumpe, Tom Mens, Kate Pek, Daniel Pletea, Daryl Posnett, Gregorio Robles, Qi Xuan, and Andy Zaidman.

I've spent a significant part of my three PhD years at UC Davis, during two visits in the springs of 2013 and 2014. I am grateful to Vladimir and Prem for the invitations, our collaboration, and for showing me a great time in and around Davis. You've made my visits to California unforgettable! I was also fortunate enough to visit Tom Mens' software engineering lab at UMons (repeatedly), Stephan Diehl's group and the DBLP team at University of Trier, Stéphane Ducasse's RMoD Team at INRIA Lille, and Arie van Deursen's SERG group at TU Delft. Thank you all for the opportunity.

I would also like to extend my sincere gratitude to the members of my reading committee, for reviewing this dissertation and your insightful comments: Peggy Storey from University of Victoria, Vladimir Filkov from UC Davis, Tom Mens from University of Mons, Pascale Le Blanc and Bettina Speckmann from Eindhoven University of Technology.

No matter where my work has taken me, be it Eindhoven, Davis, Mons, or elsewhere, my colleagues have always created a very pleasant working environment. Thank you, Ali Afroozeh, Marcel van Amstel, Suzana Andova, Tineke van den Bosch, Dragan Bosnacki, John Businge, Sacha Claessens, Sjoerd Cranen, Yanja Dajsuren, Luc Engelen, Joost Gabriels, Mohammad Gharehyazie, Mathieu Goeminne, Jan Friso Groote, Jeroen Keiren, Ruurd Kuiper, Luna Luo, Maarten Manders, Arjan van der Meer, Margje Mommers-Lenders, Neda Noroozi, Baishakhi Ray, Eric Scheffers, Ana Sutii, Ulyana Tikhonova, Zhaopeng Tu, Tom Verhoeff, Anton Wijs, Tim Willemse, Qi Xuan, and Dana Zhang.

A big hand goes out to members of the Stack Overflow, GitHub, Gnome, and R communities. Thank you for sharing your time and knowledge for the benefit of others, for welcoming our research, and for generously agreeing to take part in our user surveys and interviews. Your contributions are highly appreciated!

The cover of this dissertation is a metaphor. Similarly to collaboration being paramount to activity in successful online software communities, it is the interaction between the different objects depicted in the cover that leads to something greater than the sum of its parts. Thank you, Gabriella Sperotto, for the expressive design!

None of this work would have been possible without the financial support of NWO, the Netherlands Organisation for Scientific Research, who funded my PhD under the project NWO 600.065.120.10N235. I am also indebted to Océ Nederland BV, whose generous scholarship brought me to Eindhoven in the first place, and enabled me to pursue the Master's program that prepared me for this PhD.

Finally, I would like to thank my family and friends for their continuous support and the much appreciated distractions they provided.

Bogdan Vasilescu
Eindhoven, August 2014

Table of Contents

Chapter 1

Introduction

1.1 Background

Historically, software engineering research has focused mainly on technical improvements to the software development process or to the quality of the final product. However, creating software requires more than applying the right technical solutions in the right technical ways. Software engineering is inherently a collaborative, coordinated, human venture: software is developed by people, and for people. These people have different personalities, different educational and cultural backgrounds, and different expertise; they work in different environments and often in different time zones, have different motivations, and behave differently under different conditions.

As such, this realisation is not new and can be traced back 40 years to the seminal works of Gerald Weinberg [337] and Fred Brooks [38]. In "The Psychology of Computer Programming" [337], Weinberg was arguably the first author to address programming as human performance and a social activity, stating that when it comes to the quality of software development, technological issues are subordinate to human issues. Shortly after, Brooks reiterated this thesis in "The Mythical Man-Month" [38], further acknowledging that human aspects of software development have an important influence on software productivity.

However, despite this early realisation, it was not until recently that the study of social aspects (also known as human aspects) of software engineering took off, with a new research field, referred to as social software engineering [7] or collaborative software engineering [203], springing up. As evidence to this growing popularity stand recently-established and successful dedicated workshops, such as CHASE (International Workshop on Cooperative and Human Aspects of Software Engineering; since 2008), or SSE (International Workshop on Social Software Engineering; since 2008), various recently-established related conferences, such as SBP (International Conference on Social Computing, Behavioral-Cultural Modeling, & Prediction; since 2008), SocialCom (International Conference on Social Computing; since 2009), or SocInfo (International

Conference on Social Informatics; since 2009), as well as expansions of the programs of all major software engineering and computer-human interaction conferences (*e.g.*, ICSE—International Conference on Software Engineering, or CSCW—ACM Conference on Computer-Supported Cooperative Work and Social Computing) to accommodate this topic.

The central thesis in this line of research is that software engineering is a collaborative, knowledge-intensive, human-centric activity. Architects, designers, developers, users, testers, and managers communicate, share knowledge, exchange artifacts, and coordinate their efforts in order to create and maintain large and complex software systems. Understanding the social aspects of software engineering is, therefore, crucial to understanding how to build software effectively, improving the creation and maintenance of software systems as well as the management of software projects.

Numerous recent studies support this claim. For example, Seaman and Basili [260] provided a better understanding of how organizational structure helps or hinders communication in software development. Herbsleb and Grinter [124] revealed how distribution of development across time zones, cultures, and (natural) languages breaks down communication and leads to coordination problems. Potts and Catledge [236] offered evidence about how different patterns of collaboration affect a team's convergence on a common vision during the conceptual design phase. Bird *et al.* [28] uncovered relationships between artifact properties and social network properties (*e.g.*, between a team member's commit activity and their social status), using data derived from the team's email interactions. Pinzger, Nagappan, and Murphy [229] built socio-technical networks using the history of developer contributions to different modules, and found that social network properties have good predictive power in determining failure-prone binaries. Bettenburg and Hassan [25] reported similar findings, showing that social information augments traditional product and process-based metrics in statistical defect prediction models. Nagappan, Murphy, and Basili [210] showed that organizational metrics are related to, and are effective predictors of failure-proneness, using data from Windows Vista. Finally, Tsay, Dabbish, and Herbsleb [302] provided evidence that social information is important to developers when evaluating potential contributions to open source software projects.

1.2 Online software communities

This dissertation is set in a distributed (online) setting, endemic to open-source software (OSS) development. OSS are developed by communities of geographically- and temporally-distributed contributors, ranging from professional software developers to volunteers from varied backgrounds who, despite participating in a very decentralized process, succeed to work together effectively and productively [63, 98]. OSS contributors typically self-organise in online communities, where they participate in a collaborative effort, such as developing and evolving software systems, offering user support, or sharing knowledge. Myriads of OSS communities exist. Among the most prominent are those around the Linux operating system and its various distributions, the Gnome desktop environment, the Apache or Mozilla software foundations or, more recently, the GitHub repository hosting platform.

This dissertation assumes a broader definition of the term *OSS community*, understood here as *online software community*, in order to encompass the Stack Exchange network of question and answer sites (*e.g.*, Stack Overflow for programming-related questions).

Although Stack Exchange sites are strictly speaking not OSS, Stack Exchange communities and OSS communities have similar underlying principles, *i.e.*, both are self-organised and collaboratively maintained, allowing us to use the terms OSS community and online software community interchangeably throughout this dissertation.

The spectrum of online software communities is *diverse*. Different communities operate by different rules and offer different incentives to contribute. For example, more traditional communities such as Gnome or Apache rely mostly on the intrinsic motivations of developers [165, 242, 265]. In contrast, younger communities offer social media and gamification features (*e.g.*, contributors to Stack Overflow are rewarded for their activity with reputation points and badges) as well as increased visibility to peers (*e.g.*, the activity and achievements of GitHub and Stack Overflow contributors are aggregated and displayed on public profile pages). Moreover, the spectrum of online software communities is *dynamic*. For example, once the most important forge in the OSS ecosystem, SourceForge, has been surpassed in popularity by the much younger GitHub[1]. Similarly, mailing lists, the historical hub of user support activities in OSS, are being replaced by social question and answer (Q&A) sites such as Stack Overflow. Furthermore, online software communities are often *interdependent*. For example, a Gnome contributor may engage simultaneously in different communities part of the Gnome ecosystem, where she may take on different roles or instead choose to specialise in similar tasks. Similarly, a GitHub contributor may participate simultaneously in Stack Overflow, where she may seek help from peers or instead share her knowledge and expertise to help educate others.

Since collaboration and coordination among members in online software communities is computer-mediated, a wealth of trace data is publicly available, from different repository types. For example, source code is stored in version control systems such as Subversion (SVN) or Git, which offer conflict management features; bugs are entered in issue management systems such as Bugzilla or Jira, which facilitate traceability and add structure to the bug fixing process; email communication is publicly archived in mail archives, to increase transparency and facilitate learning by other team members; finally, data dumps with historical data have become available for both Stack Exchange[2] and GitHub [110, 112, 113].

1.3 Research questions

The previous discussion paints a picture of online software communities as **diverse** collectives, often overlapping or interdependent, dynamic and evolving with time. Yet, the lion's share of related research, although addressing numerous social aspects of collaboration in software engineering, such as organizational structure or communication social networks (recall the examples in Section 1.1), has left the issues of diversity both within and between different online software communities to a large extent underexplored. In seeking to fill this research gap, this dissertation focuses on different facets of diversity in representative online software communities, and explores how differences between participants in these communities or between the communities themselves relate to individual-level or community-level outcomes.

[1] `http://readwrite.com/2011/06/02/github-has-passed-sourceforge`, acc. June 2014
[2] `https://archive.org/details/stackexchange`, acc. June 2014

To explore these characteristics, common also for biological ecosystems, we borrow a metaphor from ecology and view online software communities as *knowledge-sharing ecosystems* [198, 277]. Software developers are the organisms, or biotic components. They interact among themselves and with the environment, understood as software and hardware tools, or abiotic components. These biotic and abiotic components are linked together forming *knowledge-flow paths*, *i.e.*, interactions between *knowledge seekers* (consumers) and *knowledge providers* (producers) via a *medium* through which knowledge flows.

Arguably the predominant knowledge-flow path within any OSS community is that between developers as knowledge providers, and the entire community, or project, as knowledge consumer. Knowledge is shared, *e.g.*, through contributions to the source code repository (the medium), with the goal of driving further the evolution of the software project and ensuring the ecosystem's sustainability. Developer expertise (knowledge), therefore, becomes the primary resource in this knowledge exchange. However, given the natural diversity of human beings, one can also expect heterogeneity of knowledge providers within an OSS ecosystem. Some contributors are likely to be considerably more active than others; some will possess different skills than others; not all will be involved in the same activity types.[3] Stated differently, OSS communities are socially and technically diverse, with contributors communicating, interacting and collaborating differently. The social interactions between OSS contributors, as well as their degree of project participation have been reported repeatedly to influence software quality and complexity (*e.g.*, [25]). Social diversity in technical teams is also known to influence the team's performance (*e.g.*, [213]). In this context we wish to understand how diverse online software communities are in terms of developer expertise, whether certain roles are exclusive to a select few developers or instead developers wear different hats in different situations, or how this specialisation (or lack thereof) relates to individual productivity. Specifically, we ask:

RQ1. How diverse are OSS communities in terms of developer expertise?

Professional expertise development has already established itself as a continuous process of life-long learning and self-driven development of up-to-date expertise [34]. In the software engineering world, numerous emerging social environments, among the most prominent being GitHub (currently the largest code hosting platform in the world, with over six million users) and Stack Overflow (the most visible social Q&A platform for programming questions and answers, currently having almost three million users), support this process by offering collaboration, experimentation and knowledge sharing opportunities, as well as a chance to advertise one's expertise to peers and potential recruiters [45].

Both GitHub and Stack Overflow are packed with social media features, ranging from the ability to follow other developers and projects (GitHub), through the ability to give positive or negative votes to questions and answers (Stack Overflow), to the ability to collaboratively edit content, or engage in discussions over pieces of code (both GitHub and Stack Overflow). Yet, despite the multitude of recent studies

[3]Many different activity types are possible within an OSS community, such as coding, translating, or writing documentation (see Chapter 2 for an extensive discussion). The artifacts resulting from all these activities are typically stored in the project's source code repository.

exploring how (professional) social media is revolutionising collaboration, coordination, communication, and learning for software developers (all important aspects of modern expertise development) [23, 24, 65, 284, 285, 298, 299], research on how programming in a socially-networked world might impact software quality and software engineering practices is in its infancy [284, 285].

Focusing on Stack Overflow, we distinguish multiple participation scenarios: developers acting as either knowledge seekers, knowledge providers, or both. Due to its fast response times [191] and good technical solutions [223], Stack Overflow has become ubiquitous in the daily information foraging (knowledge seeking) activities of software developers [298, 299]. Developers can also contribute their expertise back to the Stack Overflow community, to satisfy a demand for knowledge of other developers, perhaps less experienced, or of the users themselves. Continuing the ecological metaphor, however, one can view Stack Overflow and other social Q&A platforms like it as *disturbances* to OSS knowledge-sharing ecosystems. Clearly, participating in Stack Overflow may have a beneficial impact on the productivity of OSS developers, since Stack Overflow provides quick solutions to technical challenges. However, participating in Stack Overflow may lead to interruptions that could impair a developer's performance [285]. First, the lack of integration between Q&A sites and modern IDEs [18, 57] forces developers to interrupt their flow and change context every time they need to deal with them, thus potentially delaying their activity. Second, social communication activities, such as asking or answering questions on Stack Overflow, are yet other activities that compete for the already-limited time resources of developers [346]. We wish to understand whether participation in social Q&A sites is related to the productivity of OSS developers, how it impacts their working rhythms, and whether some groups of developers benefit more from participation than others. Focusing on Stack Overflow, the most prominent such venue, we ask:

RQ2. How does participating in social Q&A impact the working rhythms of OSS developers?

Diving deeper into what made social Q&A platforms such as Stack Overflow so popular in recent years, we note that perhaps their most distinctive feature when compared to other information sharing venues is *gamification* [74] (*i.e.*, the use of game elements in a non-game context in order to engage users). Participants compete to obtain reputation points and badges, which enable additional privileges (*e.g.*, moderation rights) once various thresholds are exceeded. Reputation points and badges on such sites can be seen as a measure of one's expertise by peers and potential recruiters [45], and are known to attract users to participate, and to motivate them to contribute more [11, 74].

However, while gamification has become omnipresent in modern social collaboration and learning environments used by software developers (with arguably its most popular implementation on Stack Overflow), such mechanisms are not without criticism. For example, in its current implementations gamification is almost synonymous with achievement, while achievement as a motivator is not necessarily well suited for many cultures around the world [150]. Scandinavian cultures, for instance, are known to encourage social equality, social stability, and uniformity [150, 259], which suggests that gamification systems for Scandinavian users that are premised on achievement and differentiation by rank would be less appropriate culturally. Similarly, gender differences also become apparent in competitive mixed-gender environments, where women tend to

be less effective than men [105]. We focus on gender, a common source of discrimination and stereotyping in the male-dominated "hacker" culture characteristic of OSS [208, 292]. OSS is considered to be unfriendly to women [304, p.194], who are underrepresented at not more than 10%[4] of all developers [70, 244]. In OSS, sexist behavior is said to be "as constant as it is extreme" [209]. Active discrimination towards women in OSS [208] also leads to the so-called "impostor syndrome", *i.e.*, despite being knowledgeable and professionally well-settled, women consider themselves to be disqualified or frauds [51].

While in the previous research question we set out to understand some of the individual-level effects (on developer working rhythms or developer productivity) associated with participating in social Q&A, here we seek to understand if and how such platforms are changing communication and knowledge-sharing practices in online software communities in general. Specifically, mailing lists have traditionally been the preferred communication medium between knowledge seekers and knowledge providers in online software communities. Currently, the different incentives for participation offered by social Q&A sites, coupled with their richer user interfaces and access to wider audiences, have the potential to challenge the supremacy of mailing lists as de facto communication medium in OSS. In this context we wish to identify whether knowledge-sharing is transitioning from an older, mailing list modality to a new, social Q&A modality, and understand some of the effects associated with such a transition. We ask:

RQ3. How are social Q&A sites changing knowledge sharing in OSS communities?

The previous questions revolve around what can be considered typical applications of repository mining techniques, themselves products of the Mining Software Repositories (MSR) community, in that they involve extracting and analysing data from *software* repositories. The MSR community, however, has already set about to go beyond traditional software repositories in its 2008 roadmap [121]. The last part of this dissertation is dedicated to reflection on the MSR community, MSR techniques, and their applicability to non-traditional repositories.

We start with the issue of inconsistent identity data, one of the most common challenges faced when analysing trace data from OSS repositories [121], with many applications outside MSR, *e.g.*, in law enforcement [331], genealogy [227], or data mining [49]. In OSS, recognising that different contributions belong to the same contributor is often challenging, since developers may use different aliases in different repositories (*e.g.*, a GNOME developer also contributing to a GNOME mailing list, or a GITHUB developer also participating in STACK OVERFLOW), and even different aliases within the same repository at different times (*e.g.*, even during the course of the same day, an OSS contributor could have contributed code from her work and her personal computers, configured to use different Git aliases, or she could have contributed to mailing list discussions from different email accounts). Although MSR researchers have devised numerous heuristics to deal with such potential inconsistencies (*e.g.*, [28, 49, 106, 231, 245]), correctly identifying who's who in OSS communities (process known as identity merging) remains challenging [106].

[4]Women are underrepresented not only in OSS but also in commercial software development, although companies such as Google report slightly larger numbers of female technology employees (17%; see `http://www.ibtimes.com/women-tech-meet-google-over-gender-gap-silicon-valley-1593168`, acc. July 2014). A self-reported dataset of the representation of female software engineers in different technology companies is available on GITHUB at `https://github.com/triketora/women-in-software-eng`.

In trying to scale MSR techniques to very large repositories [121], on the one hand, and transfer MSR techniques outside the software engineering community, on the other hand, we wish to understand how current identity merging algorithms and heuristics perform in the presence of noisy data (expected with larger repositories), and whether more general solutions, requiring less domain knowledge, exist. Specifically, we ask:

RQ4. How can we improve existing MSR identity merging techniques?

We end by reflecting back on the broader software engineering research community, of which MSR is a part. Similar to online software communities as knowledge-sharing ecosystems [198, 277], the software engineering research community can also be viewed as a knowledge-sharing ecosystem. Researchers collaborate and share their knowledge through publications in scientific conferences and journals. The spectrum of publishing venues is very diverse: different venues have different scope (some, more narrow, focus on a specific subdomain of software engineering, such as maintenance, reverse engineering, or program comprehension; others, wider-scoped, include a broader range of topics), different age (*e.g.*, MSR, the working conference on Mining Software Repositories, started in 2004, while ICSE, the International Conference on Software Engineering, is in existence since 1975), different barriers to entry for newcomers, and different scientific prestige. Similarly to contributors to OSS communities, researchers may become involved in different activity types (*e.g.*, some will only contribute as authors, others will serve on program committees or editorial boards, still others will engage in both).

We wish to understand to which extent MSR techniques can be applied to studying diversity in the software engineering research community, as manifested through software engineering conferences. Mining software engineering *conference* repositories requires overcoming some of the traditional challenges MSR research faces [121], such as exploring non-structured data (*e.g.*, data about program committee composition is typically available through a different webpage for each edition of each conference series, with different formats) and linking data between repositories (*e.g.*, authorship information can be extracted from the computer science bibliographic database DBLP[5] or from proceedings volumes; program committee membership information is typically available on the different conference webpages), while maintaining data quality (*e.g.*, between the different data sources, the names of authors and program committee members need not be consistent). Specifically, we ask:

RQ5. How can we transfer MSR techniques to studying diversity among software engineering conferences?

1.4 Outline

This dissertation contributes to the body of research in social software engineering through a series of empirical studies of representative online software communities. The studies that comprise the different chapters of this dissertation have all been published in peer-reviewed conferences and journals. While each chapter has distinct core contributions, there is some redundancy in sketching the context and describing the methods used in

[5] `http://dblp.uni-trier.de`

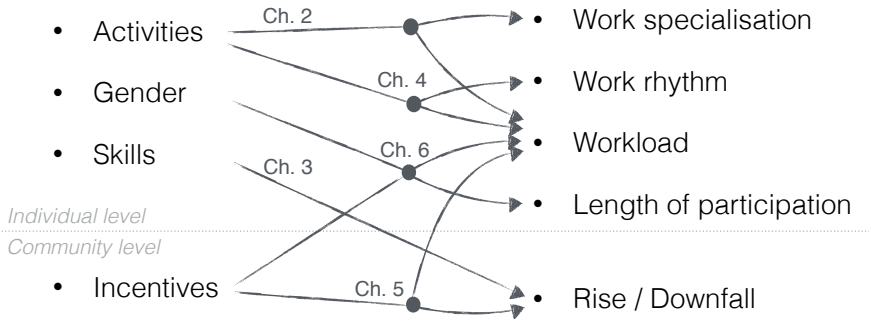


Figure 1.1: Overview of the relations explored in Chapters 2 to 6 between different facets of diversity and individual-level or community-level outcomes.

each study. This redundancy has not been eliminated in order to make it possible to read each chapter independently.

An overview of the studies included in this dissertation also appeared at the Doctoral Symposium of ICSE [313] and the Doctoral Colloquium of CSCW [314]:

[313] Bogdan Vasilescu. Human aspects, gamification, and social media in collaborative software engineering. In *Companion Proceedings of the 36th International Conference on Software Engineering (ICSE)*, pages 646–649. ACM, 2014

[314] Bogdan Vasilescu. Software developers are humans, too! In *Companion of the ACM Conference on Computer Supported Cooperative Work and Social Computing (CSCW)*, pages 97–100. ACM, 2014

The main body of this dissertation is structured in two parts. The first part, comprising Chapters 2 to 6, explores relations between different facets of diversity in online software communities and individual-level or community-level outcomes (see Figure 1.1 for an overview). The second part, comprising Chapters 7 and 8, is dedicated to reflection on MSR techniques and their applicability to non-traditional repositories.

Chapter 2: Specialisation of labour. In this chapter, we explore diversity of activities carried out by contributors to the GNOME ecosystem, and how workload and specialisation of labour vary in relation to these activities, addressing **RQ1**. We observed that next to coding the activities of localization, development documentation, and building are prevalent throughout the ecosystem. We also observed notable differences between frequent and occasional contributors in terms of the activity types they are involved in and the number of projects they contribute to. Occasional contributors and contributors that are involved in many different projects tend to be more involved in the localization activity, while frequent contributors tend to be more involved in the coding activity in a limited number of projects. This chapter consists of the following journal article:

[320] Bogdan Vasilescu, Alexander Serebrenik, Mathieu Goeminne, and Tom Mens. On the variation and specialisation of workload – A case study of the GNOME ecosystem community. *Empirical Software Engineering*, 19(4):955–1008, 2013

Chapter 3: A model of skill diversity. In this chapter, we continue addressing **RQ1** with a study of linguistic diversity in OSS. Inspired by measures of linguistic diversity from the study of natural languages, we propose a measure of expertise (skill) diversity in OSS communities with respect to a certain technical skill, such as knowledge of a given programming language. Such a measure can signal, for instance, when an OSS community is at risk of not having enough maintainers for code implemented in certain programming languages. We illustrate this scenario with a case study of Emacs, a popular OSS text editor in development since the mid-1970s, which contains code implemented in many different programming languages. This chapter consists of the following publication:

[325] Bogdan Vasilescu, Alexander Serebrenik, and Mark G. J. van den Brand. The Babel of software development: Linguistic diversity in open source. In *International Conference on Social Informatics (SocInfo)*, volume 8238 of *Lecture Notes in Computer Science*, pages 391–404. Springer, 2013

Chapter 4: Working rhythms across communities. In this chapter, we revisit the exploration of diversity of activities in online software communities from a broader perspective. Rather than distinguishing between different activities carried out *within* a given community (like in Chapter 2), we focus on developers who participate simultaneously *across* communities. We address **RQ2** with a study analysing the interplay between GitHub coding and Stack Overflow knowledge sharing activities for developers active on both platforms, focusing on effects associated with cross-platform participation on individual productivity and working rhythms. We show that active GitHub contributors typically engage in Stack Overflow as experts, asking few questions and providing many answers for others. Moreover, despite the interruptions incurred, the Stack Overflow activity rate correlates with the code changing rate on GitHub. This chapter consists of the following publication:

[318] Bogdan Vasilescu, Vladimir Filkov, and Alexander Serebrenik. Stack Overflow and GitHub: Associations between software development and crowdsourced knowledge. In *International Conference on Social Computing (SocialCom)*, pages 188–195. IEEE, 2013

Chapter 5: Transition to gamified environments. In this chapter, we start addressing **RQ3** with a study that charts the changes in behaviour of contributors to R (a widely-used tool for data analysis) mailing lists as they migrate into the gamified Stack Exchange environment. We find that user support activities in the R community show a strong shift away from mailing lists and towards Stack Exchange. Moreover, knowledge providers contributing to both communities provide faster answers on Stack Exchange than on the mailing list, and their total output increases after the transition. This chapter consists of the following publication:

[319] Bogdan Vasilescu, Alexander Serebrenik, Premkumar T. Devanbu, and Vladimir Filkov. How social Q&A sites are changing knowledge sharing in open source software communities. In *ACM Conference on Computer Supported Cooperative Work and Social Computing (CSCW)*, pages 342–354. ACM, 2014

Chapter 6: Gender issues. In this chapter, we continue addressing **RQ3** and revisit the gamified Stack Overflow environment from the perspectives of gender representation and participation patterns, comparing Stack Overflow to two communities without gamification features, Drupal and WordPress. We find that while women are under-represented in all three studied communities, they show different participation patterns on Stack Overflow, where they disengage sooner than men. However, relative to the duration of their engagement in the community, women on Stack Overflow are at least as active as men. This chapter consists of the following journal article:

[316] Bogdan Vasilescu, Andrea Capiluppi, and Alexander Serebrenik. Gender, representation and online participation: A quantitative study. *Interacting with Computers*, pages 1–24, 2013

An earlier version of this article first appeared as the conference paper:

[315] Bogdan Vasilescu, Andrea Capiluppi, and Alexander Serebrenik. Gender, representation and online participation: A quantitative study of Stack Overflow. In *ASE International Conference on Social Informatics (SocialInformatics)*, pages 332–338. IEEE, 2012

Chapter 7: Identity merging. In this chapter, we address **RQ4** by analysing identity data extracted from Gnome Git repositories, classifying the differences in developer aliases, and discussing robustness of two representative existing identity merging algorithms with respect to these types of differences. We then introduce a new identity merging algorithm inspired by Latent Semantic Analysis, a popular information retrieval technique expected to perform well in presence of noise. We evaluate the performance of our algorithm empirically by means of cross-validation, and show an improvement over existing algorithms in terms of precision and recall on worst-case input data from Gnome Git repositories. This chapter consists of the following publication:

[163] Erik Kouters, Bogdan Vasilescu, Alexander Serebrenik, and Mark G. J. van den Brand. Who's who in GNOME: using LSA to merge software repository identities. In *International Conference on Software Maintenance (ICSM)*, pages 592–595. IEEE, 2012

In addition to this study, recent work conducted by Erik Kouters for his Master's thesis [161] under the co-supervision of Bogdan Vasilescu shows that the proposed approach scales very well to datasets an order of magnitude larger, with a clear improvement over existing approaches as the size of the data set grows. This work is presented in the following publications not included in this dissertation:

[161] Erik Kouters. Identity matching and geographical movement of open-source software mailing list participants. Master's thesis, Eindhoven University of Technology, 2014

[162] Erik Kouters, Bogdan Vasilescu, and Alexander Serebrenik. Who's who on GNOME mailing lists: Identity merging on a large data set. In *12th Belgian-Netherlands Software Evolution Seminar (BeNeVol)*, pages 33–34, 2013

Chapter 8: Diversity among software engineering conferences. In this chapter, we move away from diversity aspects in online software communities and instead turn to diversity aspects in scientific conference communities. Specifically, we address **RQ5** with a study of the health of 11 software engineering conferences, monitored over a period of more than 10 years. We show that MSR techniques can be successfully applied to studying diversity among software engineering conferences, by integrating data from multiple repositories, and introducing a suite of metrics that measure stability of a conference community, openness to new authors, introversion, representativeness of the program committee with respect to the authors' community, availability of program committee candidates, and scientific prestige. In general, we find that software engineering conferences are healthy, with some notable differences between conferences with a wide scope and those with a more narrow scope. This chapter consists of the following journal article:

[322] Bogdan Vasilescu, Alexander Serebrenik, Tom Mens, Mark G. J. van den Brand, and Ekaterina Pek. How healthy are software engineering conferences? *Science of Computer Programming*, 89, Part C:251–272, 2014

The data set assembled during this work has been published separately, as:

[321] Bogdan Vasilescu, Alexander Serebrenik, and Tom Mens. A historical dataset of software engineering conferences. In *International Working Conference on Mining Software Repositories (MSR), Data Track*, pages 373–376. IEEE, 2013

Chapter 9: Conclusions. This final chapter concludes the thesis. It revisits the research questions and gives directions for future research.

The research for the publications that constitute Chapters 3, 4, 5, 6, and 8 was conducted by Bogdan Vasilescu as first author. First authorship is shared with Mathieu Goeminne on the journal article [320] that makes up Chapter 2, and with Erik Kouters on the conference paper [163] that constitutes Chapter 7.

1.4.1 Other publications

In addition to the publications listed above, most of which are included in this dissertation, we have also published several other papers over the course of this PhD project, as listed below.

[326] Bogdan Vasilescu, Stef van Schuylenburg, Jules Wulms, Alexander Serebrenik, and Mark G. J. van den Brand. Continuous integration in a social-coding world: Empirical evidence from GitHub. In *International Conference on Software Maintenance and Evolution (ICSME), ERA Track*. IEEE, 2014

[332] Shaowei Wang, David Lo, Bogdan Vasilescu, and Alexander Serebrenik. EnTagRec: An enhanced tag recommendation system for software information sites. In *International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2014

[101] Mohammad Gharehyazie, Daryl Posnett, Bogdan Vasilescu, and Vladimir Filkov. Developer initiation and social interactions in OSS: A case study of the Apache software foundation. *Empirical Software Engineering*, 2014

[113] Georgios Gousios, Bogdan Vasilescu, Alexander Serebrenik, and Andy Zaidman. Lean GHTorrent: GitHub data on demand. In *International Working Conference on Mining Software Repositories (MSR), Data Track*, pages 384–387. ACM, 2014

[244] Gregorio Robles, Laura Arjona-Reina, Bogdan Vasilescu, Alexander Serebrenik, and Jesús M. González-Barahona. FLOSS 2013: A survey dataset about free software contributors: Challenges for curating, sharing, and combining. In *International Working Conference on Mining Software Repositories (MSR), Data Track*, pages 396–399. ACM, 2014

[230] Daniel Pletea, Bogdan Vasilescu, and Alexander Serebrenik. Security and emotion: Sentiment analysis of security discussions on GitHub. In *International Working Conference on Mining Software Repositories (MSR), Challenge Track*, pages 348–351. ACM, 2014

[66] Yanja Dajsuren, Christine M. Gerpheide, Alexander Serebrenik, Anton Wijs, Bogdan Vasilescu, and Mark G. J. van den Brand. Formalizing correspondence rules for automotive architectural views. In *International ACM Sigsoft Conference on Quality of Software Architectures (QoSA)*, pages 129–138. ACM, 2014

[317] Bogdan Vasilescu, Vladimir Filkov, and Alexander Serebrenik. Crowdsourced knowledge catalyzes software development. In *12th Belgian-Netherlands Software Evolution Seminar (BeNeVol)*, pages 60–62, 2013

[258] Bram Schoenmakers, Niels van den Broek, Istvan Nagy, Bogdan Vasilescu, and Alexander Serebrenik. Assessing the complexity of upgrading software modules. In *Working Conference on Reverse Engineering (WCRE)*, pages 433–440. IEEE, 2013

Chapter 2

Specialisation of labour

Most empirical studies of open source software repositories focus on the analysis of isolated projects, or restrict themselves to the study of the relationships between technical artifacts. In contrast, we have carried out a case study that focuses on the actual contributors to software ecosystems, being collections of software projects that are maintained by the same community. To this aim, we defined a new series of workload and involvement metrics, as well as a novel approach—$\widetilde{\mathbf{T}}$-graphs—for reporting the results of comparing multiple distributions. We used these techniques to statistically study how workload and involvement of ecosystem contributors varies across projects and across activity types, and we explored to which extent projects and contributors specialise in particular activity types. Using Gnome *as a case study we observed that, next to coding, the activities of localization, development documentation and building are prevalent throughout the ecosystem. We also observed notable differences between frequent and occasional contributors in terms of the activity types they are involved in and the number of projects they contribute to. Occasional contributors and contributors that are involved in many different projects tend to be more involved in the localization activity, while frequent contributors tend to be more involved in the coding activity in a limited number of projects.*

2.1 Introduction

Since the early 2000s, empirical studies aiming to understand open source software development by mining software repositories have continued to gain popularity. Two main causes of this popularity are: the abundance of projects for which the entire history of all software artifacts can be freely analysed; and the growing popularity of the open source paradigm, even in industrial settings [33, 336].

In this chapter, we go beyond existing research in software repository mining by focusing on the *community* of contributors to a software *ecosystem*. In particular, we wish to get insights in the variation of workload across the contributors to the different projects that make up the ecosystem. All contributors need to communicate, interact and

collaborate in order to adapt and maintain the ecosystem and its constituent projects. However, some of these contributors are considerably more active than others, some contribute to multiple projects, and many are involved in different types of activities. The social interactions between open source contributors, as well as their degree of project participation have been reported repeatedly to influence software quality and complexity [25, 293]. Such information needs to be carefully and empirically analyzed in order to get a better understanding of how open source contributors interact as part of a large ecosystem built up from multiple interrelated projects.

Another important aspect that is largely unexplored in empirical analyses of software repositories is how contributors specialise themselves in a restricted number of activity types. As proposed by Robles, González-Barahona, and Merelo [248] and Hindle, Godfrey, and Holt [129], one can distinguish different activity types such as coding, development documentation, building, testing, and so on. Both German [98] and the GNOME developers themselves recognised the importance of non-coding activities for GNOME[1], as well as contributors specialising themselves in these activities:

> "GNOME Community Celebrates 10 Years of Software Freedom, Innovation and Industry Adoption: Since 1997, the GNOME project has grown from a handful of developers to a contributor base of coders, documenters, translators, interface designers, accessibility specialists, artists and testers numbering in the thousands." [335]
>
> "Just on this note, let me state that I in no way consider translators as second-class citizens; nor documenters, UI dudes, general organisers, or anyone whatsoever just because they don't code." [283]

This is why we use the GNOME ecosystem as a case study in this chapter. The aim of this case study is to explore the variation in workload of projects and contributors of GNOME, taking into account the activity types they are involved in.

This chapter is structured as follows. Section 2.2 presents our two research goals and explains the research methodology followed. We introduce a novel set of metrics to study the variation of workload and involvement, and we present $\widetilde{\mathbf{T}}$-graphs as a novel approach to report the results of comparing multiple distributions. Sections 2.3 and 2.4 report on the statistical evaluation carried out for each research goal, and discuss the results. Section 2.5 presents the threats to validity, Section 2.6 reviews related work, Section 2.7 discusses future work, and Section 2.8 concludes.

2.2 Methodology

2.2.1 Research Goals

Following Lungu, Lanza, Gîrba, and Robbes [188] we define a *software ecosystem* as "a collection of software *projects* that are developed and evolve together in the same environment". Accompanying this notion of ecosystem we define its *ecosystem community* as the "collection of all *contributors* to the projects in the software ecosystem".

As mentioned above, we believe that studying the *contributors* to a software ecosystem (its ecosystem community) is equally important as studying the *contributions* to the ecosystem themselves. Therefore, we focus on participation of individual contributors,

[1] www.gnome.org

and study variations of the amount of participation across projects of the ecosystem and across contributors of the ecosystem community. For this reason, we explore the following two research goals:

1. How does workload vary across projects of the software ecosystem?
2. How does workload vary across contributors to the software ecosystem?

We decided to use the term *workload* as an objective measure of the amount of participation. Its formal definition will be given in section 2.2.5. As explained in the introduction, and as observed in an earlier exploratory study [199], the workload of projects or contributors may vary a lot depending on the *type of activity* that is being considered. Therefore we will take the type of activity into account while studying both research goals.

Section 2.3 will study the first research goal, and Section 2.4 will study the second research goal. It is in these sections that we will formulate the research questions and how they contribute to each goal.

2.2.2 Selected case study

In order to address the research goals we need to select as a case study a software ecosystem with at least the following characteristics:

- it should have a long development history (at least several years);
- it should possess a large *ecosystem community* involving many different contributors;
- its contributors should be active in other activity types besides coding;
- it should contain a large number of projects, many of which should still be actively maintained today;
- the projects should be *open source* as it facilitates data extraction and replication of our research results;
- the ecosystem should be well-known to researchers and open source developers.

We have selected the GNOME ecosystem as a case because it satisfies all of these requirements. The GNOME community develops a popular free and open source desktop environment for GNU/Linux and UNIX-type operating systems. In total, GNOME contains 1358 projects, 699 of which (*i.e.*, 51.5%) belong to the `archived` category, and 4 belong to the `deprecated` category.[2] Each of the GNOME projects has a corresponding Git distributed source code repository containing all information about the evolution history of the project. We only considered a subset of 1316 GNOME projects (including 691 archived projects) due to technical reasons: some of the Git repositories were not available at the time of extraction, some of the extractions did not produce any results, and some of the projects did not contain any committers. The lifetime of the considered projects varies widely. Some of the GNOME projects (*e.g.*, `gnome-disk-utility`) have started in 1997 and are still evolving today (corresponding to a lifetime of 15 years), others (*e.g.*,

[2]These values were computed on October 28, 2011, based on the project list available at `git.gnome.org/browse`. The number of GNOME projects has increased since this date.

	authors	committers	commits	files
min	1	1	1	25
Q1	3	2	23	61
med	12	9	131	112
Q3	59	46	517	237
max	1142	692	35191	7097
mean	62.07	45.78	760.2	252.3

Table 2.1: Variation of Git project characteristics across 1316 GNOME projects. For each project, the number of **commits**, **committers** and **authors** was computed for the entire considered project history. The number of **files** was computed for the last considered commit only.

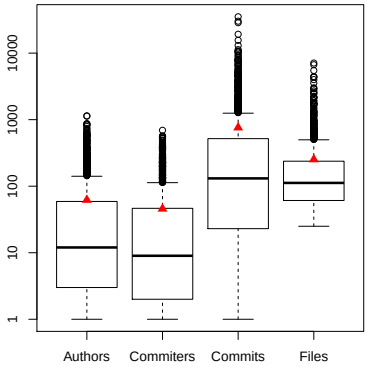


Figure 2.1: Boxplots showing the variation of Git characteristics from Table 2.1 (log y-axis). Triangles show the mean value.

`gnome-contacts`) were created more recently and were merely a couple of months old at the moment we extracted the data. In addition, many of the GNOME projects (over 900 of them) appeared to be inactive recently, their latest commit dating before 2011. This is in particular the case for most (but not all) projects belonging to the `archived` category.

The research goals of Section 2.2.1 use the notion of *contributor* belonging to the ecosystem community. Since all GNOME projects are stored in a Git repository, we will use the technical Git terminology to refer to a specific kind of contributor. Git makes an explicit distinction between a project *committer* and a project *author*. The *committer* is the person that has the right to commit files to the version repository. The *author* is the person that actually made the changes to the committed files. The reason for this distinction is that, for ease of management, an author does not always have commit rights, implying that his changes need to be committed by a different person. We will from now use the term *author* instead of *contributor*, to reflect the fact that we restrict our case study to only those persons that contribute to the Git project repositories of GNOME[3].

To extract relevant data from these Git repositories, we used CVSAnaly[4], a specialized tool able to populate a database having a particular structure [247]. For each project, we created and populated a database containing its entire change history (from its very beginning until September 2011) at file level. For each project commit, the database contains the date of creation of the commit, the committer name, the author name, and the files touched in the commit. A *file touch* corresponds to any action carried out on a file by its author: addition, removal, modification, copy or rename. Within a single commit, the same file can only be touched once. All files belonging to the same commit are touched by the same author.

Table 2.1 shows how some project characteristics vary across GNOME projects. For each project we have retrieved the number of committers, number of commits, number of authors and number of files (that latter values are computed only for the latest commit

[3]If we were to consider other data sources, such as mailing lists and bug trackers, we would be able to study other types of contributors as well.

[4]`metricsgrimoire.github.com/CVSAnalY`

retrieved for each project). Based on these values we computed the median, minimum, maximum, lower quartile (Q1), upper quartile (Q3) and mean values. The boxplots in Figure 2.1 visualise the distribution of these results.

2.2.3 Identity matching

One of the challenges when studying software ecosystems containing many different projects stored in different version control repositories, and communities involving a large number of contributors, is identity matching. The same author can use different names when contributing to different projects (*e.g.*, 'A S Alam' and 'Amanpreet Singh Alam'). Within the same project, an author may use different names (*e.g.*, 'Gabor Keleman' and 'Gabor Kelemen'), or even different types of names (*e.g.*, the name 'Yaakov Selkowitz' and the login 'yselkowitz'). Since we wish to study the specialisation of authors contributing to a software ecosystem, we need a unique identity representing the same author across all projects, even if the author has used different names or logins. To achieve this, we need to use a name matching algorithm.

Several identity matching algorithms have been proposed in literature [49]. Such algorithms either compute a similarity measure for each pair of names (*e.g.*, based on the Levenshtein distance or on phonetic encoding), attempt to match names and logins adhering to known naming conventions (*e.g.*, 'dmitrym' and 'Dmitry Mastrukov'), or use additional information to aid in the matching process (*e.g.*, GPG key servers[5] to determine coupled e-mail addresses [245]). However, all known existing approaches produce false positives (names that are incorrectly matched to the same identity) and false negatives (different names that correspond to the same author but for which no match is identified). Moreover, name structure and format are often influenced by project-specific, ecosystem-specific or community-specific rules and constraints (*e.g.*, in the GNOME Git repositories we found several name aliases corresponding to the name of an author prefixed by a timestamp in different formats, such as '(13:16)' or '23:32:57 BST'), thus increasing the risk of misclassification when blindly applying automated identity matchers.

Certain matching algorithms use other data sources (*e.g.*, mailing lists) to facilitate the matching. However, a preliminary study of two GNOME projects, *brasero* and *evince*, has shown a significant overlap in contributors per project, in the version repository, mailing lists and bug tracker [107]. Therefore, mailing lists and bug trackers were not considered in the current study and we decided to rely only on the data extracted from the Git code repositories.

To overcome these challenges we combine automatic identity matching with a manual postprocessing phase to reduce the number of false positives and false negatives. The different steps are schematised in Figure 2.2. First, GNOME-specific naming artifacts, such as the timestamp prefixes in different formats, are identified by manual inspection (step 0.1), then the author names are automatically preprocessed to remove these prefixes (step 0.2).

Next, a list of candidate matches is computed for each name (step 1.1) using a number of similarity measures[6] based on pattern matching (*e.g.*, the Levenshtein distance [177]), phonetic encoding (*e.g.*, soundex [154]), or a combination of both (*e.g.*, editex [357]). [49] provides an overview of such similarity measures. Although numerous name similarity

[5] GNU Privacy Guard, a free implementation of the OpenPGP standard for public key encryption.

[6] The implementations of the similarity measures are part of Febrl – a parallel open source data linkage system [50].

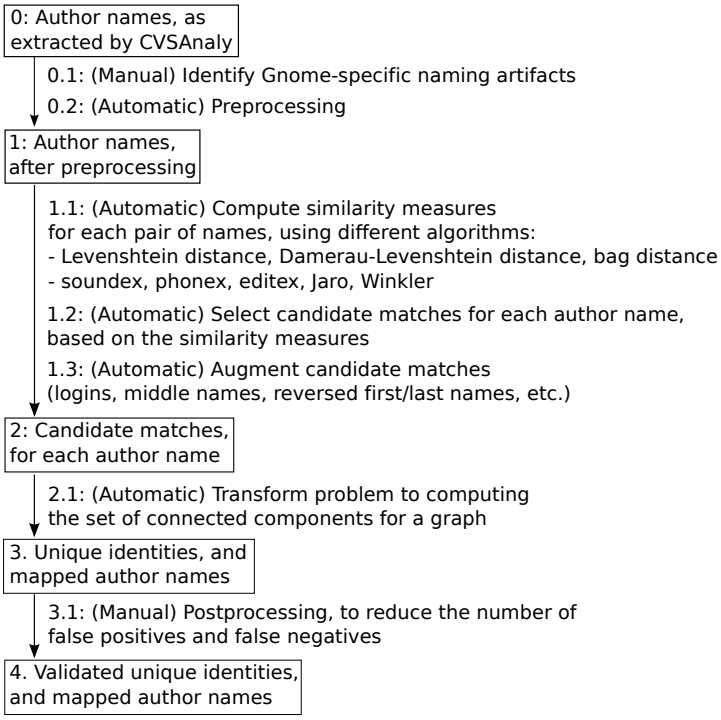


Figure 2.2: Identity matching steps.

measures exist and have available implementations, *e.g.*, as part of Febrl [50], computing such measures is often computationally expensive. Moreover, there is no single best technique available. Therefore, we applied the Bertillonage approach proposed by Davies, German, Godfrey, and Hindle [71] to reduce the search space using fast techniques, followed by more expensive computations on this reduced data set. Applying this to identity matching, we started by using a subset of the available similarity measures and only then performed a manual postprocessing.

From the set of available similarity measures of Christen [49] we required a limited subset (to ensure fast computation), well-balanced in terms of complementary types of similarity measures (pattern matching, phonetic encoding, combinations of both, or more advanced measures), and containing techniques that have been shown to perform well in practice. In this sense we selected the Levenshtein distance, Damerau-Levenshtein distance, and bag distance pattern matching techniques, the soundex and phonex phonetic encoding techniques, the editex combined measure, and the Jaro and Winkler data linkage algorithms. All of these algorithms are presented in detail by Christen [49], who also shows experimentally that they perform well on real name data sets.

A name is considered a candidate match (step 1.2) when at least one of the selected similarity measures exceeds a certain threshold. For a given similarity measure and a given name, the higher the threshold, the fewer the candidate matches and, conversely, the lower the threshold, the more the candidate matches for that name. We observed that a threshold value of 0.8 offered a good tradeoff between the number of candidate

matches and the number of false positives. If needed, the threshold can be changed, since it only impacts the amount of manual postprocessing required.

Similarity measures are sensitive to the ordering of name parts (*e.g.*, 'Attila Hammer' and 'Hammer Attila') and to the presence of middle names or initials (*e.g.*, 'Lars R. Clausen' and 'Lars Clausen'). They also fail to recognize as candidate matches login names corresponding to the same identity, even when logins are formatted according to commonly-adopted naming conventions, such as the first letter of the first name followed by the last name. Step (1.3) extends the list of candidate matches to incorporate these cases automatically.

Similarity measures are not necessarily transitive. In step (2.1), in order to have a complete list of candidates, we represent the candidate match relation as a graph in which names are nodes, and there is an edge between two nodes if one of them is a candidate match for the other. The sets of aliases used by the same authors is therefore the set of connected components of the graph.

In the manual postprocessing step (3.1) one of the authors of this paper matched the remaining logins, not adhering to commonly-adopted naming conventions (*e.g.*, 'mrhappypants'), to existing names by searching on the internet for email addresses used in common both by the nicknames (logins) and the existing names. Another author independently checked all matches, without being aware of which matches were suggested by the algorithm or by the manual postprocessing.

Applying the above identity matching approach to the data about the GNOME contributors allowed us to quantify the scale of the problem: without name matching, we found 6982 different author names across all considered GNOME projects. After name matching, only 5155 unique identities remained (*i.e.*, 73.83%). When counting the number of different names associated to these unique identities, we found a median value of 1, a mean value of 1.355, and a maximum value of 168. In fact, in 4344 cases (*i.e.*, 84.26%) the unique identities correspond to a single name. In 555 cases (*i.e.*, 10.77%) the identities correspond to two different names. The remaining 4.97% unique identities correspond to persons that have used 3 or more different names to identify themselves. The maximum number of aliases (168) corresponded to an author that used commit messages instead of his name.

In the remainder of this chapter, whenever we use the term *author*, we refer to the unique identities obtained as a result of the identity matching process.

2.2.4 Identifying activity types

The type of development activity carried out by authors in a project can be estimated on the basis of the types of files that are touched during each commit in the project's version control repository. To this end, we collect fully qualified paths, including the directory hierarchies, and the names and extensions of the files that have been touched for each commit in the version control history of each project.

Our approach is similar to that of Robles *et al.* [248] and Hindle *et al.* [129], who distinguish between different activity types based on the file names and extensions. Hindle *et al.* [129] distinguished between four types of files: source, test, build and documentation. Robles *et al.* [248] proposed 8 different activity types. We expand upon the classification by Robles *et al.* [248] by considering additional types such as *testing*, *database* and *library*.

In total, we defined 14 activity types. *Documentation* (`doc`) helps the final user in getting acquainted with the application. *Image* (`img`) refers to all picture files

Activity type t	Acronym	**Regular expression** e
Code	code	.*\.cpp
Development documentation	devdoc	.*/changelog.*
Documentation	doc	.*\.man, .*/doc(s?)/.*, .*/copyright
Images	img	.*\.jpg
Localization	l10n	.*\.po(~?), .*/locale(s?)/.*
Multimedia	media	.*/media(s?)/.*, .*\.mid

Table 2.2: Excerpt of the rules (t, e) used to identify the activity types from the file paths and file names. The regular expressions follow the traditional POSIX Basic Regular Expression syntax [137]. Backslash \ is used as escape character distinguishing between `.` representing any character and `\.` representing the character 'dot', *i.e.*, `.*\.cpp` represents all files with the `.cpp` extension. Forward slash / is used as a directory separator in file paths, *i.e.*, `.*/doc(s?)/.*` represents all files in `doc` and `docs` subdirectories. The complete list of rules is given in Appendix A.

used as part of the software project (*e.g.*, button icons, illustration in documentation). *Localization* (`l10n`) consists in adapting the software for other cultures, and includes translation activities. *User interface* (`ui`) is concerned with providing a graphical user interface to interact with the application. Files associated to *multimedia* (`media`) contain sounds, videos, and other multimedia resources (excluding images, which are categorized separately) that are used in the software. *Code* (`code`) files describe the software logic, whereas *test* (`test`) files contain the instructions needed to automatically test this logic. Files pertaining to the *meta* (`meta`) activity type are not a direct artifact of the projects, but support the software development process. *Configuration* files (`config`) are used by developers to describe some project properties, whereas *build* (`build`) files are used to help the developers and/or users to build a binary from the available resources. The *development documentation* (`devdoc`) aims to help persons involved in the project's development to maintain and improve the system. Files attached to the *database* (`db`) activity are used by the application as knowledge management resources. *Library* (`lib`) files contain third-party software. The last activity type, labelled `unknown`, contains all files not contained in any of the previous activity types.

Using information extracted from the file paths and file names, we iteratively build a collection of rules mapping files to activity types. Initially, the collection is empty and no files are mapped to activity types. For each file that has not been associated yet with an activity type, we add a pair (t, e), where e is a case-insensitive regular expression matching the fully qualified file path, and t is the corresponding activity type. For example, (`code`, `.*\.c`) is a rule specifying that any file with extension `.c`, regardless of its file path, corresponds to a `code` activity (since it is a C source code file). More examples of rules can be found in Table 2.2, while the complete set of rules can be found in Appendix A.

To define the regular expressions we have used domain knowledge. For instance, programming languages have traditional extensions for their source code files (*e.g.*, `.java` for Java programs), hence these extensions can be used as regular expressions associated with the `code` activity type. Other examples of commonly used naming conventions are the use of file paths, such as `/library/`, or parts of file names, such as `copyright`, to provide an indication of the corresponding activity type (in this case `lib` and `doc`, respectively).

In order to resolve situations where multiple regular expressions may be applicable to the same file, the rules are treated as an ordered list. Thus, the last rule that matches the file will be used to classify the file under the associated activity type. For example, a file `/test/ClassTest.java` matches the rules (`code`, `.*\.java`) and (`test`, `.*/test.*\..*`). Because the rule for the `test` activity type is checked after the one for `code`, `/test/ClassTest.java` will be associated with a `test` activity.

Files for which none of the regular expressions are applicable are classified as `unknown`. Examples of such files include (i) container files (having the extension `.zip`, `.rar`, etc.), (ii) files having an ambiguous, unusual or no extension, as well as files having no specific name or a non-specific path. Since files pertaining to the `unknown` activity type have little in common, we choose not to present and discuss their results during our analysis. However, in order not to distort our data, we do include `unknown` files in the computation of all our metrics.

Note that the above approach for classifying files per activity type is restrictive: files cannot be associated with multiple activity types since this would pose problems in the definition of some metrics and the statistical analysis and interpretation of some results. In some cases, multiple classification would have been useful. For example, `/test/ClassTest.java` could be classified as `test` because it presumably contains unit tests as well as `code` because Java test files are also source code files. Another limitation of our naive approach is that we classify files in a particular activity type based on the file path and file extension. This is not always sufficient. To determine if a file is really a test file, for example, one would need to parse the file's contents. For more details, we refer to Zaidman *et al.* [351] who used open source repository mining to study the co-evolution of production code and test code. A more refined classification and treatment of files per activity types is beyond the scope of this chapter.

2.2.5 Metrics

Having defined the research goals, and having selected GNOME as a case study, we now present a novel set of metrics that we have created to be able to answer the research questions for each research goal in Sections 2.3 and 2.4. These metrics are somehow restricted by the type of data that we can extract from the different GNOME project repositories in reasonable time. We decided to focus on file-level metrics as the most suitable level of granularity for our case study. While analyzing data below file level would allow us to be more precise, it turns out to be too time-consuming and resource-consuming. In addition, the file contents is only useful for text-based files such as code files, while a more in-depth analysis of code files would necessitate the use of different parsers (one for each language used). Ignoring files by studying commits would be too coarse grained, as it does not allow us to approximate the workload of individual authors at a sufficient level of detail. In particular, it does not allow us to identify the different activity types carried out by authors (see Section 2.2.4), while this is a prerequisite for addressing the second research goal.

Let P be the set of all GNOME projects, A the set of all unique GNOME authors (*i.e.*, the GNOME contributors after matching different logins to the same identity), T the set of all considered activity types. Each file belonging to some commit in the version control repository of a project $p \in P$ can be directly linked to an author $a \in A$ that touched this file, and the type $t \in T$ of the activity corresponding to this file is computed as explained in section 2.2.4.

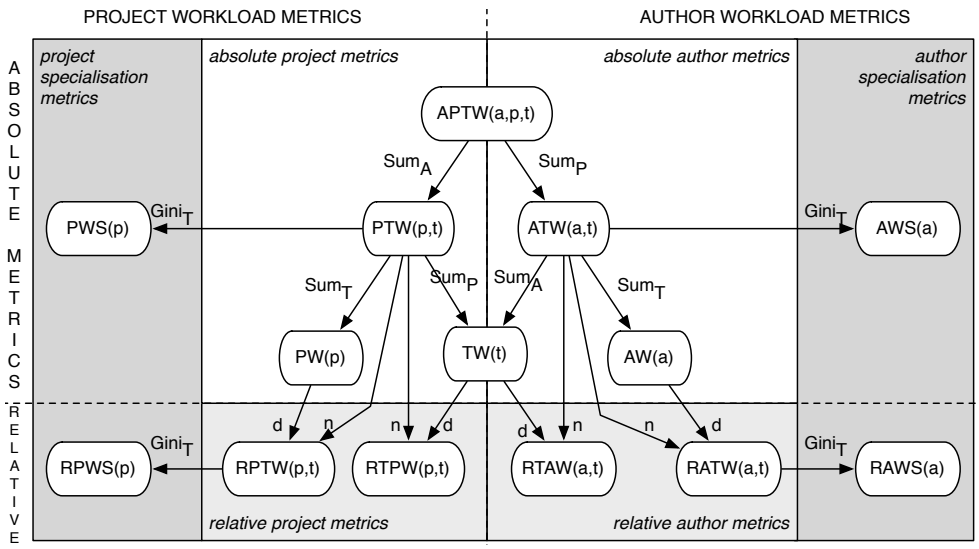


Figure 2.3: Workload metrics. The following naming convention is adopted for the metric acronyms: **A** = Author; **P** = Project; **T** = activity Type; **W** = Workload; **R** = Relative; **S** = Specialisation. Relative metrics are defined as a fraction $\frac{n}{d}$ and represent a percentage (*i.e.*, a value between 0 and 1). $Gini_T$ denotes the application of the *Gini* inequality index [102] over all activity types. Similarly, Sum_T, Sum_A and Sum_P aggregate values through summation.

The basic metric we compute using data extracted from the Git logs is the **Author-Project-Type Workload** $APTW$:

$$APTW(p,a,t) = \text{number of touches to files of activity type } t \text{ by author } a \text{ for project } p \text{ over its entire history.} \quad (2.1)$$

If the same file is touched in different commits, it will be counted multiple times. Based on this metric, we can also derive the **Author-Project-Type Involvement** $APTI$ that determines for project p if an author a has been involved in at least one (*i.e.*, has touched at least one file of) activity type t:

$$APTI(p,a,t) = \begin{cases} 1, & \text{if } APTW(p,a,t) > 0; \\ 0, & \text{otherwise.} \end{cases} \quad (2.2)$$

Using these two basic metrics, we can derive higher-level aggregate metrics. Figure 2.3 presents the workload metrics that are derived from $APTW$, while Figure 2.4 presents the involvement metrics that are derived from $APTI$. In both figures we distinguish between project-level metrics (on the left) and author-level metrics (on the right).

The way these metrics are computed is similar. We therefore only present the project-level metrics definitions in Tables 2.3 and 2.4. The main distinction between workload metrics and involvement metrics is that the latter rely on counting. For example, $NAP(p)$ counts how many authors are involved in project p. If the same author is involved in different activity types for this project, she needs to be counted only once. This explains

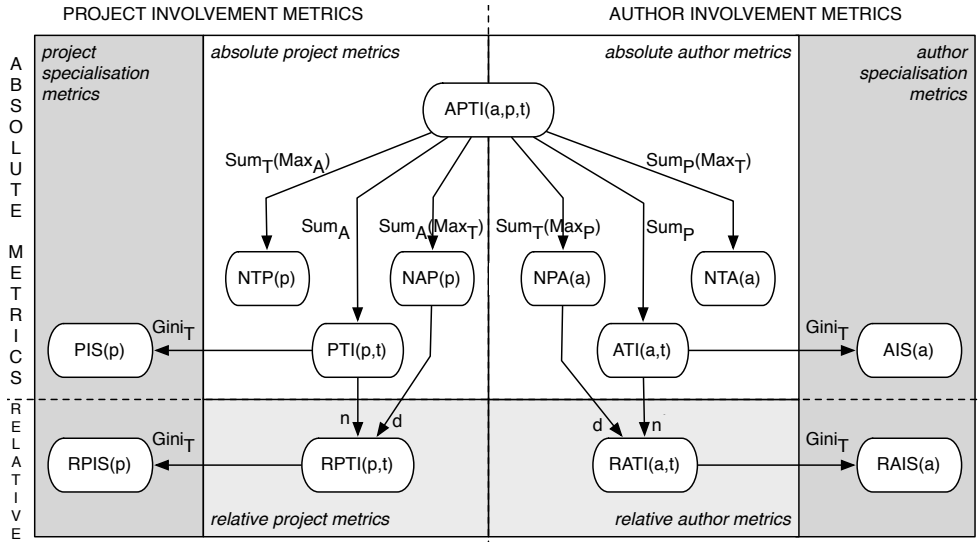


Figure 2.4: Involvement metrics. The same naming convention is followed as in Figure 2.3, except that we now use **I** for *involvement* and **N** for *number of*.

why we first compute the maximum over all types, and then compute the sum over all authors.

Acronym	Description	Definition
$PTW(p,t)$	(absolute) project-type workload	$\sum_{a_j \in A} APTW(p, a_j, t)$
$PW(p)$	project workload over all authors and activity types	$\sum_{t_k \in T} PTW(p, t_k)$
$TW(t)$	type workload over all authors and projects	$\sum_{p_i \in P} PTW(p_i, t)$
$RPTW(p,t)$	workload in project p for activity type t, relative to the total project workload	$\frac{PTW(p,t)}{PW(p)}$
$RTPW(p,t)$	workload in project p for activity type t, relative to the total type workload	$\frac{PTW(p,t)}{TW(t)}$
$PWS(p)$	specialisation (imbalance) of workload across activity types for project p, over all authors contributing to p	$Gini_{t_k \in T}(PTW(p, t_k))$
$RPWS(p)$	specialisation (imbalance) of relative workload across activity types for project p, over all authors contributing to p	$Gini_{t_k \in T}(RPTW(p, t_k))$

Table 2.3: Definitions of $APTW$-based project-level workload metrics. (The author-level workload metrics are defined similarly.)

Tables 2.3 and 2.4 and Figures 2.3 and 2.4 also refer to *specialisation* metrics that need some further explanation. To quantify the degree of specialisation of authors (towards a particular activity type), as well as the degree of project specialization (towards a particular activity type), we rely on the *Gini* inequality index [102]. We have used it to define

Acronym	Description	Definition
$PTI(p,t)$	(absolute) project-type involvement	$\sum_{a_j \in A} APTI(p, a_j, t)$
$NTP(p)$	number of types for project p	$\sum_{t_k \in T} max_{a_j \in A} APTI(p, a_j, t_k)$
$NAP(p)$	number of authors for project p	$\sum_{a_j \in A} max_{t_k \in T} APTI(p, a_j, t_k)$
$RPTI(p,t)$	author involvement in project p for activity type t, relative to the total number of authors involved in the project	$\frac{PTI(p,t)}{NAP(p)}$
$PIS(p)$	specialisation (imbalance) of involvement across activity types for project p, over all authors contributing to p	$Gini_{t_k \in T}(PTI(p, t_k))$
$RPIS(p)$	specialisation (imbalance) of relative involvement across activity types for project p, over all authors in p	$Gini_{t_k \in T}(RPTI(p, t_k))$

Table 2.4: Definitions of $APTI$-based project-level involvement metrics. (The author-level involvement metrics are defined similarly.)

the project specialisation metrics PWS, $RPWS$, PIS and $RPIS$, as well as the author specialization metrics AWS, $RAWS$, AIS and $RAIS$ by aggregating over all activity types in T. The Gini inequality index [102] is one of the many inequality indices commonly applied in econometrics to study inequality of income or welfare distributions [58, 59]. As opposed to traditional aggregation techniques such as mean or median, inequality indices provide reliable results for highly-skewed distributions. Similarly to such traditional aggregation techniques, inequality indices do not require complex application procedures. The Gini index is defined based on the Lorenz curve [184], and ranges between 0 and $1 - \frac{1}{n}$ [10], where n is the number of values being aggregated (*e.g.*, $n = 14$ in the case where we aggregate over all activity types). We could have chosen other measures of inequality [58, 207, 262, 294], but Gini has been shown to convey the same information as the other applicable inequality indices [323, 324].

Finally, we have chosen to focus on the specialisation of authors and projects towards a particular activity type. Alternatively, one could have studied specialisation of authors towards a particular project $Gini_{a \in A}(\sum_{t \in T} APTW(p, a, t))$ (similar to the project Work Concentration measure of Tsay *et al.* [301]) or specialisation of projects towards a particular author $Gini_{p \in P}(\sum_{t \in T} APTW(p, a, t))$.

2.2.6 Data analysis

In order to facilitate replication of our case study, we have created a webpage and a replication package[7] containing the data, tooling, and detailed results of the statistical analysis performed. In this section we briefly introduce the techniques we have used to perform statistical analysis. We relied on the R project for statistical computing [238], including packages such as `ineq` to calculate the Gini index [352], `Matching` to perform the bootstrapped Kolmogorov-Smirnov test [261], `agricolae` to determine the Kendall

[7]The data set can be found here: `www.win.tue.nl/mdse/gnome`

correlation coefficient [72], and `nparcomp` to compute relative contrast effects when comparing two distributions [157].

Correlation When measuring statistical correlation between two groups of data we have a choice between linear or rank correlation coefficients. Linear coefficients (*e.g.*, Pearson [226]) are sensitive only to a linear relation between two variables. Rank coefficients [149, 278] are more robust to nonlinear relations since they only measure the extent to which an increase in one variable (not necessarily linear) corresponds to an increase in the other variable. Since we do not make assumptions about the shape of each relation, we use a rank coefficient and we opt for Kendall's τ since Spearman's ρ is known to be difficult to interpret [219]. We account for ties as described by Press *et al.* [237]. Whenever we measure Kendall correlation between two metrics, the null hypothesis H_0 is that there is no relation between the two metrics, and the alternative hypothesis H_a is that there is a relation between the two metrics. We report Kendall's τ and the corresponding p-value.

Linear regression When a linear relation between the dependent variable and one or more independent variables can be suspected, we also perform linear regression, *i.e.*, based on the data we estimate parameters of the linear function of the independent variables to obtain as close values as possible to the values of the dependent variable. To check the adequateness of the fitted model we analyze the residual plot: the points in the residual plot should appear randomly dispersed around the horizontal axis. Moreover, we report the p-values for the significance of regression with the F-statistic, as well as p-values for the coefficients and the intercept. Finally, we report the *adjusted* coefficient of determination $\bar{R}^2$ [295, pp. 164,175–178] that takes into account the number of parameters used by the regression model.

Distribution fitting In order to understand how data values are distributed, we try to fit a theoretical distribution to it. Specifically, as many distributions in software follow a *power law* $x^{-\alpha}$ [185] or are *log-normal* [22, 181], in this chapter we only attempt to fit these types of distributions. To evaluate the goodness-of-fit of a log-normal distribution we use the Kolmogorov-Smirnov test. The original test cannot calculate correct p-values in presence of ties. In those cases we use the bootstrapped version of the Kolmogorov-Smirnov test [261] instead. In this test we use the two-sided alternative hypothesis and the default number of bootstraps to be performed (1000). Considering a power law distribution, it often applies only for values greater than some minimum value, so we need to estimate this value in addition to α that determines the form of the distribution. Using the methodology proposed by Clauset, Shalizi and Newman ([53]) we estimate the aforementioned parameters and calculate the goodness-of-fit between the data and the power law. If the resulting p-value is lower than the threshold of 0.1 proposed by Clauset [53], we reject the hypothesis that the distribution follows a power law. If the p-value is higher than 0.1, it is possible that other distributions can be fitted as well. Therefore, we have to compare the likelihood of the data under two competing distributions. Depending on the families these distributions belong to, we either exploit the closeness test of Vuong [329] or a slightly modified likelihood ratio test [53].

Excluding zeros As part of our research goals we study the influence of the activity type (*e.g.*, coding, localization) on workload variations across projects and across project contributors. To this end we distinguish between, and compute metrics per, different activity types. Whenever we compute a metric that takes the activity type into account, we consistently exclude zero values; for each activity type, we only focus on the projects (contributors) that contain (participate in) activities of that type, cf. discussion of active committers of Robles *et al.* [248]. The only exception is when we compare specialisation of projects and contributors in few activity types (*i.e.*, Figures 2.8 and 2.16), computed using the Gini index. In these cases, since not all projects contain, and not all contributors participate in, activities of all types, we *do not* exclude zero values (*e.g.*, for a project we do not ignore the activity types not present in that project), since this would lead to incomparable Gini index values.

2.2.7 $\widetilde{\mathbf{T}}$ procedure and $\widetilde{\mathbf{T}}$-graph

When studying the specialisation of projects and authors towards different activity types, we need to assess whether the distributions of a given metric are different for the different activity types. Traditionally, comparison of multiple groups follows a two-step approach: first, a global null hypothesis is tested, and then multiple comparisons are used to test sub-hypotheses pertaining to each pair of groups. The first step is commonly carried out by means of ANOVA or its non-parametric counterpart, the Kruskal-Wallis one-way analysis of variance by ranks [133]. The second step uses the t-test or the rank-based Wilcoxon-Mann-Whitney test [340], with Bonferroni correction [80, 268]. Unfortunately, the global test null hypothesis may be rejected while none of the sub-hypotheses are rejected, or vice versa [95]. Moreover, simulation studies suggest that the Wilcoxon-Mann-Whitney test is not robust to unequal population variances, especially in the unequal sample size case [355]. Therefore, one-step approaches are preferred: these should produce confidence intervals which always lead to the same test decisions as the multiple comparisons.

Moreover, since we have identified 13 different activity types[8], we had to conduct $\frac{13*12}{2} = 78$ comparisons and report 78 results. For the sake of brevity we summarize the test results as a directed acyclic graph. Nodes of the graph correspond to activity types, edges to results of pairwise comparisons. Because plotting a graph with 13 nodes and in the worst case 78 edges would result in visual clutter, we would like to omit direct edges between A and B if there is a path from A to B passing through at least one other node C. Hence, we need an approach that respects transitivity. Unfortunately, this is not necessarily the case for traditional pairwise or multiple comparison approaches: *e.g.*, Brown and Hettmansperger [39] show that no transitive reduction is possible for the traditional pairwise Wilcoxon-Mann-Whitney tests. Transitivity is, however, respected by the recently proposed multiple contrast test procedure $\widetilde{\mathbf{T}}$ [158]. Moreover, $\widetilde{\mathbf{T}}$ is robust against unequal population variances.

The $\widetilde{\mathbf{T}}$ procedure takes as input a type of *contrast* and the threshold for the family-wise error rate, *i.e.*, the probability of falsely rejecting one or more null sub-hypotheses [168] (we use the traditional threshold of 5%). The $\widetilde{\mathbf{T}}$ procedure returns an estimator for the difference of each pair of the distributions being compared, the corresponding 95% confidence interval, test statistics and the corresponding p-values.

[8] As explained in Section 2.2.4 we do not include the `unknown` activity type.

Contrasts, represented as the contrast matrix, express which sub-hypotheses should be tested. Formally, matrix $\mathbf{C}$ is called a contrast matrix if $\mathbf{C} \cdot \mathbf{1} = \mathbf{0}$, where $\mathbf{1}$ is the column vector of appropriate length consisting solely of ones and $\mathbf{0}$ is the row vector consisting solely of zeroes, *i.e.*, the sum of all rows in $\mathbf{C}$ is 0 [41]. To illustrate the notion of a contrast matrix consider the following matrices:

$$C_D = \begin{pmatrix} -1 & 1 & 0 & \dots & 0 \\ -1 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ -1 & 0 & 0 & \dots & 1 \end{pmatrix} \quad C_T = \begin{pmatrix} -1 & 1 & 0 & \dots & 0 & 0 \\ -1 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -1 & 0 & 0 & \dots & 0 & 1 \\ 0 & -1 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & -1 & 0 & \dots & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -1 & 1 \end{pmatrix}$$

Matrix C_D expresses comparisons of multiple alternative hypotheses (treatments) with a specific one (control group) and is known as "many-to-one" or Dunnett-type contrast [81]. Matrix C_T expresses all pairwise comparisons (up to symmetry) and is known as "all pairs" or Tukey-type contrast [303]. Since our goal is to compare all groups pairwise, we consider only Tukey-type contrasts.

Next we introduce $\widetilde{\mathbf{T}}$-graphs, a new and more intuitive visualisation that we propose for reporting the results of the $\widetilde{\mathbf{T}}$ procedure:

- First, for each pair of groups we analyse the 95% confidence interval to test whether the corresponding null sub-hypothesis can be rejected. If the lower boundary of the interval is greater than zero for groups A and B, then we claim that the metric value is higher in A than in B. Similarly, if the upper boundary of the interval is less than zero for groups A and B, then we claim that the metric value is lower in A than in B. Finally, if the lower boundary of the interval is less than zero and the upper boundary is greater than zero, we conclude that the data does not provide enough evidence to reject the null hypothesis.

- Second, based on the results of the comparisons we construct the graph with nodes being groups and containing edges (A, B) if the metric value is higher in A than in B. After removal of transitive edges [8], we obtain a directed acyclic graph that we call a $\widetilde{\mathbf{T}}$-graph.

A visual comparison of multiple distributions using $\widetilde{\mathbf{T}}$-graphs enables us to focus on "interesting" groups, *e.g.*, activity types located "high" in the graph, *i.e.*, those activity types with metric values higher than most of the remaining activity types, or "low" in the graph, *i.e.*, those activity types with metric values lower than many remaining activity types.

To illustrate the $\widetilde{\mathbf{T}}$ procedure and a $\widetilde{\mathbf{T}}$-graph consider the following artificial example inspired by and extending the $drug_1$ data of [9]. Figure 2.5 (left) shows the commit activity per activity type (A, B, C or D) for a group of twenty developers: *e.g.*, developer #1 has performed two commits for activity A, one commit for activity B, one commit for activity C and one commit for activity D. Using the $\widetilde{\mathbf{T}}$ procedure and a $\widetilde{\mathbf{T}}$-graph we would like to clarify the relationship between the four activity types. We start by invoking

Activity type	Developers																			
A	2	2	2	3	3	3	3	3	3	3	4	4	4	4	4	4	4	4	5	5
B	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2
C	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	3	3	
D	1	1	1	2	2	2	2	2	2	2	3	3	3	3	3	3	4	4	4	4

Pair	Lower	Upper	p-value
B–A	-0.560	-0.444	0.000
C–A	-0.503	-0.313	7.536e-10
D–A	-0.320	-0.027	1.997e-02
C–B	-0.014	0.242	9.742e-02
D–B	0.237	0.470	1.200e-06
D–C	0.090	0.404	2.432e-03

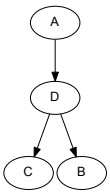


Table 2.5: Illustration of $\widetilde{\mathbf{T}}$ procedure and $\widetilde{\mathbf{T}}$-graph based on artificial data. Left: Commit activity per type for 20 developers (columns). Middle: Results of the $\widetilde{\mathbf{T}}$ procedure. The p-value reported as zero is too small to be calculated exactly. Right: The resulting $\widetilde{\mathbf{T}}$-graph.

the $\widetilde{\mathbf{T}}$ procedure for the Tukey-type contrast and 95% confidence level. Results of the $\widetilde{\mathbf{T}}$ procedure are summarized in Figure 2.5 (middle). For five out of six comparisons the $\widetilde{\mathbf{T}}$ procedure reports $p < 0.05$ or, equivalently, the corresponding 95% confidence interval does not contain zero. Since the lower boundary of the confidence interval for D–B and D–C is greater than zero, the corresponding graph should contain edges from D to B and from D to C. Similarly, since the upper boundary of the confidence interval for B–A, C–A and D–A is smaller than zero, the corresponding graph should contain edges from A to B, A to C and A to D. After removal of transitive edges we obtain the $\widetilde{\mathbf{T}}$-graph with three edges shown in Figure 2.5 (right).

A special case of comparison of multiple distributions is the comparison of two distributions. We need to test whether one of two samples of independent observations tends to have larger values than the other. Traditionally, distributions of software metrics have been compared using the Wilcoxon-Mann-Whitney two-sample rank-sum test [13, 151]. However, Wilcoxon-Mann-Whitney is not robust against differences in variance [40, 355]. The $\widetilde{\mathbf{T}}$ procedure as described above cannot be applied to comparison of two distributions [158]. We therefore prefer the two-distributions equivalent of the $\widetilde{\mathbf{T}}$ procedure, *i.e.*, we perform two sample tests for the nonparametric Behrens-Fisher problem [40], and compute confidence intervals for the relative effect of the two samples. If the relative effect $p(a, b) > 0.5$ then b tends to be larger than a. Moreover, since software metrics are frequently being compared using the Wilcoxon-Mann-Whitney two-sample rank-sum test [13, 151], we also report the results of this test.

2.3 Project-centric view

Our first research goal consists in *understanding how workload varies across projects belonging to the same ecosystem.* In order to address this goal we study cross-project variation of measurable project-level properties (*e.g.*, project workload PW, number of authors involved in a project NAP, number of activity types per project NTP) by answering the following research questions:

1. How does project workload vary across the ecosystem?
2. Which types of projects are more active?

3. How specialised are projects towards different activity types?
4. What are the characteristics of specialised projects?

2.3.1 How does project workload vary across the ecosystem?

We start by studying the variation of the project workload $PW(p)$ across the ecosystem, for all $p \in P$. The distribution is left-skewed and the maximal value is more than an order of magnitude larger than the mean: two features typical for *heavy-tailed* distributions [291]. We first hypothesise that the project workload follows a power law. This hypothesis can be rejected since the p-value of the goodness-of-fit test equals 0.0496, which is lower than the threshold of 0.1 [53]. Next, we consider the log-normal distribution. Since the data contains ties we opt for the bootstrapped Kolmogorov-Smirnov test [261]. The corresponding p-value equals 0.533, and, hence, the hypothesis that the project workload follows the log-normal distribution cannot be rejected. A histogram of $\log PW(p)$ is presented in Figure 2.5.

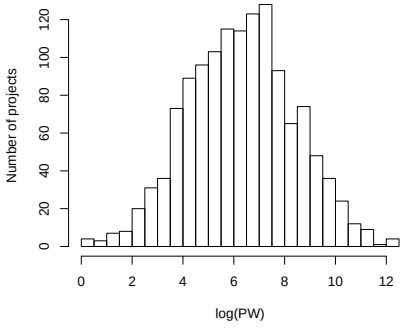


Figure 2.5: The workload $PW(p)$ is distributed log-normally.

Project workload is distributed log-normally across the software ecosystem.

Figure 2.5 also reveals exceptional projects. At the lower end of the scale we distinguish archived projects, and projects with very little activity. Further inspection of the commit logs and GNOME mailing list archives revealed that since some of the latter modules have not seen any recent activity or are closed in the issue tracker for new bug entries, they are likely to be archived soon as well. This was for example the case for *gnome-audio*, that had very little activity until October 2011 (the latest date considered in our case study) and is indeed listed as archived in October 2012. Other projects with small workload either have incomplete repositories, potentially as a result of migration from CVS to Git (*e.g.*, *O3web*), or are auxiliary (*e.g.*, *perl-Clutter* which, although stand-alone, represents only a set of Perl bindings for Clutter 1.x). At the higher end of the scale we distinguish very active projects such as *GIMP*, the GNU image manipulation program, or *Evolution*, the email, contacts and scheduling manager.

2.3.1.1 Which projects are more active?

2.3.1.2 Are projects containing more activity types more active?

To study this first question, we compare the number of activity types per project $NTP(p)$ and the project workload $PW(p)$. With a Kendall correlation test we observe a strong correlation ($\tau = 0.6$), and reject H_0 (p-value $< 2.2 \times 10^{-16}$). Closer inspection of the scatter plot in Figure 2.6 suggests a linear relation between $NTP(p)$ and $\log PW(p)$. Using linear regression we obtain the model $\log PW(p) = 0.64562 \cdot NTP(p) + 1.57412$ ($\bar{R}^2 = 0.6129$). The fitted linear model is adequate: F-statistic equals 2109 on 1 and 1314 degrees of freedom with the corresponding p-value $< 2.2 \times 10^{-16}$, p-values for the coefficient and the intercept do not exceed 2.2×10^{-16}. The points in the residual plot appear randomly dispersed around the horizontal axis. We conclude that the project activity increases exponentially (due to the use of $\log PW$ in the formula) as projects include more activity types: increasing the number of activity types by one increases the effort almost twice ($e^{0.64562} \simeq 1.9$).

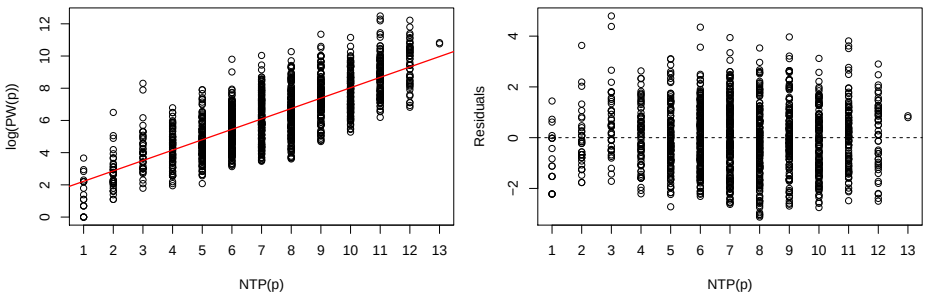


Figure 2.6: Left: observed linear relation between $NTP(p)$ and $\log PW(p)$ (regression line drawn). Right: residuals plot.

The more activity types a project contains, the more active it is: increasing the number of activity types by one approximately doubles the project workload.

Figure 2.6 also reveals exceptional projects, being either very diverse (*e.g.*, the *Anjuta* integrated development environment and the *Banshee* music player both contain activities of all 13 types considered), or very specialised (*e.g.*, *GTK tutorial* is an archived project associated with the GTK toolkit for creating graphical user interfaces, and contains only documentation activities, while *O3web* is an archived project containing only build activities). The 18 projects containing activities of a single type are all categorised as *archived*.

2.3.1.3 Are projects with larger communities more active?

Does the number of authors $NAP(p)$ involved in project p influence the total workload $PW(p)$? As a result of Kendall's correlation test we observe a strong correlation between $NAP(p)$ and $PW(p)$ ($\tau = 0.64$) and reject H_0 (p-value $< 2.2 \times 10^{-16}$), suggesting that project workloads are higher as more authors are involved in the projects. We do not describe the relation between $NAP(p)$ and $PW(p)$ further since we could not obtain

adequate linear regression models, for which the points in the residual plot would appear randomly dispersed around the horizontal axis.

The larger its community of contributors, the more active the project.

We observed some exceptional projects. For example, 218 projects (16.56%) are developed by a single author. Among them we find projects such as *GSAPI* and *GSpeech* (variations on the Java Speech API), which eventually became *archived* and were refined into *Gnome Speech*, which has a larger community of 15 authors. There are also non-archived projects developed by a single author. For example, *Grits*, a Virtual-Globe-like library that handles coordinates and the OpenGL viewport, is still actively maintained today by a single developer.

2.3.2 How specialised are projects towards different activity types?

Let us first explore how the ecosystem workload varies across the different activity types. Figure 2.7 displays this variation using the type workload $TW(t)$ aggregated over all projects. We observe a high inequality between the different activity types. `code`, `devdoc`, `l10n`, and `build` account for the highest shares of the ecosystem workload, representing together 78% of the total workload. All `code` activities by themselves account for more than 40% of the total workload.

Across the ecosystem, `code`, `devdoc`, `l10n`, and `build` activities account for the highest share of the workload. `code` is the predominant activity type.

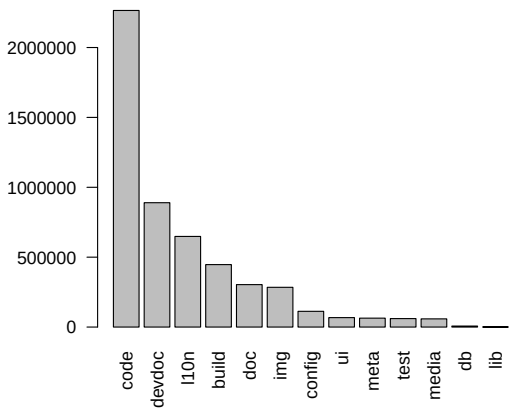


Figure 2.7: Type workload: activity types `code`, `devdoc`, `l10n`, and `build` account for the highest workload share in the ecosystem.

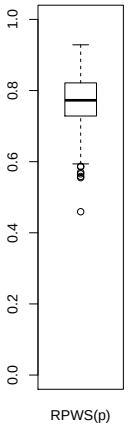


Figure 2.8: Relative project workload specialisation RPWS(p): Most projects concentrate their workload in few activity types.

2.3.2.1 To what extent are projects specialised in few activity types?

The specialisation $RPWS(p)$ of a project p can be interpreted as how a project p specialises its relative workload $RPTW(p,t)$ towards few activity types t. It is computed

by applying the Gini index to aggregate the $RPTW(p,t)$ values over all types $t \in T$. A high value of $RPWS(p)$ reflects a high inequality in the distribution of workload across the different activity types for project p. This suggests that most of the project's workload is concentrated in few activity types, while the remaining activity types only account for a very small fraction of the workload. A low value of $RPWS(p)$ reflects a more equal distribution of the project's workload across the different activity types.

Figure 2.8 displays the variation across projects of $RPWS(p)$. We observe that most projects are specialised in few activity types, since the median is high (0.77). Our observation is similar to the findings of Vasa *et al.* [311], according to which the typical overall range of Gini coefficients for multiple software metrics is between 0.45 and 0.75, thus values above 0.75 can be considered high. The highest values of $RPWS(p)$ have been observed for projects such as *O3web* that focus on one activity only. For these projects $RPWS(p)$ reaches the highest theoretically possible value for Gini coefficient on a data set of 14 elements, *i.e.*, $\frac{13}{14} \simeq 0.93$. The lowest value of $RPWS(p)$ is 0.459, *i.e.*, it still belongs to the [0.45, 0.75] range of Gini coefficients observed by Vasa *et al.* [311]. The lowest value of $RPWS(p)$ has been observed for *gnome-applets*, the project that distributes the activity in a most egalitarian way. *Gnome-applets* is a collection of small unrelated applications for the GNOME desktop, including various monitors, weather report, trash bin and eyes following the mouse pointer around the screen. The second lowest $RPWS(p)$ value (0.555) was obtained for *gnome-utils*, another collection of small unrelated desktop applications. Closer inspection of the $RPTW$ values for *gnome-applets* and *gnome-utils* reveals that both projects have a relatively high share of the `build` activity: 13% and 11%, respectively. This can be explained by the fact that one should be capable of compiling separately individual applications comprising *gnome-applets* and *gnome-utils*, implying that each one of the application has its own makefile and related files. Moreover, since *gnome-applets* and *gnome-utils* comprise desktop applications, a relatively high part of the effort is dedicated to `l10n` and `image`. All this leads to relatively egalitarian workload distribution, reflected in relatively low $RPWS$ values.

Most of a project's workload is concentrated in few activity types.

Note that not each activity type is present in each project (*e.g.*, not all projects contain `db` activities). However, since we are interested in comparing specialisation for different projects (*i.e.*, comparing Gini index values, computed for $RPTW(p,t)$ data over all activity types $t \in T$), we consider for each project the set of all possible activity types. As explained in Section 2.2.6, we do not ignore the activity types t for which $RPTW(p,t) = 0$ when computing $RPWS(p)$ since this would render Gini index values incomparable.

2.3.2.2 To what extent are projects specialised towards different activity types?

Workload. The specialisation of a project p towards a certain activity type t can be expressed in terms of the relative project's *workload* $RPTW(p,t)$, defined as the workload in project p for activity type t relative to the total workload in p. High $RPTW(p,t)$ values reflect that most of the workload of project p is concentrated in t, *i.e.*, t is a predominant activity type in p in terms of number of file touches. Similarly, low $RPTW(p,t)$ values reflect that activities of type t are but auxiliary in p.

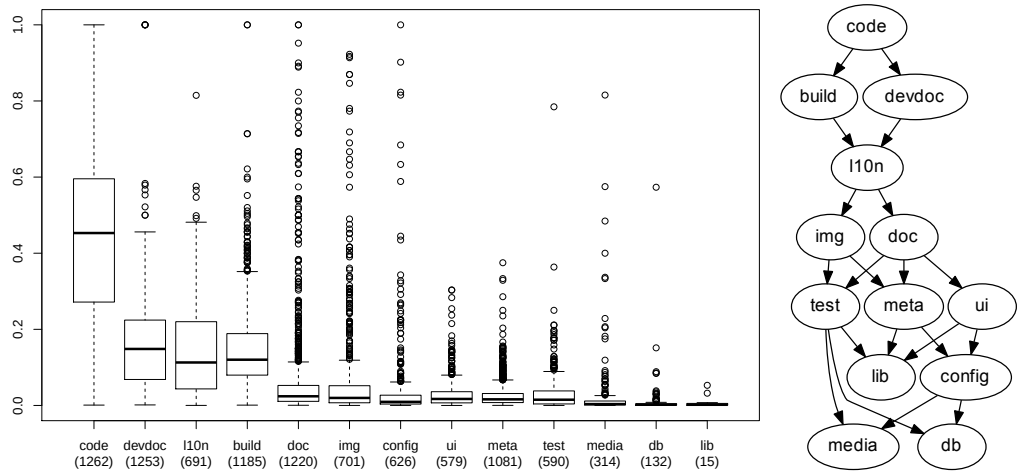


Figure 2.9: Boxplots for relative project workload per activity type t. Per boxplot, zero values are excluded. `code` is the predominant activity type at project level: in most projects that contain coding, it represents around 50% of the workload. `devdoc`, `l10n`, and `build` each account for 10-20% of the workload on average. The $\widetilde{\mathbf{T}}$-procedure with respect to Tukey-type contrasts and 5% family-wise error rate shows differences between the activity types in the $\widetilde{\mathbf{T}}$-graph on the right (cf. Section 2.2.6).

Figure 2.9 illustrates the variation across projects of $RPTW(p,t)$ for each activity type t. In each boxplot, we only consider the projects for which activity type t is present (*i.e.*, the workload $PTW(p,t) > 0$), since we are only interested in understanding how the workload varies for projects that contain activities of that type. The number of projects (out of the total 1316 projects considered) for which $PTW(p,t) > 0$ is displayed below each activity type in the boxplot.

We observe two groups of activity types. On the one hand, `code`, `build`, `devdoc`, and `l10n` (the same four main activity types observed at ecosystem level, Figure 2.7) have the highest values, with `code` being the predominant one in the $\widetilde{\mathbf{T}}$-graph. Since the median for $RPTW(p, code)$ is slightly less than 0.5, we can say that in most projects that contain coding activities, coding represents around 50% of the workload. There are 54 (4.1%) projects that do not contain any `code` activities at all. Further manual investigation of the source code repositories and mailing list archives revealed that such projects without `code` activities are often auxiliary, *e.g.*, *Gnome Backgrounds*—a collection of desktop background images, or *Gnome Cookbook*—a cookbook used and developed by the GNOME community. On the other hand, `lib`, `media`, and `db` have the lowest $RPTW(p,t)$ values, all being leafs in the $\widetilde{\mathbf{T}}$-graph.

> At the level of individual projects, most of the workload is concentrated in `code`, followed by the `devdoc`, `l10n` and `build` activity types.

Workforce. Another way to express the specialization of a project towards a certain activity type t is in terms of the relative project's *workforce* $RPTI(p,t)$, defined as the number of authors of p involved in activity type t relative to the total number of authors in

p. High $RPTI(p,t)$ values reflect that most of the authors involved in p are contributing to activities of type t. Low $RPTI(p,t)$ values indicate that activities of types t that only attract a small fraction of the authors involved in p.

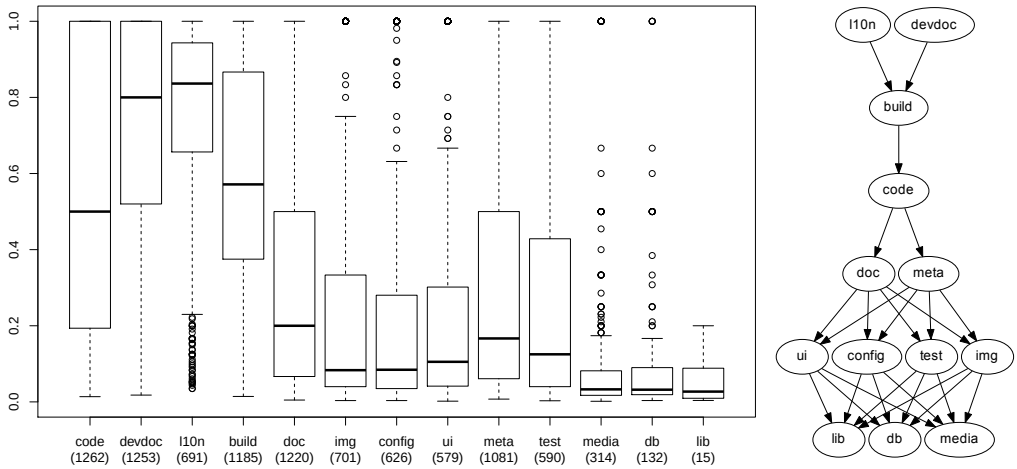


Figure 2.10: Boxplots for relative project involvement per activity type t. Per boxplot, zero values are excluded. `l10n` attracts the highest fraction of the project authors. `db` activities are performed only by a small fraction of the project authors. The multiple contrast test procedure $\widetilde{\mathbf{T}}$ with respect to Tukey-type contrasts and 5% family-wise error rate shows differences between the activity types in the directed acyclic graph on the right (cf. Section 2.2.6).

The variation of $RPTI(p,t)$ across projects per activity type t is illustrated in Figure 2.10. Similarly as before, we only consider the projects for which there is at least one author involved in activities of type t, *i.e.*, the involvement $PTI(p,t) > 0$ (their number is displayed below each activity type in Figure 2.10). On the one end of the spectrum we observe that `l10n` and `devdoc` (which encompasses updating the ChangeLog, a common practice of authors whenever they perform changes) attract most of the authors involved in projects (they are the dominant activity types in the $\widetilde{\mathbf{T}}$-graph), followed by `build` and only then `code`. On the other end of the scale we observe activity types such as `lib`, `db`, and `media` (the bottommost activity types in the $\widetilde{\mathbf{T}}$-graph), in which for most projects only a small fraction of the authors are involved.

> `l10n` and `devdoc` activities attract most of the authors involved in projects, followed by `build` and then `code`.

We illustrate the variation of $RPTI(p,t)$ across projects by taking a closer look at two projects, *Gevice*, GNOME's Network Device Manager, and *Evolution*, GNOME's contact manager, address manager and calendar. The $RPTI$ values of *Gevice* are extremely high: $RPTI(Gevice,t) = 1$ for all t but `lib`, `media`, and `test`, meaning that for any other activity 100% of the *Gevice* authors are involved in it. This is not surprising since *Gevice* has only a single author. As opposed to *Gevice*, the $RPTI$ values of *Evolution* are always lower than 1, *i.e.*, there is no activity that would attract all 723 *Evolution* authors.

2.3.3 What are the characteristics of specialised projects?

In order to understand the characteristics of highly-specialised projects, we study the correlation between metrics representing the specialisation of projects, *i.e.*, $RPWS(p)$, $PTW(p,t)$, $RPTW(p,t)$, $PTI(p,t)$ and $RPTI(p,t)$, on the one hand, and general project characteristics, *i.e.*, project workload $PW(p)$, number of authors involved in a project $NAP(p)$ and number of activity types per project $NTP(p)$, on the other hand.

2.3.3.1 Which project characteristics are observed when it is specialised towards few activity types?

In Section 2.3.2.1 we observed that, while some GNOME projects exhibit relatively low specialisation values (*e.g.*, 0.459 for *gnome-applets*), the opposite is true for other projects (*e.g.*, 0.93 for *O3web*). In this section we investigate which characteristics of a project influence its specialisation. We expect that projects with more workload (measured by $PW(p)$), as well as projects with larger communities (measured by $NAP(p)$) tend to be less specialised since more opportunities for diversity arise from higher workload and more authors. On the other hand, it is unclear whether more specialised projects consist of few activity types (which thus concentrate the workload), or consist of many activity types of which only few concentrate the workload. In order to answer the question we study Kendall correlation between $RPWS(p)$ and each one of the project-specific $PW(p)$, $NTP(p)$, and $NAP(p)$ metrics.

For $PW(p)$ we confidently reject H_0 at 0.01 significance level (p-value $= 1.26 \times 10^{-13}$). However, the correlation coefficient is very small and negative ($\tau = -0.13$), indicating a very weak relation between $RPWS(p)$ and $PW(p)$. For $NAP(p)$ we again confidently reject H_0 at 0.01 significance level (p-value $= 7.33 \times 10^{-37}$), and observe a small negative correlation ($\tau = -0.24$). We hence did not find conclusive evidence that projects with more activity or projects with larger communities tend to be less specialised.

Finally, we confidently reject H_0 at 0.01 significance level for $NTP(p)$ (p-value $= 6.85 \times 10^{-90}$) and observe a slightly higher negative correlation ($\tau = -0.39$). This suggests that the more activity types are present in a project, the lower the project's specialisation towards those activity types, as measured by $RPWS(p)$. It follows that highly unequal distributions of workload across different activity types are due to few activity types being present in a project and thus the project's workload being concentrated in those types, rather than many activity types being present in a project, with most of the project workload concentrated in one of these types.

> Highly specialised projects comprise few activity types (as opposed to many activity types out of which only few would concentrate the workload).
> In contrast, we have not found enough evidence that projects with more workload or larger communities are less specialised towards few activity types.

2.3.3.2 To what extent does project community size relate to the workload (share) for a particular activity type?

In Section 2.3.1.3 we observed that projects with larger communities have higher workloads. We wish to understand how the size of a project community (measured by $NAP(p)$) relates to the workload $PTW(p,t)$ and the workload share $RPTW(p,t)$ of this project generated for a particular activity type t.

To answer the question we compute Kendal correlation between $NAP(p)$ and $PTW(p,t)$ on the one hand, and between $NAP(p)$ and $RPTW(p,t)$ on the other hand, for all activity types $t \in T$. For each activity type t, we only look at projects for which $PTW(p,t) > 0$, as discussed in Section 2.3.2.2. The results of the correlation tests are visually summarised in Figure 2.11 for PTW (drawn in black) and $RPTW$ (drawn in gray). The shape and fill of a point represent the p-value of the correlation test, and determine whether H_0 can be rejected (*i.e.*, filled square: $p < 0.01$; empty square: $0.01 \leq p < 0.05$; empty circle: $p \geq 0.05$). The ordinate of a point represents the value of Kendall's τ coefficient.

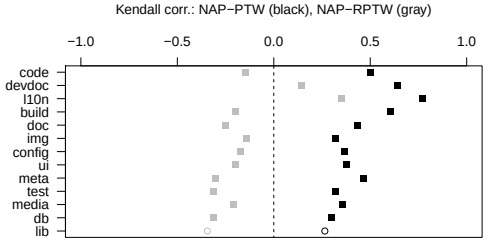


Figure 2.11: As more authors become involved in the projects, the workload increases the most in `l10n`, `devdoc`, `build`, and `code` (black). In terms of shares, it is the workload in *localization* that increases the most relative to those in other activity types (gray).

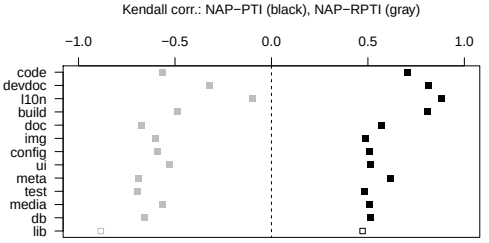


Figure 2.12: As more authors become involved in the projects, they mostly contribute to `l10n`, `devdoc`, `build`, and `code` (black). The percentage of developers involved in `l10n` does not decrease as more developers become involved in the projects (gray).

For $PTW(p,t)$ we reject H_0 at 0.01 confidence level for all activity types except `lib`. Due to insufficient projects that contain `lib` activities (only 15), H_0 cannot be rejected for this activity type even at 0.05 confidence level. We observe the strongest correlation for the four main activity types, `l10n` ($\tau = 0.77$), `devdoc` ($\tau = 0.64$), `build` ($\tau = 0.60$), and `code` ($\tau = 0.50$). This suggests that as more authors are involved in the projects, the project workload is higher in these activity types.

> Projects with more authors correspond to more absolute workload in `l10n`, `devdoc`, `build`, and `code` than in other activity types.

For $RPTW(p,t)$ we again reject H_0 at 0.01 confidence level for all activity types except `lib`, for which H_0 cannot be rejected even at 0.05 confidence level. As opposed to $PTW(p,t)$, correlation is now negative for all activity types except `devdoc` and `l10n`, and is low for all activity types. `l10n` shows the strongest positive correlation (0.35), suggesting that as more authors are involved in a project, it is the share of the workload in `l10n` that is the highest most relative to the workload in other activity types. The positive correlation for `devdoc` is also due to the authors contributing to `l10n`, since as they perform `l10n` activities, they often also update the ChangeLog, which is part of `devdoc`. This observation generalises that of German [99], who reports similar co-updates of the ChangeLog for *Evolution*, one of the GNOME projects.

> Projects with more authors correspond to higher fractions of workload in `l10n` rather than other activity types.

2.3.3.3 To what extent does project community size relate to the involvement (share) of authors in different activity types?

We wish to understand whether the number of authors $NAP(p)$ involved in project p influences how the authors become involved in a particular activity type, in terms of their absolute involvement $PTI(p,t)$ and their involvement share $RPTI(p,t)$ for this project.

To answer the question we compute Kendall correlation between $NAP(p)$ and each of $PTI(p,t)$ and $RPTI(p,t)$, for all activity types $t \in T$. Figure 2.12 visually summarises the results of the correlation tests for PTI (drawn in black) and $RPTI$ (drawn in gray), with the same conventions as in the previous question.

For $PTI(p,t)$ we reject H_0 at 0.05 confidence level for `lib`, and at 0.01 confidence level for all other activity types. Similarly to the previous question, we observe the strongest correlation (now higher) for the four main activity types, `l10n` ($\tau = 0.88$), `devdoc` and `build` ($\tau = 0.81$), and `code` ($\tau = 0.70$). This suggests that as more authors are involved in the projects, they are involved mostly in these activity types.

> As more authors are involved in a project, most of them tend to be translators rather than coders.

For $RPTI(p,t)$ we again reject H_0 at the same confidence levels, and observe negative correlation for all activity types. `lib` shows now the strongest correlation ($\tau = -0.88$), suggesting that as more authors are involved in a project, the share of authors involved in this activity type is lower. This confirms that `lib` is the smallest activity type, and that it is performed by a limited number of developers. In addition, `l10n` shows the lowest correlation ($\tau = -0.09$), leading us to the following conclusion:

> The number of authors involved in a project is not related to the share of authors involved in localization activities.

Summarising the preceding discussions of Section 2.3.3, we observed the following relations between project characteristics and project specialisation. The more specialised a project, the less activity types are present in it. As more authors are involved in a project, they tend to be mostly translators and they generate a higher workload for the activity type `l10n`. However, we have found no evidence that higher project workload or larger project community are correlated to the overall specialisation values.

2.3.4 Summary for Goal 1

Our first research goal consisted in *understanding how workload varies across projects belonging to the same ecosystem*, taking into account the different types of activities performed within these projects.

First, we observed that project activity across the ecosystem follows a log-normal distribution. Next, we investigated what project properties are correlated to the project activity, and we found such a correlation for the number of activity types in which the developer community participates, and for the size of the community. Specifically, a project having a high number of activity types or having a large developer community is more active than a project having a small number of activity types or a small developer community.

By focusing on different activity types we observed that *coding*, *development documentation*, *localization*, and *build* are the four most important ones, at the ecosystem level as

well as at the level of individual projects. It is also these four activity types that attract most of the authors involved in projects. However, while it is *coding* that concentrates most of a project's workload, *localization* and *development documentation* attract most of the contributors.

Finally, we observed that most projects concentrate their workload in few activity types. We investigated the factors associated to this specialisation and found evidence that highly specialised projects are also projects including few activity types. Moreover, as projects contain more contributors, they are more commonly translators rather than coders.

2.4 Developer-centric view

Our second research goal consists in *understanding how workload varies across authors belonging to the same community.* In order to address this goal we study cross-author variation of measurable author-level properties (*e.g.*, author workload AW, number of projects NPA in which an author is involved, number of activity types per author NTA) by answering the following research questions in each of the next subsections:

1. How does workload vary across authors?
2. Which kind of authors are more active?
3. How specialized are authors towards different activity types?
4. What are the characteristics of specialised authors?

2.4.1 How does workload vary across authors?

We start by studying the variation of the author workload $AW(a)$ across all projects and activity types (Figure 2.13). As in the case of the project workload, distribution of the author workload is heavy-tailed and does not follow a power law: the p-value equals 0.0499 and does not exceed the recommended threshold of 0.1 [53]. As opposed to the project workload, hypothesis of log-normal distribution of the author workload can be rejected since the p-value corresponding to the bootstrapped Kolmogorov-Smirnov test [261] is lower than 2.2×10^{-16}.

Most authors have low workload. Few authors have high workload.

The heavy tail of the distribution of $AW(a)$ suggests a more refined analysis. To mitigate the potentially confounding effect of size, we distinguish between authors with low activity (occasional contributors) and authors with high activity (frequent contributors). Specifically, based on their $AW(a)$ values we apply equal-frequency binning and split the authors into two groups: $AW < 14$ and $AW \geq 14$. Figure 2.14 displays the breakdown of authors after binning.

Approximately half of the authors performed less than 14 file changes ($\log 14 \simeq 2.64$). In contrast, the most active author performed 185,874 file changes ($\log 185874 \simeq 12.13$).

This conclusion is concurrent with the observation by Neary and David [214] that the top 40 developers have made 31% of all changes, while the most prolific 5% of developers have made 65% of all changes.

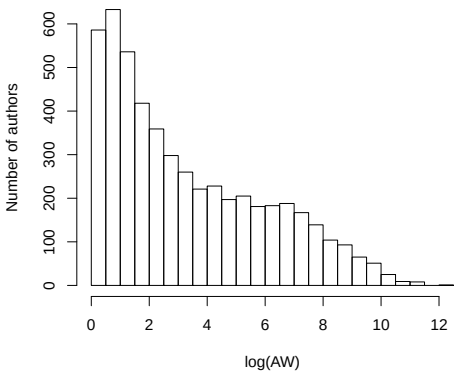


Figure 2.13: Distribution of author workload AW is heavy-tailed but does not follow a power law or log-normal distribution.

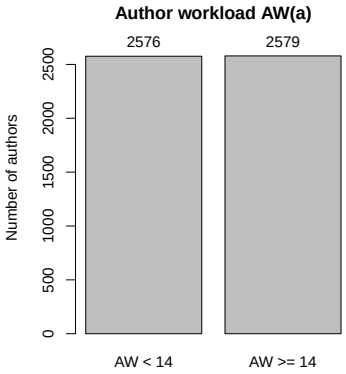


Figure 2.14: Approximately half of the authors performed less than 14 touches to GNOME files.

2.4.2 Which kind of authors are more active?

2.4.2.1 Are authors that participate in more activity types more active?

We first investigate whether the number of activity types $NTA(a)$ an author a contributes to across the ecosystem is related to her total workload $AW(a)$. As a result of Kendall's correlation test we confidently reject H_0 ($p < 2.2 \times 10^{-16}$) and observe a strong correlation between $NTA(a)$ and $AW(a)$ ($\tau = 0.737$).

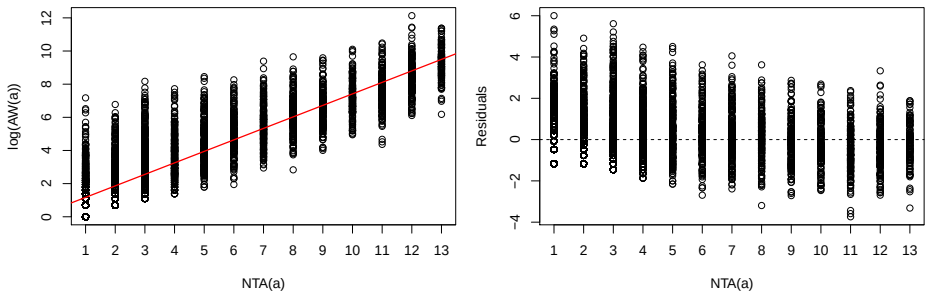


Figure 2.15: Left: observed linear relation between $NTA(a)$ and $\log AW(a)$ (regression line drawn). Right: residuals plot.

The scatter plot of Figure 2.15 suggests a linear relation between $NTA(a)$ and $\log AW(a)$. We obtain the following linear regression model: $\log AW(a) = 0.69326 \cdot NTA(a) + 0.47786$, with $\bar{R}^2 = 0.7971$. The fitted linear model is adequate: F-statistic equals 20030 on 1 and 5146 degrees of freedom with the corresponding p-value not exceeding 2.2×10^{-16}, p-values for the coefficient and the intercept do not exceed 2.2×10^{-16}. The points in the residual plot appear randomly dispersed around the horizontal axis. We conclude that the author activity increases exponentially (due to the use of $\log AW$ in the formula) as authors contribute to more activity types. Increasing the number of activity types by one doubles the effort ($e^{0.69326} \simeq 2$). Figure 2.15 also reveals that 1452 (28%)

authors are involved in a single activity type. In Section 2.4.3.2 we investigate in which activity types these authors specialise themselves.

The more activity types an author participates in, the more active she is: increasing the number of activity types by one doubles the workload.

2.4.2.2 Are authors that contribute to more projects more active?

How does the number of projects $NPA(a)$ an author a is involved in correlate to the total workload $AW(a)$ for that author? As a result of the Kendall correlation test we confidently reject H_0 ($p < 2.2 \times 10^{-16}$), and observe above average correlation ($\tau = 0.573$). This suggests that the author workload increases as authors become involved in more projects. We do not describe the relation between $NPA(a)$ and $AW(a)$ further since we could not obtain linear regression models for which the points in the residual plot would appear randomly dispersed around the horizontal axis, hence the linear models were not appropriate for the data.

The more projects an author contributes to, the more active she is.

2.4.3 How specialized are authors towards different activity types?

2.4.3.1 To what extent are authors specialised in few activity types?

We intuitively expect that authors are mostly specialised in few activity types, similar to what we observed for the specialisation of projects in Section 2.3.2. The specialisation $RAWS(a)$ of an author a can be interpreted as how this author specialises her relative workload towards few activity types. It is computed by applying the Gini index to aggregate the $RATW(a,t)$ values over all types $t \in T$. Figure 2.16 displays the variation across authors of $RAWS(a)$, for the entire ecosystem community as well as for each of the two groups obtained after binning.

Note that different authors contribute to different activity types (*e.g.*, not all authors contribute to `test` activities). Since we are interested in comparing specialisation for different authors, we consider for each author the set of all possible activity types (*i.e.*, we do not ignore the activity types t for which $RATW(a,t) = 0$ when computing $RAWS(a)$).

Using the same equal-frequency binning for AW as in Section 2.4.1, we observe a clear distinction between the specialisation of occasional ($AW < 14$) and frequent ($AW \geq 14$) contributors. Since they contribute very few changes in total to the ecosystem, the occasional contributors are very specialised, more than the frequent contributors: the median for the occasional contributors group equals 0.9285, which is also the maximal value of the Gini index for populations of size 14, *i.e.*, $1 - 1/14$. By definition of the Gini index it follows that most of the occasional contributors participate in a single activity type. Note that the double usage of the value 14 is coincidental: in "$AW < 14$", 14 was the threshold found for AW as a result binning, while in "$1 - 1/14$", 14 refers to the number of activity types.

Our observation that the occasional contributors are specialised more than the frequent contributors is supported by statistical tests. The relative effect for (frequent,occasional) is 0.815 and the corresponding p-value is too small to be computed exactly. Since the relative effect exceeds 0.5, the specialisation values for the occasional contributors tend

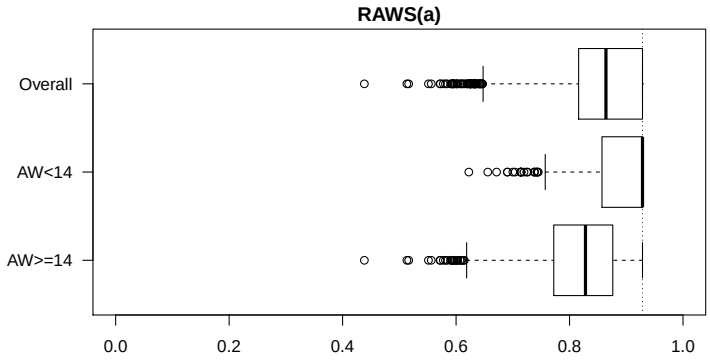


Figure 2.16: Relative author-workload specialisation $RAWS(a)$.

to be larger than those for the frequent contributors. The Wilcoxon-Mann-Whitney test allows us to derive the same conclusion, $p < 2.2 \times 10^{-16}$.

Even though less specialised, the frequent contributors also display a very high median of $RAWS(a)$ (0.82), even higher than the corresponding median of $RPWS(p)$ from Goal 1 (0.77, Figure 2.8). Therefore, there is high inequality in the distribution of workload across the different activity types for most of the frequent authors. Overall, we conclude that most of the authors' workload is concentrated in few activity types, while the remaining activity types only account for a small fraction of the workload.

> While contributing to different projects within the ecosystem, most authors concentrate their workload in few activity types. Moreover, occasional contributors typically participate in a single activity type.

2.4.3.2 To what extent are authors specialised towards different activity types?

Workload. The specialisation of an author a towards a certain activity type t can be expressed as the specialisation of her relative *workload* $RATW(a,t)$, *i.e.*, the total number of file touches that a performed for activity type t across the ecosystem relative to the the total number of file touches that a performed for all activity types across the ecosystem. High $RATW(a,t)$ values reflect that most of the workload of a across the ecosystem is directed towards activities of type t, *i.e.*, t is a predominant activity type for a. Low $RATW(a,t)$ values reflect that activities of type t are but auxiliary for a.

Figure 2.17 (top left) depicts the overall variation across authors of $RATW(a,t)$ for each activity type t, for all authors. As before, for each boxplot we only consider the authors that contribute to t (*i.e.*, the workload $ATW(a,t) > 0$), and display their number below each activity type. We observe the same outstanding activity types as in Figures 2.7 and 2.9, namely `code`, `devdoc` and `l10n`. Specifically, we observe that the third quartile for `l10n` coincides with 1, *i.e.*, approximately 25% of the translators (526 out of 2008) focus exclusively on `l10n`, corresponding to slightly more than 10% of the entire GNOME community. A similar finding has been reported for KDE by Robles *et al.* [248].

The $\widetilde{\mathbf{T}}$-graph on the right confirms that `code` is the dominant activity type, while `db` and `lib` have the smallest values. Therefore specialisation of authors towards certain

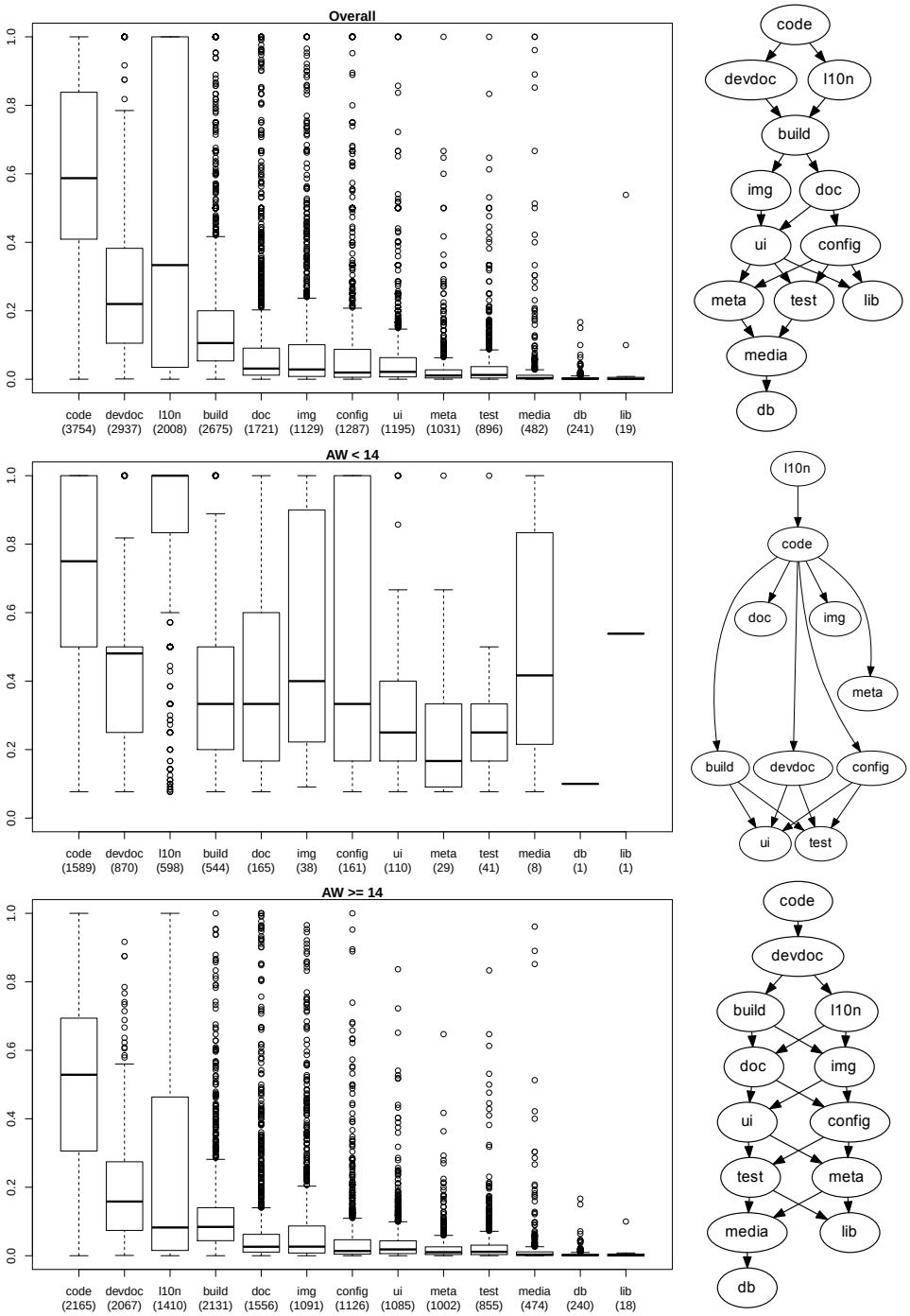


Figure 2.17: Boxplots for $RATW(a,t)$ per activity type t. Zero values are excluded. By definition of $RATW(a,t)$ ($AW(a)$ appears in the denominator), the lower whiskers in the $AW < 14$ boxplots cannot be lower than $1/13$. For $AW < 14$ the $\widetilde{\mathbf{T}}$-graph does not include `db` and `lib` since the $\widetilde{\mathbf{T}}$ procedure is not applicable for groups of size one.

activity types follows specialisation of projects, since most authors specialise in the four previously-observed main activity types. The predominance of `code` and `l10n` is also recognised by members of the ecosystem community in mailing list discussions:

> "[...] gnome *is* a code-centric organization. The coders are our sine qua non—without them, we have nothing. Translators are probably a close second to that—without them, we have no international coders. Past that, no group of people in the project are indispensable to the current state of the project, or even close to it." [328]

Most authors specialise in `code`, `l10n` and `devdoc` activities. Among those activity types, `code` is predominant.

Using equal-frequency binning we can obtain more fine-grained information for the occasional contributors ($AW < 14$) and the frequent contributors ($AW \geq 14$). We displayed both groups in the middle and bottom boxplots of Figure 2.17. Visual comparison of these sets of boxplots reveals a clear distinction in behaviour for the `l10n` activity: in the $AW < 14$ case, the median for $RATW(a, \texttt{l10n})$ is 1, while in the $AW >= 14$ case it is around 0.1. Indeed, statistical tests show that occasional contributors are specialised in localisation more than the frequent contributors. The relative effect for (frequent,occasional) is 0.907 and the corresponding p-value is too small to be computed exactly. Since the relative effect exceeds 0.5, the $RATW(a, \texttt{l10n})$ for the occasional contributors tend to be larger than those for the frequent contributors. The Wilcoxon-Mann-Whitney test allows us to derive the same conclusion, $p < 2.2 \times 10^{-16}$.

Since the overall median for $RATW(a, \texttt{l10n})$ is also relatively small ($\simeq 0.3$), the preceding discussion suggests that occasional contributors prefer to specialise in localisation rather than other activity types. This observation is supported by the $\widetilde{\mathbf{T}}$-graphs, in which we observe an inversion of the relation between `l10n` and `code` from $AW < 14$ to $AW \geq 14$: while `code` is the dominant activity type for frequent contributors, `l10n` is the dominant one for occasional contributors. On the other hand, the relations between `code` and `devdoc`, and between `code` and `build` are consistent.

For frequent contributors, `devdoc` and `build` remain the other two predominant activity types. For occasional contributors, although `img` or `config` might appear visually to have higher values, the data does not provide enough evidence to support this observation using the $\widetilde{\mathbf{T}}$-procedure (see $\widetilde{\mathbf{T}}$-graph).

Frequent contributors tend to specialise in the `code` activity, while occasional contributors tend to specialise in the `l10n` activity. For both types of contributors `devdoc` and `build` remain important activities.

A similar difference in behaviour between occasional and frequent GNOME contributors with respect to `code` and `l10n` has also been observed by Neu *et al.* [215].
The authors assume persons "who contributed a lot but only to a relatively small number of projects" to be coders (*i.e.*, the people located under a logarithmic-like curve in Figure 2.18–left), while those "who committed less often but to more projects" to be translators (*i.e.*, the people placed above an exponential-like curve in Figure 2.18–left). While this classification is only qualitative, it is confirmed by our quantitative analysis (Figure 2.18–right). In our case the y-axis also corresponds to the number of projects per author $NPA(a)$, while the x-axis corresponds to the author workload $AW(a)$ – expressed as number of file

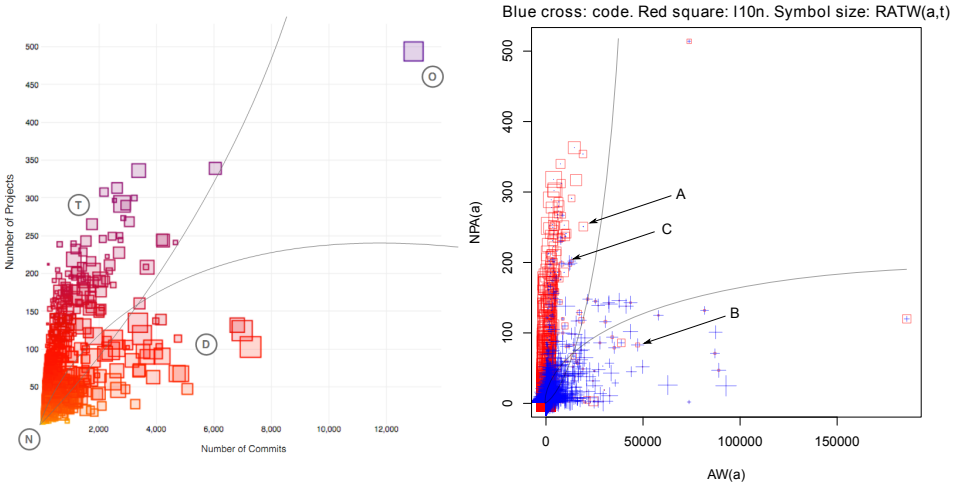


Figure 2.18: *Left*: visualization by Neu *et al.* [215]: each square depicts a committer; the number of projects is encoded both on the y-axis and in the colour of each square; the size of a square corresponds to the lifetime in days of a committer within GNOME; Ⓣ: translators, Ⓓ: developers, Ⓞ: outlier, Ⓝ: no man's land. Persons under the logarithmic-like curve are assumed to be coders, persons above the exponential-like curve are assumed to be translators. *Right*: Our own visualization.

touches per author, so more fine-grained than the number of commits per committer used by Neu *et al.* [215]. In the plot we overlay per author a the $RATW(a, \texttt{code})$ values (blue crosses) and $RATW(a, \texttt{l10n})$ values (red squares), where the size of each symbol encodes the $RATW$ value. Our results are consistent with those of Neu *et al.* [215]: coders are typically very active contributors involved in a relatively small number of projects, while translators are less active but are involved in more projects. Examples of potential misclassification following the qualitative approach include developer C, a coder with low activity but involved in many projects, or developer B, a contributor active in both `code` as well as `l10n`, having high activity but involved in relatively few projects. Mixed patterns of involvement in `code` and `l10n` (for which developer A is an example) are in line with the following excerpts from the mailing list discussions: while translators do not typically code, some start out with just translating, but continue with fixing bugs and then coding.

> "Furthermore, in GNOME, we have many translators that started out with just translating, but continued with fixing bugs, and some are full time coders now. We should be proud of this integration." [251]
>
> "As you have pointed out yourself, translators are usually not hackers/-coders." [250]

Involvement. The relative author involvement $RATI(a, t)$ is defined as the number of projects in which author a performs activities of type t relative to the total number of projects in which she is involved. High $RATI(a, t)$ values reflect that in most of the projects author a is involved in, she contributes to activities of type t. Similarly, low

$RATI(a,t)$ values reflect that a performs activities of type t only sporadically, *i.e.*, she only performs activities of type t in a small fraction of the projects she is involved in.

For all authors, the variation of $RATI(a,t)$ per activity type t is illustrated in Figure 2.19 (top). Zeros are again ignored. The median value of 1 for the `code`, `devdoc`, and `l10n` activity types signifies that, once authors are involved in these activity types, they perform the same activities in all projects they contribute to. Less pronounced is the recurrence of authors in `build`, `config`, and `doc` activity types, for which there is more spread. Even though these activity types are common in software projects, they are less specialised and thus can be performed by different authors in different projects. As expected, the lowest median values correspond to the activity types least common to the software projects considered, *i.e.*, `lib` and `db` (both leafs in the $\widetilde{\mathbf{T}}$-graph). Since the boxplot for $RATW(a, \texttt{db})$ from Figure 2.17 is very low, we conclude that authors who prefer to specialise in `db` activities contribute to other activity types in the projects in which `db` activities are absent.

> Most authors contributing to `code`, `devdoc`, and `l10n` activity types in one project, do so in all other projects they contribute to. In contrast, database developers "wear many hats", *i.e.*, they contribute to other activity types in projects where `db` is not available.

To illustrate the point of the versatility of database developers we mention that one of the most active database contributors in *anjuta* has been involved in *gdl* as coder, translator, builder and even UI designer.

To obtain additional insights we study the difference between occasional ($AW < 14$) and frequent ($AW \geq 14$) contributors in the middle and bottom boxplots of Figure 2.19. Visual comparison of both sets of boxplots reveals striking differences: in the $AW < 14$ case, all activity types except `db` have a median of 1, suggesting that once occasional authors are involved in these activity types, they perform the same activities in all projects they contribute to. This should not come as a surprise: further inspection of the data revealed that 77.3% (1991 out of 2576) of the occasional contributors only participate in a single project.

On the other hand, the results (and $\widetilde{\mathbf{T}}$-graphs) for $AW \geq 14$ are similar to those for the entire ecosystem community: `code`, `devdoc`, `build`, and `l10n` are again the predominant activity types, suggesting that authors involved in these activity types choose to specialise in them in all projects they contribute to. For example, we have identified a frequent contributor ($AW = 1459$) involved in 28 projects, who dedicates more than 94% of his effort to `code`.

> Most occasional contributors participate in a single project. Frequent contributors specialise in `code`, `devdoc`, and to a lesser extent `build` and `l10n`, *i.e.*, they perform these activities in most projects they participate in.

2.4.4 What are the characteristics of specialised authors?

We wish to understand which of the author characteristics studied previously (*i.e.*, author workload $AW(a)$, number of activity types per author $NTA(a)$, and number of projects per author $NPA(a)$) are related to her degree of specialisation.

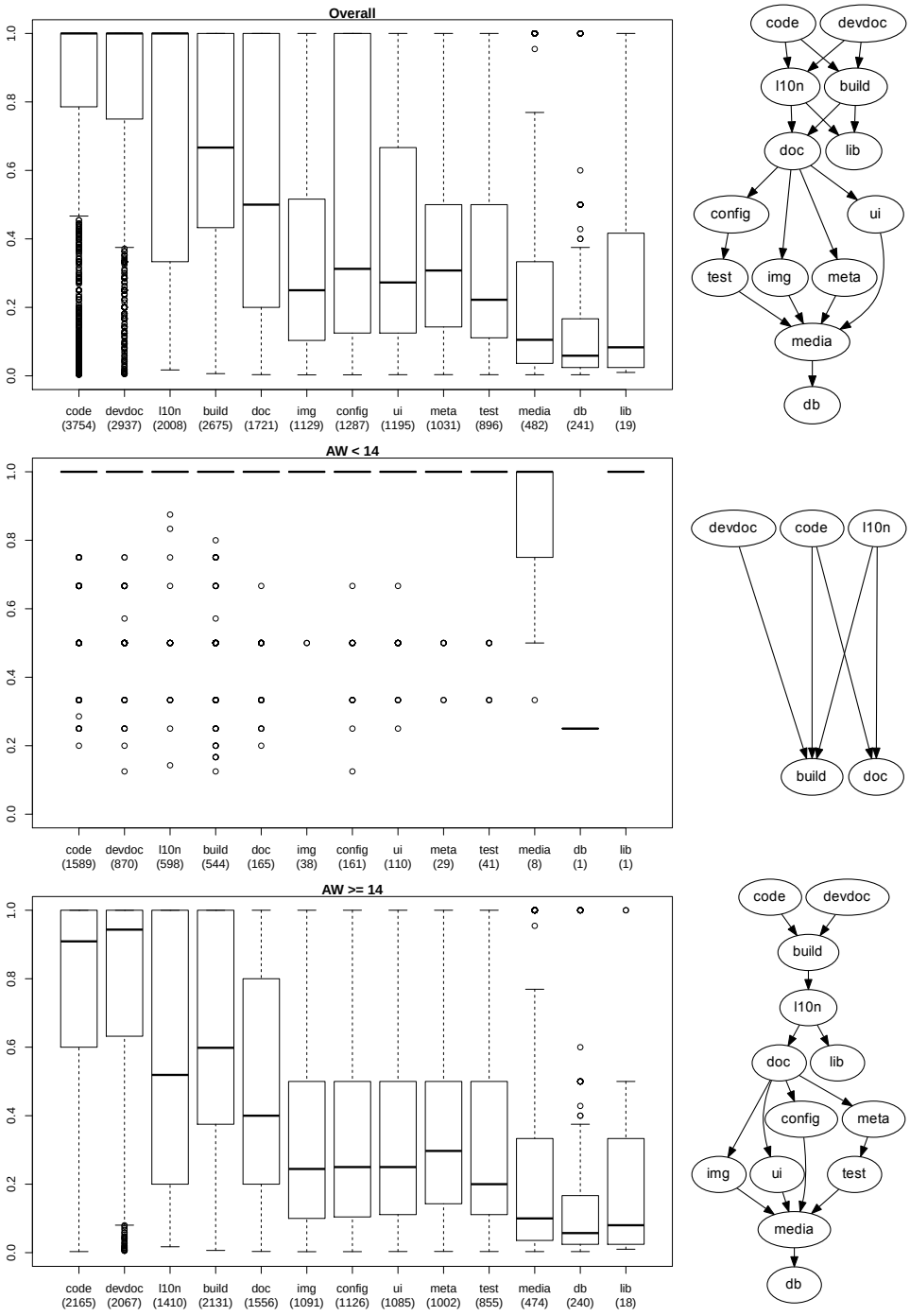


Figure 2.19: Boxplots for $RATI(a,t)$ per activity type t. Zero values are excluded. For $AW < 14$ the $\widetilde{\mathbf{T}}$-graph does not include `db` and `lib` since the $\widetilde{\mathbf{T}}$ procedure is not applicable for groups of size one.

2.4.4.1 Which characteristics of an author are observed when she is specialised towards few activity types?

In Figure 2.16 we observed that most authors specialise their workload in few activity types. We expect that authors contributing to many activity types prefer to spread their work across these types rather than concentrate it in few of them. Thus, we expect that authors involved in many activity types, as well as authors involved in many projects tend to be less specialised.

To answer the question we study Kendall correlation between $RAWS(a)$ and each of the author-specific $AW(a)$, $NTA(a)$, and $NPA(a)$ metrics. For all three metrics, we confidently reject H_0 at 0.01 significance level (p-value $< 2.2 \times 10^{-16}$). Similarly to the complementary question in Section 2.3.3.1, we observe negative correlation for all three metrics. However, now the correlation is much stronger: $\tau = -0.342$ for $NPA(a)$, $\tau = -0.497$ for $AW(a)$, and $\tau = -0.751$ for $NTA(a)$.

> Three factors (*i.e.*, number of activity types, number of projects, and number of file touches) are negatively correlated to specialisation of authors: the higher a factor, the less specialised an author is towards few activity types.

2.4.4.2 To what extent does the number of projects an author is involved in relate to her workload (share) for a particular activity type?

We wish to understand whether the number of projects $NPA(a)$ author a is involved in correlates to her workload $ATW(a,t)$ and her relative workload $RATW(a,t)$ for a particular activity type t. We expect that authors that participate in many projects contribute to `l10n` rather than `code` activities, hence the more projects an author contributes to, the higher the workload in `l10n` should be.

To answer the question we compute Kendall correlation between $NPA(a)$ and $ATW(a,t)$ on the one hand, and between $NPA(a)$ and $RATW(a,t)$ on the other hand, for all activity types $t \in T$. In concordance to Section 2.4.3.2, we discard the authors for which $ATW(a,t) = 0$. Figure 2.20 visually summarises the results of the correlation tests for ATW (black) and $RATW$ (gray), using the same visual conventions as in Figures 2.11 and 2.12.

For both $ATW(a,t)$ and $RATW(a,t)$ we confidently reject H_0 at 0.01 confidence level for all activity types except `lib`. In case of $ATW(a,t)$ correlation is positive for all activity types (except `lib` which is statistically insignificant), suggesting that the workload of authors increases in all activity types as they contribute to more projects. The four main activity types we previously observed at project level are therefore also confirmed at author level, with the highest correlation being observed for `l10n` ($\tau = 0.58$) and `devdoc` ($\tau = 0.57$) activity types. In contrast, correlation is negative for all activity types in case of $RATW(a,t)$ (*e.g.*, $\tau = -0.33$ for `code` and $\tau = -0.17$ for `l10n`) The statistical analysis therefore supports the observation one can make by inspecting Figure 2.18: the upper two-thirds of the picture are dominated by large red symbols (= `l10n`), while blue symbols (= `code`) in this region remain small and barely visible.

> As authors are involved in more projects, they contribute to `l10n` rather than `code` activities.

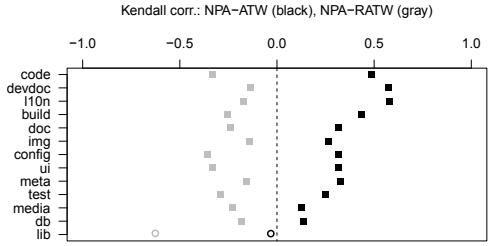


Figure 2.20: As they are involved in more projects, more authors concentrate their workload in `l10n` and `devdoc` than in other activity types (black). In addition, the share of their workload in `code` decreases in comparison to that in `l10n` (gray).

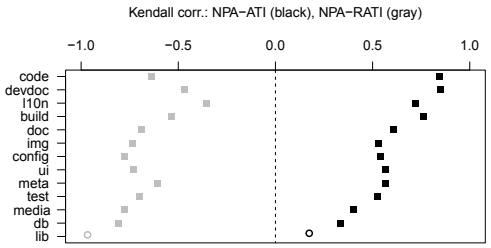


Figure 2.21: As they are involved in more projects, more authors contribute to `code` and `devdoc` in their new projects than to `l10n` (black). In addition, the share of their projects in which they `code` decreases in comparison to `l10n` (gray).

2.4.4.3 To what extent does the number of projects an author is involved in relates to the share of projects in which she performs a particular activity type?

We wish to understand whether the number of projects $NPA(a)$ an author is involved in relates to the absolute author involvement $ATI(a,t)$ or the involvement share $RATI(a,t)$ of these projects in which she performs activities of a particular type. We expect that not all activity types can support the same growth in terms of the number of projects in which they are performed. For example, we expect that the (relative) number of projects in which an author contributes to `db` or `lib` activities does not increase significantly as the author is involved in more projects since these activity types are performed in few projects in total. Moreover, even for main activity types such as `l10n` and `code`, we expect that it is more common for authors to perform `l10n` rather than `code` activities in most of the projects they are involved in, as this number grows.

To answer the question we compute Kendall correlation between $NPA(a)$ and $ATI(a,t)$ on the one hand, and between $NPA(a)$ and $RATI(a,t)$ on the other hand, for all activity types $t \in T$. Figure 2.21 visually summarises the results of the correlation tests for ATI (drawn in black) and $RATI$ (drawn in gray), for all activity types, under the usual conventions. For both $ATI(a,t)$ and $RATI(a,t)$ we confidently reject H_0 at 0.01 confidence level for all activity types except `lib`. For $ATI(a,t)$ we observe the strongest correlation for `code` and `devdoc` ($\tau = 0.85$), `build` ($\tau = 0.76$), and `l10n` ($\tau = 0.72$), while `db` exhibits the lowest correlation among the statistically significant activity types ($\tau = 0.33$). For $RATI(a,t)$ we observe negative correlation for all activity types (*e.g.*, $\tau = -0.63$ for `code`, and $\tau = -0.35$ for `l10n`). This indeed confirms our expectation. Recall that $RATI(a,t)$ is defined as the ratio between $ATI(a,t)$ and $NPA(a)$. Therefore, although increases in $ATI(a, \texttt{code})$ and $ATI(a, \texttt{l10n})$ both match increases in $NPA(a)$ (high positive correlation), it is only $RATI(a, \texttt{code})$ that decreases as $NPA(a)$ increases (high negative correlation). Therefore, we can say that the percentage of projects in which an author participates for the `l10n` activity does not decrease as she is involved in more projects.

Authors that are involved in more projects tend to participate more in `l10n` for these projects than in `code`, *i.e.*, the fraction of projects in which an author participates in `l10n` does not decrease as she is involved in more projects. (Recall the complement: as projects attract more developers, these are more commonly translators rather than coders).

Our finding concurs with the observation of Jergensen *et al.* [142] made for a subset of six GNOME projects. The overlap between the committer communities of these projects increases when documentation and translation committers are included as opposed to source code committers only. We conjecture that the ease of participation in cross-project translation activities is fostered by `Damned Lies`, the Web application used to manage the localization of GNOME[9] (cf. description of `intltool`, one of the predecessors of `Damned Lies` in [275, p. 44]).

2.4.5 Summary for Goal 2

Our second research goal consisted in *understanding how workload varies across authors belonging to the same community*, taking into account the different types of activities they perform.

First, we observed that author activity across the community follows a heavy-tailed distribution: most authors are occasional contributors with little activity, while relatively few authors are frequent contributors that are very active. Specifically, approximately half of the authors performed less than 14 file touches, while the most active author performed 185,874 file touches.

Next, we investigated the factors that influence how active authors are. We found that the more activity types an author participates in, or the more projects she contributes to, the more active she is. Moreover, we observed that when contributing to different projects within the ecosystem, authors prefer to specialise in a small number of activity types. In particular, occasional contributors typically participate in a single project, and a single activity type.

Focusing on different activity types, we observed that *coding*, *development documentation*, *localization*, and *build* are also the 4 activities in which members of the ecosystem community prefer to specialise. While frequent contributors prefer *coding*, occasional ones specialise in *localization*. Both frequent as well as occasional contributors are attracted to *development documentation* and *build*.

In terms of their versatility across projects, we observed that most authors contributing to *coding*, *development documentation*, and *localization* in one project, do so as well in other projects they contribute to within the ecosystem. However, as authors become involved in more projects, it is more common for them to participate in *localization* rather than *coding* across the different projects. On the other hand, authors contributing to scarce activities (such as *database*) "wear many hats", *i.e.*, they contribute to other activity types when their preferred ones are not available.

[9] `l10n.gnome.org`

2.5 Threats to validity

As in any empirical study, there are many potential threats to validity in our research.

Construct validity seeks agreement between a theoretical concept and a specific measuring device or procedure. In this respect, we equated the notion of *contributor* to the notion of Git *author* in our study. By taking into account other data sources (*e.g.*, mailing lists and bug trackers) we could consider a larger set of GNOME contributors and activity types, which could affect our results. Another potential threat is the lack of agreement between the theoretical concept of a "author" and a specific author identification technique described in Section 2.2.3. As recognised by Goeminne and Mens [106], *identity matching* can never be perfect due to the presence of false positives and false negatives. Using a wide portfolio of complementary algorithms we have reduced these to a minimum. In addition, we manually checked and corrected the remaining problems. Even if some incorrect identity matches may remain, their number will be limited and will not influence the results presented in this chapter. Note that we used a threshold of 0.8 for the similarity measures used during identity matching. This threshold was chosen based on a limited number of tests. A more appropriate value could be computed following the approach of Goeminne and Mens [106].

Construct validity might also have been affected by our operationalization of an author's *activity*. Our activity identification builds on and extends the work of Robles *et al.* [248]. We stress, however, that changing the rules (regular expressions) and the order in which they are evaluated may lead to different results. Looking at the exact changes made to each file may lead to a more precise identification of the activity type. In addition, our approach does not allow to associate more than one activity type to the same file. Construct validity is also related to our definition of *workload* and *involvement* as proxies for the actual development effort. To determine the workload we counted the number of file touches by an author for a project, without taking into account the size of the file change or the effort that was needed for making such a change. In considering the number of file touches rather than the modification size we follow a popular approach in software evolution research [68, 308]. A similar proxy of the development effort has been used by German [98].

Internal validity is related to validity of the conclusion within the experimental context of GNOME. A first threat is that we were not able to extract and analyse data from all 1358 GNOME projects (*i.e.*, 97%). We have left out 42 projects (3%) due to data extraction errors, but given the low percentage this will have little influence on the validity of the results. Since our study did not involve repeated application of a treatment, typical threats to internal validity such as history, maturation, or mortality [342] could not have affected the results of our study. Furthermore, we have paid special attention to the appropriate use of statistical machinery [268] (cf. Sections 2.2.6 and 2.2.7).

External validity is the validity of generalisations based on this study beyond GNOME. External validity is of no importance for this study as no claims are made about the generalisability of our results to other ecosystems. Although the studies presented in this chapter can be replicated on other open source ecosystems[10], the obtained results may vary, as each ecosystem has its own specific community and process. For instance, the significant share of translation activities in GNOME might be related to the special

[10]We have provided a replication package here: `www.win.tue.nl/mdse/gnome`. However, it will first need to be adapted in order to be applicable to other software ecosystems.

GNOME Live! translation project or `Damned Lies`, the Web application used to manage the localization of GNOME.

2.6 Related work

In the case study of this chapter we have studied the relationship between projects, authors and activity types in the open source software ecosystem GNOME. Throughout the paper we have added references to related work pertaining to individual steps in our analysis process. For example, existing work related to data analysis is discussed in Sections 2.2.6 and 2.2.7, while identity matching approaches are discussed in Section 2.2.3. In Subsection 2.6.1 we discuss existing results related to studies of developers and their activities, and in Subsection 2.6.2 we discuss studies of the GNOME ecosystem.

2.6.1 Studies of open source software contributors

Many researchers have investigated the roles developers play in open source software projects. Capiluppi *et al.* [44] attempted to characterize open source projects, their evolution, and the developer communities responsible for their maintenance, by studying 400 projects hosted by the FreshMeat[11] portal. While they distinguish between *stable* and *transient* developers based on the amount of changes they perform, we classify developers based on the types of files they touch. In addition, the projects analysed by them are not necessarily related, hence could be maintained by independent authors. In contrast, GNOME community members participate in multiple projects across the GNOME ecosystem. Shibuya and Tamai [269] also distinguish between frequent and occasional contributions, based on the number of commits developers contribute each month. However, even though they distinguish between different *activities* related to involvement in a project (*e.g.*, participating in mailing lists, reporting bugs, developing), they do not distinguish between different *development activities* (*e.g.*, coding, testing, writing documentation).

Mockus *et al.* [205] performed two case studies on the Apache and Mozilla projects where they investigated the roles and responsibilities of developers. Their approach distinguishes between developers who contributed code submissions, performed bug fixes, or reported problems. Although they do not study differences between development activity types, they observe specialisation of contributors towards a single role. Our data shows similar features: approximately 20% of the developers involved in *code* activities, and 25% of the developers involved in *localization* activities do not contribute to other activity types. Similarly to Mockus *et al.* [205], Nakakoji *et al.* [211] distinguish between developers, bug fixers and bug reporters. Furthermore, they have proposed a more refined classification of developer kinds: peripheral developers, active developers, core members and project leaders. The "onion model" proposed suggests that there are more core members than project leaders, more active developers than core members, more peripheral developers than active ones, etc. Similar hypotheses have been studied by Dinh-Trong and Bieman [76]. The Open Source community itself recognises that developers play different roles as witnessed, *e.g.*, by recording "credited developers" and "maintainers" as opposed to uncredited developers or maintainers in LINUX [206].

Benefits of incorporating the more refined classification in our work include discovering whether persons classified as core members in a number of projects tend to limit their

[11] `freecode.com`

involvement in other projects to bug reporting. However, as shown by Poncin *et al.* [231], integration of the refined classification proposed by Nakakoji *et al.* [211] necessitates additional analysis of bug tracker information. Yu and Ramaswamy [348] made a similar distinction between core and associate project members, but unlike Nakakoji *et al.* [211] their approach infers roles automatically based on clustering developers using frequency of their interaction.

Our approach is based on, and extends that of, Robles *et al.* [248], who also distinguish between development activity types. However, while they follow a holistic approach and classify *commits* in different activity types, we perform the distinction both at the level of individual software *projects*, as well as that of individual *authors* participating in these projects.

2.6.2 Studies of the GNOME ecosystem

GNOME, being a large open-source software ecosystem, comprising a wide variety of diverse projects, is a very popular case study in software evolution research.

GNOME was part of the MSR 2009 and 2010 Mining Challenges. In 2009, there was a general challenge to demonstrate the usefulness of mining tools on the GNOME case study, and a prediction challenge to predict the code growth at project level. [180] and [255] mined the GNOME Bugzilla database, and [270] mined the GNOME Internet Relay Chat (IRC) meeting channels. In this chapter, we investigated a different data source, namely the version control repositories. [189] focused on the visualization of the GNOME ecosystem. Although they are interested in similar questions as we are, they do not provide any statistical evidence. [47] found an inverse relation between file size and the notion of *author entropy*, suggesting that large files are more likely to have a dominant author than small files. The notion of author entropy characterizes the distribution of author contributions to a file, and is therefore related (at least in spirit) to our use of inequality indices such as Gini or Theil. The main difference is that we did not focus on the author collaboration for individual files.

In 2010, MSR focused on software ecosystems and, more in particular, on the relationships between packages, by relying on information stored in the SVN version control system and the mailing list archives [130]. There were two contributions to this mining challenge that used GNOME as a case study. Krinke *et al.* [164] focused on the reuse and cloning of code between the different GNOME projects. Luijten *et al.* [187] focused on the process and efficiency of issue handling. These topics did not take activity types or GNOME contributors into account and are thus different in scope from the research in this chapter.

Similarly to Neu *et al.* [215] we recognize the importance of combining the analyses of the ecosystem and an individual project, as well as the community and an individual contributor. However, while Neu *et al.* [215] focused on visualization of GNOME data based on a number of assumptions, we conducted statistical analyses. Our findings support their assumptions, since we also observed that persons "who contributed a lot but only to a relatively small number of projects" are typically coders, while those "who committed less often but to more projects" are typically translators.

In their study of effort, co-operation and co-ordination in GNOME, Koch and Schneider [155] have observed significant differences between contributions of different developers in terms of lines of code. Both their data and our data on the workload in terms of file touches (Section 2.3.1) show similar features: the distributions are left-skewed and

the maximal value is more than an order of magnitude larger than the mean, *i.e.*, both distributions are heavy-tailed [291]. A similar distribution seems to be suggested by partial data on percentages of modification requests per developer, as reported by German [98]. As opposed to our work, Koch and Schneider [155] consider a more advanced approach to effort estimation that takes into account lines of code added or deleted as well as the communication between the developers via mailing lists. Furthermore, while Koch and Schneider [155] follow a holistic approach, we augment such results with more fine-grained analyses of individual GNOME projects. Similarly to Koch and Schneider [155], Gousios *et al.* [111] developed an advanced measure of individual developer contribution based on information from the source code repository, the mailing lists and the bug tracking systems, and applied the measure to a number of GNOME projects. Jergensen *et al.* [142] have studied six GNOME projects in order to understand how developers join, socialize and develop within GNOME. They observed, among others, that very experienced developers are *less* involved in the actual coding. Similarly to our work, Jergensen *et al.* [142] have observed that translation and documentation are more cross-project activities than coding. Summarizing this discussion we observe that most of the studies so far either followed a holistic approach and considered GNOME as one system, or focused on a number of example GNOME projects such as Evince or Nautilus.

Finally, Lopez-Fernandez *et al.* [183] studied relations between the GNOME developers by means of social network analysis, while Ernst and Mylopoulos [85] studied perception of software quality requirements in some of the GNOME projects.

2.7 Future work

Our research can be extended in numerous ways.

While in this chapter we have considered all GNOME projects as being equal, in reality they are classified under different categories (see `git.gnome.org/browse/`): Archived, Administration tools, Bindings, Desktop, Development tools, Infrastructure, Platform, Productivity tools, Other and Deprecated. GNOME projects could also be clustered along other dimensions. For example, all GNOME projects related to multimedia activity (*e.g.*, *bonobo-media* would belong to this cluster). We intend to look into different such classifications and clusterings to statistically investigate whether projects belonging to the same category share common properties, and to what extent differences between projects can be explained by these categories [59, 262, 263].

An important area of future work is to look at how the presented metrics and statistics *evolve* over time. This would allow us to detect certain trends (or trend breaks) in how ecosystems and communities evolve, predict future evolutions, and compare the evolution of projects (or authors) against one another. Recently we started to explore this temporal dimension [108].

We intend to take into account other data repositories in future studies. In particular, we wish to integrate data coming from bug trackers and mailing lists [111, 231]. On the one hand, this gives us access to a richer source of information. On the other hand, it makes the integration of these different data sources more challenging. We also intend to apply our study to other software ecosystems, such as APACHE, KDE and GNU.

The distinction between different development activities, *e.g.*, *coding* and *translation*, can be used to refine measures of experience and recognition intended for quantification and comparison of the contributions of open-source software developers in an objective,

open and reproducible way [46]. These measures can be used both by software developers looking for a job and by recruiters evaluating suitability of such candidates [45].

Finally, we intend to use more characteristics when studying the variation across projects and authors. For projects we intend to include, among others, project size, project maturity, main programming language used, and application domain. For communities we intend to take into account, among others, developer seniority, team size, and team structure.

2.8 Conclusions

This chapter studied the workload variation of the projects belonging to an open source software ecosystem, and the workload variation of the contributors belonging to the ecosystem community. To achieve this, a portfolio of statistical techniques was applied on the GNOME case study, a large and well-known open source ecosystem and associated community (with over 1300 different projects and over 5000 active authors).

By analysing the GNOME mailing list archives, we observed that GNOME contains both paid contributors and volunteers, and it can be expected that their workload is different. The GNOME community also acknowledges that, while coding is the most important activity, other activities such as translation/localisation are indispensable. In addition, translators are usually not coders, and most of the other GNOME activity types (such as documentation, user interface design, etc.) are considered to be less important.

To confirm these informal observations, we studied the GNOME ecosystem with two research goals in mind: to understand *how workload varies across projects* and to understand *how workload varies across contributors*. To achieve this, we defined a novel set of metrics, parametrised by project, author and activity type, with *coding*, *localization* and *development documentation* being the most important activity types. This set of metrics can be considered as a contribution in its own, as it can be reused easily for studying other software ecosystems. Of particular importance are the specialisation metrics that are defined based on the Gini inequality index. An additional contribution consists in introducing $\widetilde{\mathbf{T}}$-graphs, a novel approach for reporting the results of comparing multiple distributions.

Concerning the first research goal, we observed that project workload across the ecosystem follows a log-normal distribution. We found two characteristics that positively correlated to the project workload: the size of the project community and the number of activity types contained in the project. The workload was mainly concentrated in four activity types, that also attracted most of the project's contributors: coding, development documentation, localization, and build. Of these, coding concentrates most of a project's workload, while localization and development documentation attract most of the contributors. We also found evidence that highly specialised projects only include few activity types. In contrast, we did not find evidence that projects with more activity types or larger communities are less specialised.

Concerning the second research goal, we observed that author workload across the GNOME ecosystem follows a heavy-tailed distribution: most contributors have little activity (approximately half of the contributors performed less than 14 file touches), while a small number of contributors have a very high workload. We found that a contributor's workload is positively correlated to the number of projects she contributes to, as well as to the number of activity types she participates in. We also observed that contributors

prefer to restrict themselves to a small number of activity types. In particular, the many occasional contributors typically restrict themselves to a single project and a single activity type. While occasional contributors are mainly specialised in the localization activity, frequent contributors tend to prefer coding. Both kinds of contributors are also often involved in development documentation and build. Most contributors to one of these four activity types in one project, also tend to contribute to these types in the other projects they are involved in. However, the more projects a contributor is involved in, the more she tends to participate in localization as opposed to coding.

Overall, our empirical case study has allowed us to confirm that there is no such thing as a uniform ecosystem of projects and contributors: when taking into account the activity types and the workload, there is a lot of variation across projects and across contributors, but with a clear preference towards the activity types of coding, localization, development documentation and building. It is quite possible that other ecosystems than Gnome may reveal other activity patterns.

Chapter 3

A model of skill diversity

OSS communities are typically very specialised, on the one hand, and experience high turnover, on the other. Combination of specialization and turnover can cause, for example, parts of the system implemented in a certain programming language to become unmaintainable, if knowledge of that language has disappeared together with the retiring developers.

*Inspired by measures of linguistic diversity from the study of natural languages, we propose a measure of skill diversity in OSS communities. When applied to knowledge of programming languages, for instance, our measure can signal when an OSS community is at risk of not having enough maintainers for code implemented in those languages. To illustrate our approach, we studied risks associated with different languages in Emacs, and found examples of low risk due to high popularity (*e.g.*, C, Emacs Lisp); low risk due to similarity with popular languages (*e.g.*, C++, Java, Python); or high risk due to both low popularity and low similarity with popular languages (*e.g.*, Lex). Our results show that methods from the social sciences can be successfully applied in the study of information systems, and open numerous avenues for future research.*

3.1 Introduction

Open source software (OSS) development is typically characterised as a decentralised, self-directed, highly interactive, and knowledge-intensive process [123]. In OSS, programmers with different skill sets and skill levels, supporters, and users organise themselves in virtual (online) communities, and voluntarily contribute to a collaborative software project [211].

OSS communities are typically very specialised [232, 248, 320]: contributors focus on few activity types and are very territorial, touching only few parts of the system [99] (also Chapter 2). OSS communities also co-evolve together with the associated OSS systems [211]: faced with turnover [246], these communities are sustained and reproduced over time through the progressive integration of new members [79]. However, with the abandonment of existing developers, OSS communities lose human resources with

knowledge of the system or of some of its components, or, stated differently, with mastery of certain programming languages. Ensuring the heterogeneity of an OSS community in terms of the skills of its members is important for a project's survival and performance [104]. To further put this issue into context, software systems are increasingly developed using multiple programming languages, as illustrated by the growing proportion of multi-language software developed in the United States from 1998 (30%) [144] to 2008 (50%) [145]. In addition, as languages become obsolete and development teams are faced with the problem of maintaining legacy code, or migrating it in order to survive, finding developers with knowledge of obsolescent technologies becomes more challenging. As the case may be, OSS communities are exposed to the *risk of not finding suitable contributors with knowledge of certain programming languages.*

Although new to software maintenance research, quantifying the risks associated with knowledge of programming languages in OSS communities around multi-language systems is related to the well-known concept of linguistic diversity from the study of natural languages [116]. Drawing inspiration from measures of linguistic diversity (Section 3.2), in this paper we attempt to quantify the aforementioned risk, associated with a given programming language in an OSS community (Section 3.3). Our model assumes that contributors are polyglot, *i.e.*, they can "speak" more than one programming language. Moreover, analogously to dialects of a natural language being regarded as similar (mutually intelligible), our model also considers certain programming languages to be related. To quantify the strength of this relation, we mine patterns of shared knowledge of programming languages from developers participating in Stack Overflow, a popular Q&A website (Section 3.4). Such relations need not be symmetrical: just like "Swedish is more easily understandable for a Dane, than Danish for a Swede" [204], our Stack Overflow-based measure considers, *e.g.*, that a C++ developer would be able to take over code written in C with less difficulty than the other way around.

By design, we can distinguish between two types of programming languages: those causing *high risk* within an OSS community (due to limited spread and low "similarity" with other more popular languages), and those causing *low risk* (either due to their popularity, or to their closeness to other more popular languages known to members of the community). Finally, to illustrate our risk measure, we track its evolution throughout the evolution of Emacs (Section 3.5).

3.2 Linguistic diversity for natural languages

Measuring linguistic diversity for natural languages is an old research topic, dating back to Greenberg in 1956 [116]. For a given geographical area, Greenberg considers the probability that two randomly-chosen individuals *do not* speak the same language as a measure of the region's linguistic diversity. In this model (the first in a series of eight such measures proposed by Greenberg), if everyone speaks the same language, the probability that two randomly-chosen individuals speak the same language is, naturally, 1. Similarly, if everyone speaks a different language, this probability is 0. In general, for a language ℓ, the probability p_ℓ that a randomly-chosen individual speaks ℓ is the proportion of ℓ-speakers to the total population, *i.e.*, $p_\ell = \frac{|S_\ell|}{|P|}$, where S_ℓ is the set of ℓ-speakers, P is the entire population, and $|\cdot|$ denotes cardinality. Consequently, the probability that two randomly-chosen individuals speak ℓ is p_ℓ^2, hence the probability that two randomly-chosen individuals speak the same language is $\sum_{\ell \in L} p_\ell^2$, where L is the set of all languages spoken

in that region. The Greenberg linguistic diversity index A [116], corresponding to this simple model, is defined as[1]

$$A = 1 - \sum_{\ell \in L} p_\ell^2 \tag{3.1}$$

A reaches its minimum of 0 when everyone speaks the same language (linguistic uniformity). Similarly, A reaches its maximum of 1 when everyone speaks a different language (maximal linguistic diversity). However, this model is overly simplistic, since (i) it does not consider *mutual intelligibility between different languages* (linguistic diversity should be lower in areas where related languages or dialects are spoken), and (ii) it does not consider *polyglotism* (a member of the population can speak more than one language). Concerns (i) and (ii) above are orthogonal. To account for (i), Greenberg proposed B [116], defined as

$$B = 1 - \sum_{\ell, m \in L} p_\ell p_m \cdot sim(\ell, m), \tag{3.2}$$

where $sim(\ell, m)$ is a measure of mutual intelligibility interpreted as the similarity between languages ℓ and m (ranging between 0 when ℓ and m are completely independent, and 1 when $\ell = m$). Clearly, B reduces to A if $sim(\ell, m) = 1$ when $\ell = m$, and 0 otherwise.

To account for (ii), if polyglot, an individual is considered equally probable to speak any of the languages she commands, hence the expressions for A and B above are adjusted accordingly. Let $\mathcal{L}$ be the power set (set of all subsets) of L, excluding the empty set; $|\mathcal{L}| = 2^n - 1$, where $n = |L|$. For example, if $L = \{A, B, C\}$, then $\mathcal{L} = \{A, B, C, AB, AC, BC, ABC\}$. Let X_ℓ be the set of *exclusive* ℓ speakers, *i.e.*, individuals that speak ℓ but do not speak any other language besides ℓ. For a subset $s \in \mathcal{L}$, by abuse of notation, we write X_s to denote the set of individuals that speak the combination of languages in s exclusively (*i.e.*, they speak all languages in s, but no other languages besides those). By definition, $\sum_{s \in \mathcal{L}} |X_s| = |P|$. Index B becomes [116]

$$F = 1 - \sum_{s, t \in \mathcal{L}} p_s p_t \cdot \frac{\sum_{\ell \in s, m \in t} sim(\ell, m)}{|s| \cdot |t|}, \tag{3.3}$$

where $p_s = \frac{|X_s|}{|P|}$, for all $s \in \mathcal{L}$. Clearly, F reduces to B in the monolingual case, since $|s| = 1$ for all $s \in \mathcal{L}$. B further reduces to A as discussed above.

Indices B and F as defined above interpret mutual intelligibility as similarity and hence assume it to be symmetric. However, it is well-known that mutual intelligibility is not necessarily symmetric: *e.g.*, Swedes have more difficulties understanding Danish as opposed to Danes attempting to understand Swedish [204]. Therefore, one can define $sim(\ell, m) = \max(mi_\ell(m), mi_m(\ell))$ where $mi_\ell(m)$ is the measure of intelligibility of the language m for speakers of language ℓ. Similarly to $sim(\ell, m)$ we require $mi_\ell(m)$ to range between 0 and 1, such that $mi_\ell(m) = 0$ if m is unintelligible for the speakers of ℓ and $mi_\ell(m) = 1$ if m is intelligible for all speakers of ℓ. In particular, if $\ell = m$ then $mi_\ell(m) = 1$.

[1] In the original paper [116] Greenberg only describes the linguistic diversity indices, but does not formalise them. The current formalisation is ours.

3.3 Risk of using a programming language

There are many risks impacting software development, and many methods to estimate them [216]. In this chapter we do not aim to cover all possible facets of risk, but rather focus on a particular scenario. For a (open source) software project using multiple programming languages, we study the readiness of the developer community to take over code implemented in a certain language, and evaluate the risk of not finding contributors that can "speak" that language.

Based on the discussion of linguistic diversity above, we require that a measure of this risk be *domain-specific*, *i.e.*, aware of relations between programming languages. To simplify our model, we assume *perfect fluency* of developers in all the features of the languages they speak, even as the languages evolve. In addition, we assume *constant knowledge in time* (*i.e.*, once a developer speaks a certain programming language, she never "forgets" how to speak it). For the purpose of empirically illustrating the risk measure (Section 3.5), we need to approximate at each point in time the developers with knowledge of a certain programming language. To this end, we furthermore assume *instant fluency*: a developer is said to "speak" a certain programming language at time τ if she has performed at least one change to a source code file in that language, prior to τ. Relaxing these assumptions is considered as future work.

3.3.1 Risk measure

Let S be a multi-lingual software system, and let D be its developer community at time τ. Let L be the set of programming languages in use in S at time τ, *i.e.*, those for which there exist source code files at time τ that need to be maintained. Similarly to the formalisation of the Greenberg indices from Section 3.2, let $\mathcal{L}$ be the power set of L, excluding the empty set. Let X_ℓ be the set of developers at time τ that speak ℓ *exclusively*, *i.e.*, they speak ℓ but do not speak any other language besides ℓ. For a subset $s \in \mathcal{L}$, let X_s be the set of developers at time τ that speak the combination of languages in s exclusively (*i.e.*, they speak all languages in s, but no other languages besides those). By definition,

$$\sum_{s \in \mathcal{L}} |X_s| = |D|. \tag{3.4}$$

For a programming language $\ell \in L$, we define the risk of S at time τ of not finding developers that *can* speak ℓ as

$$risk_\ell = 1 - \sum_{s \in \mathcal{L}} p_s \cdot \max_{k \in s} mi_\ell(k), \tag{3.5}$$

where $p_s = \frac{|X_s|}{|D|}$ is the probability at time τ that a developer speaks the combination of languages in s exclusively, and $mi_\ell(k)$ is an asymmetric mutual intelligibility measure as above. To illustrate the need for an asymmetric measure recall, for example, that C was originally a subset of C++ (the version of C defined by C89 is commonly referred to as the "C subset of C++" [256]), hence we perceive C to be more similar to C++ than C++ is to C. Therefore, assuming fluency of developers in all language features, and comparable complexity of the different components, we expect a C++ developer to be able to take over C code with less difficulty than the other way around.

As opposed to Greenberg [116] who is interested in an "average" case (*i.e.*, as discussed in Section 3.2, if polyglot, an individual is considered equally probable to speak any of the languages she commands) when computing the linguistic diversity index F (3.3), we opt for the $\max(\cdot)$ function in (3.5) to denote that if polyglot, it is the language most intelligible to the language in question that will influence how difficult it is for a developer to take over that code.

To obtain a better understanding of how (3.5) can provide insights in the risk of not finding developers that can speak ℓ, we distinguish between developers D_ℓ that speak ℓ, and developers $D_{\neg\ell} = D \setminus D_\ell$ that do not speak ℓ. Similarly, let $\mathcal{L}_\ell$ be a subset of $\mathcal{L}$ such that $\forall s \in \mathcal{L}_\ell, \ell \in s$, and let $\mathcal{L}_{\neg\ell} = \mathcal{L} \setminus \mathcal{L}_\ell$. Then, we can rewrite (3.5) as $risk_\ell = 1 - \sum_{s\in\mathcal{L}_\ell} p_s \cdot \max_{k\in s} mi_\ell(k) - \sum_{s\in\mathcal{L}_{\neg\ell}} p_s \cdot \max_{k\in s} mi_\ell(k)$ which, given that $mi_\ell(k) = 1$ if $k = \ell$, and $\max_{k\in s} mi_\ell(k) = 1$ for all $s \in \mathcal{L}_\ell$, further simplifies to:

$$risk_\ell = \frac{|D_{\neg\ell}|}{|D|} - \sum_{s\in\mathcal{L}_{\neg\ell}} p_s \cdot \max_{k\in s} mi_\ell(k) \tag{3.6}$$

Closer inspection of (3.6) reveals that the risk of not finding developers that can speak ℓ is *high* if very few developers speak ℓ (*i.e.*, $\frac{|D_{\neg\ell}|}{|D|} \simeq 1$) and other languages are very distinct from ℓ rendering ℓ barely intelligible for "speakers" of those languages (*i.e.*, $\sum_{s\in\mathcal{L}_{\neg\ell}} p_s \cdot \max_{k\in s} mi_\ell(k) \simeq 0$ because the languages in the collection are very different from ℓ, $\max_{k\in s} mi_\ell(k) \simeq 0$). By a complementary argument, two typical *low*-risk scenarios are when almost everybody can speak ℓ (*i.e.*, $\frac{|D_{\neg\ell}|}{|D|} \simeq 0$, hence $p_s \simeq 0$ for $s \in \mathcal{L}_{\neg\ell}$), or when almost nobody can speak ℓ (*i.e.*, $\frac{|D_{\neg\ell}|}{|D|} \simeq 1$) but popular languages make ℓ easily understandable (*i.e.*, $\max_{k\in s} mi_\ell(k) \simeq 1$ for $s \in \mathcal{L}_{\neg\ell}$). To distinguish between these two scenarios in the empirical evaluation (Section 3.5), we also consider the percentage of developers that do not speak ℓ, $\frac{|D_{\neg\ell}|}{|D|}$.

3.4 Mutual intelligibility for programming languages

Mutual intelligibility, while being distinct from traditional notion of similarity, is still close to it. Therefore, in this section we mostly focus on measures of similarity of natural languages [88, 116] and their counterparts in programming linguistics [96] (the study of programming languages), and introduce our mutual intelligibility measure based on analysing Stack Overflow[2].

3.4.1 Approaches to similarity between programming languages

In linguistics, two complementary approaches to compute similarity between languages are commonly pursued. First, a similarity measure can be obtained "using an arbitrary but fixed basic vocabulary, *e.g.*, the most recent version of the glottochronology list", by computing "the proportion of resemblances between each pair of languages to the total list" [116] (a similar approach has been recently pursued to study asymmetric mutual intelligibility [204]). Second, a similarity measure can be obtained using the distance between the branches languages fall into in a classification tree [88]. Using this approach, the more features two languages have in common, the more similar they are.

[2] http://stackoverflow.com

In programming linguistics, the approaches above are to a large extent unfeasible. First, application of approaches based on a common vocabulary would require establishing an agreed list of universal concepts present in all programming languages, akin to the Swadesh list for the natural languages [288]. The "word list" approach is being criticized in linguistics [122]; moreover, it introduces the need for identifying so-called cognates, or etymologically related words. The process of identifying cognates is complicated, since cognates do not necessarily look similar and words that look similar are not necessarily cognates [119]. Choosing the word list approach for similarity of programming languages assumes presence of universal concepts common to all (or at least most) programming languages. Even if compilation of such a list is possible at any given moment, it would rapidly become obsolete, since programming languages emerge much faster than natural languages. Moreover, the word list approach can be expected to trigger similar discussions about possible cognates, *e.g.*, whether notions of a function in Lisp and C should be considered cognates or not.

One could also base a similarity measure on the shared concepts that underlie the design of both languages (*e.g.*, data and types, variables and storage) and the paradigms to which they adhere (*e.g.*, imperative or object-oriented) (cf. [88]). "We can master a new programming language most effectively if we understand the underlying concepts that it shares with other programming languages" [334, p4]. Again, the more attributes two languages would share in common, the more similar they would be considered. However, selecting the right attributes is challenging, to say the least. Most reliably, one could make use of taxonomies of programming languages [240]. However, as languages evolve, such taxonomies become inherently out of date and their categories change [141].

As an alternative to word-list and classification-tree approaches, one may consider recent studies [73, 146] that targeted the joint usage of programming languages. Karus and Gall [146] studied 22 open-source systems and observed two groups of languages for which the source code files frequently co-change, namely XML, XML Schema, WSDL (Web Service Definition Language) and Java on the one hand, and JavaScript and XSL (*e.g.*, XSLT, XPath) on the other hand. In a larger-scale study of 9,997 projects, Delorey *et al.* [73] observed that JavaScript and PHP, Java and JavaScript, C and C++, and C and Perl are commonly used both by the same authors as well as in the same projects.

As opposed to the actual usage, reflected in implementation of software systems, Doyle and Stretch [77] studied services offered by British software companies. The authors considered two programming languages to be similar if multiple companies offered these languages as part of their services. While Doyle and Stretch [77] employ the term "related by usage" to describe this relation, we prefer to call it "related by knowledge" and to reserve the term "related by usage" to such approaches as [73, 146]. Indeed, companies offering multiple programming languages as part of their services do not necessarily use these languages in the same project. Instead, these companies have employees, potentially different, knowledgeable about each of these languages.

Both the "related-by-usage" and "related-by-knowledge" approaches can be seen as pertaining to pragmatics of programming languages which, together with semantics, are considered the most decisive for quantifying the similarity between programming languages [334, p5]. Therefore, we also expect that as opposed to both the word-list and classification-tree approaches, *pragmatic similarity* more accurately reflects developer expertise and ease of switching from one programming language to another. In Section 3.4.3 we also propose a pragmatics-pertaining mutual intelligibility measure, refining the "related-by-knowledge" insights of Doyle and Stretch [77]. Specifically, we base

the mutual intelligibility measure on shared knowledge of the programming languages, as reflected in Stack Overflow tags representing programming languages.

3.4.2 Stack Overflow

Stack Overflow (SO) is a free programming questions and answers (Q&A) site known to foster knowledge sharing among the developers [318, 319] (Chapters 4 and 5). When posting a question, SO users associate at least one and at most five different tags to it, and, in turn, become associated with these tags. When answering a question, SO users inherit all the tags associated with this question. Therefore, while each question can have at most five tags, a user can inherit an arbitrarily large collection of tags from all the questions she asked and answered. Tags can be related to programming languages (*e.g.*, `c#`, `java`, `php`), operating systems (*e.g.*, `windows`, `linux`), specific frameworks or technologies (*e.g.*, `hibernate`, `grails`), specific versions of either of the above (*e.g.*, `c#-4.0`, `windows-7`, `hibernate-4.x`), cross-cutting concerns (*e.g.*, `logging`, `algorithm`), or other topics. SO tags can be collaboratively edited: while anyone can suggest an edit to question tags, only higher ranked users can review and edit tags, ensuring quality and reliability of the tags. Here we explore the public SO data from September 2011 (2,010,348 questions and 756,694 users).[3] The data is organised such that one can distinguish between *question tags* and *user tags*; one can also distinguish between user tags collected from asking questions, and user tags inherited by answering questions.

Question tags. Frequent pairs of tags (*e.g.*, `javascript–jquery`, `asp.net–c#`) indicate that these languages are commonly used together. However, this approach has several drawbacks. First, the number and skills of the users answering these questions is not considered (potentially leading to false positives). For example, there may be many questions tagged τ_1 and τ_2, suggesting that these languages are related, but only few people answering them. The relation between τ_1 and τ_2 might therefore not be representative of the entire (large) developer community (*e.g.*, although few gurus with knowledge of both τ_1 and τ_2 exist, average developers may not possess the skills to easily switch between them).

User tags – answering questions. Since users inherit tags from questions they answer, frequent pairs of tags indicate that developers who possess knowledge of one language commonly also possess knowledge of the other. A frequent pair of tags (τ_1, τ_2) can emerge from multiple situations:

- many users inheriting τ_1 and τ_2 by answering questions tagged (τ_1, τ_2): either there are few questions tagged (τ_1, τ_2), but many users answering them (*i.e.*, although τ_1 and τ_2 do not seem to be commonly associated in practice – *e.g.*, they used to be but are by now obsolete, there is still a large pool of developers mastering both), or there are many questions tagged (τ_1, τ_2), and many users answering them (*i.e.*, τ_1 and τ_2 are both commonly associated in practice, and supported by a large user base);

[3] `http://www.clearbits.net/torrents/1836-sept-2011`

- many users inheriting τ_1 from questions tagged τ_1, and τ_2 from different questions tagged τ_2 (hence the pair τ_1–τ_2). In addition to an argument similar to the previous one, this also indicates languages that although rarely related in practice, are commonly mastered by developers. Hence, although seemingly unrelated, it seems easy for developers to switch from one language to another since they frequently master both.

User tags – asking questions. Users also inherit tags from all the questions they ask. A frequently occurring pair (τ_1, τ_2) indicates that developers are frequently faced with joint usage of τ_1 and τ_2, irrespective of the expertise available. In turn, this can suggest an emerging trend in relating τ_1 and τ_2 by usage. Note that as opposed to the variation across questions (many questions [of the same person]), frequent pairs (τ_1, τ_2) are now supported by large user pools (many questions of many persons). However, this does not indicate developer expertise.

Questions vs. users. In conclusion, both the questions-based and the users-based approaches are subject to potential false positives resulting from competing rather than interacting languages. However, we opt for user tags rather than of question tags, since the former can suggest relations between languages representative of the skills of the developer community (*i.e.*, there are many users that share knowledge of both—fewer false positives), as well as relations between independent languages (*i.e.*, languages which are seemingly unrelated, but knowledge of both is frequently shared by users—fewer false negatives).

3.4.3 Stack Overflow-based mutual intelligibility measure

To quantify shared knowledge of programming languages by developers participating in SO discussions, we perform association rule mining [6] on SO tags representing programming languages. We say that a language k (with tag τ_k) is mutually intelligible or "related by knowledge" to a language ℓ (with τ_ℓ) if many of the SO users having inherited τ_k are also associated with τ_ℓ. As mutual intelligibility measure of language k with respect to language ℓ we choose *confidence*, one of the measures typically used to quantify the strength of association rules.

$$mi_\ell(k) = conf(\tau_k \Rightarrow \tau_\ell) = \frac{nBoth}{nLeft}, \tag{3.7}$$

where *nLeft* is the number of users associated with τ_k, and *nBoth* is the number of users associated with both τ_k and τ_ℓ.

To ensure quality of the association rules, we perform a number of pre- and postprocessing steps. Preprocessing consists of filtering out potentially unreliable posts (either questions or answers with negative or zero score, as reflected by the number of votes), and infrequent pairs of tags (encountered for a single user). This limits the number of eligible SO contributors to slightly over 400,000 (out of 756,694 initially). Postprocessing is based on *lift*, another popular quality measure for association rules. If $lift > 1$, the tags appear more frequently together in the data than expected under the assumption of conditional independence [37]. Moreover, we require it to be unlikely that $lift > 1$ is observed only by chance and perform Fisher's exact test to determine statistical significance. Hence,

	Asm	C	C++	Cobol	CSS	Groovy	HTML	Java	JavaScript	Perl	PHP	Shell	XML
Asm	100	55	54	1	15	1	23	39	28	12	28	1	18
C	8	100	48	0	12	1	17	31	21	8	21	0	13
C++	5	32	100	0	10	1	15	26	18	6	18	0	11
COBOL	12	35	40	100	24	3	29	48	38	17	37	1	28
CSS	2	10	13	0	100	1	61	21	54	5	39	0	16
Groovy	3	15	18	1	17	100	26	63	32	7	23	0	26
HTML	2	11	14	0	46	1	100	25	56	5	40	0	18
Java	2	12	15	0	10	2	15	100	19	4	16	0	12
JavaScript	2	9	11	0	25	1	35	20	100	4	31	0	13
Perl	5	25	27	1	18	2	26	31	30	100	31	1	19
PHP	2	9	11	0	19	1	26	17	33	4	100	0	12
Shell	12	34	38	1	19	3	32	43	33	24	35	100	24
XML	3	14	19	0	20	2	29	34	35	7	31	0	100

Table 3.1: SO-based mutual intelligibility measure ([column] with respect to [row], percentages).

we say that k is unintelligible to the speakers of ℓ (we redefine when $mi_\ell(k) = 0$) if $lift \leq 1$, or $lift > 1$ is not statistically significant at 5% significance level. Approximately 7% of the pairs were removed by this filtering step (*e.g.*, Python ⇒ Visual FoxPro has lift 0.83; Curry ⇒ C# has lift 1.78, $p = 0.31$). Finally, to reduce the amount of data processing required, we limit our scope to a subset of 160 programming languages, hence 160 corresponding SO tags. Our subset includes the most popular programming languages mentioned by TIOBE[4], Wikipedia[5], and the Transparent Language Popularity Index[6], as well as exotic languages such as M4 and RelaxNG, in use in Emacs. The complete list of languages included in our selection is part of the online appendix.[7]

3.4.4 Empirical results

Table 3.1 displays values of the mutual intelligibility measure for a subset of the programming languages considered (also studied in [73, 77, 146]). An entry (*row*, *column*) represents the similarity of the language in *column* with respect to the one in *row*. For complete results we refer to the online appendix[7]. By definition each language is perfectly mutually intelligible with itself (100% on the main diagonal). Next we observe that Assembly programmers are usually well-versed in other languages, including HTML (23%), Java (39%) and JavaScript (28%). Since there are only 44 posts (questions+answers) tagged `assembly` and `java`, these languages are unlikely to be related by usage, but are related by knowledge (more than 1000 developers with knowledge of both). Hence, if replacement developers are required for Java, one might consider the Assembly developers as candidates. We further observe that all the languages considered exhibit low intelligibility with such languages as COBOL, Groovy and Shell. This means that when COBOL, Groovy or Shell programmers leave, finding their replacement among programmers versed in other languages in Table 3.1 might be problematic. For COBOL one could argue that the low values can be explained by under-representation of legacy technologies on SO. This is, however, highly unlikely for Groovy, an object-oriented programming language first released in 2007. Low mutual intelligibility values of other languages with COBOL, Groovy and Shell contrast sharply with more easily replaceable

[4] `http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html`
[5] `http://en.wikipedia.org/wiki/List_of_programming_languages`
[6] `http://lang-index.sourceforge.net`
[7] `http://www.win.tue.nl/~bvasiles/languages/list.html`

developers in such languages as C, C++, HTML or Java. As expected, the table also shows a high degree of asymmetry. For instance, 63% of Groovy programmers know Java but only 1% of Java programmers know Groovy (not surprising since Groovy has been developed for Java, but constitutes only a minor fraction of the overall Java development).

Although we are measuring different things (similarity by usage in case of Karus and Gall [146] and Delorey *et al.* [73] versus mutual intelligibility or similarity by knowledge in our case), we expect that similarity by usage implies similarity by knowledge, since languages used together by the same person are likely to be known together by that person. Our results partly support the findings of Karus and Gall [146] (strong relation between XML and Java—34%, and limited evidence for C/C++ and XML—13% and 11%, respectively). Our results also support the findings of Delorey *et al.* [73], who observed strong relations between JavaScript and PHP ($\Rightarrow$ 31%, $\Leftarrow$ 33%), Java and JavaScript ($\Rightarrow$ 19%, $\Leftarrow$ 20%), C and C++ ($\Rightarrow$ 48%, $\Leftarrow$ 32%), and C and Perl ($\Rightarrow$ 8%, $\Leftarrow$ 25%).

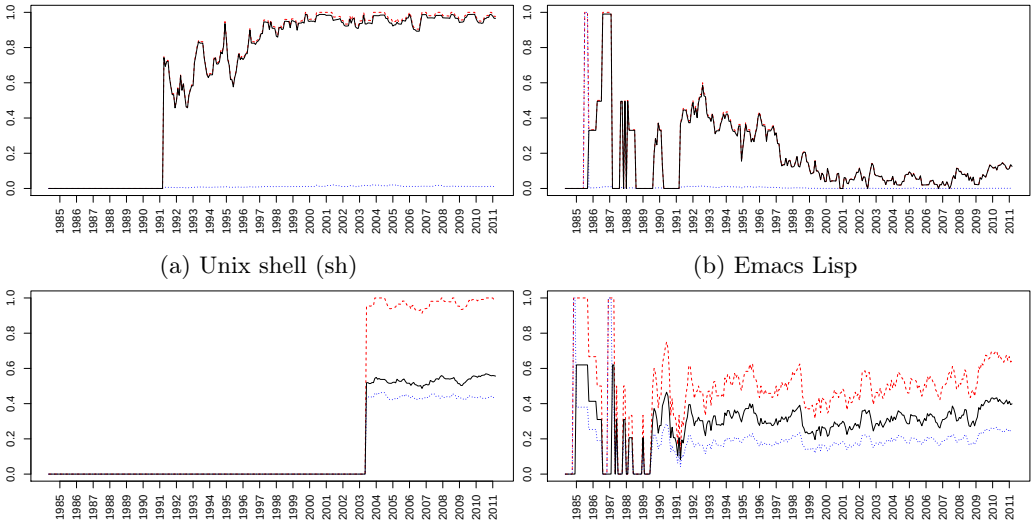
3.5 Illustration of the approach

To illustrate our approach in a real-world context, we performed a case study on Emacs [281], a popular text editor in development since the mid-1970s.

We identified 446 different (*name*, *email*) pairs in the *author* field for each change recorded in the Git log, corresponding to 27 years of Emacs development (1985-2012). Since there were multiple email addresses associated with the same names, and multiple names associated with the same email addresses, we performed identity merging [163, 320] (see also Chapters 2 and 7), after which 369 unique identities remained. To track the evolution of our risk measure throughout the evolution of Emacs, we extracted the programming languages used. We analysed the filename extensions of all the source files mentioned in the Git log. After filtering out files without extensions (mostly related to documentation), configuration files, make files, documentation files, and auxiliary files (*e.g.*, used by the version control system), we uncovered the following 26 different programming languages: Assembly, Awk, Bash, Bison, C, C++, Cocoa, C shell, Emacs Lisp, Grammar, HTML, Java, Lex, Lisp, M4, Objective-C, Pascal, Perl, Prolog, Python, RelaxNG, Unix shell, SRecode, Termcap, Windows Batch, and XML. Next, we estimated the development community from which replacement developers can be sought, at one-month intervals. To filter out inactive contributors, at each point in time (*e.g.*, February 2002), we considered that the community (per programming language) consisted of those developers who performed at least one change to a source code file (implemented in that language) in the past six months (*e.g.*, between September 2001 and February 2002). Finally, we computed $risk_\ell$ at one month intervals.

We discuss four representative examples (Figure 3.1). Detailed plots for all 26 languages are available in the online appendix[8]. We start with Unix shell (Figure 3.1a). The risk measure and the percentage of non-speakers are very close, *i.e.*, evolution of the risk measure can be explained predominantly by the evolution of the percentage of non-speakers. The increasing trend observed from 1991 to 2000, followed by the stabilisation from 2000 onwards reflects the diminishing proportion of Unix shell developers. Moreover, very low values of the dotted blue line indicate that Unix shell is not commonly known by developers programming in popular languages (*e.g.*, Emacs Lisp and C). Emacs Lisp (Figure 3.1b) exhibits similarly close values of the risk measure and the percentage of

[8] http://www.win.tue.nl/~bvasiles/emacs/risk.html



(a) Unix shell (sh)

(b) Emacs Lisp

(c) Python

(d) C

Figure 3.1: The risk measure $risk_\ell$ (black solid line), the share of the community that does not speak ℓ (dashed red line), and the difference between the two (dotted blue line) for Unix shell, Emacs Lisp, Python, and C.

non-speakers, but both values are low (below 0.2 starting from 1998). The lion's share of the development community is, hence, familiar with Emacs Lisp. In contrast, Python (Figure 3.1c), although spoken by a similarly small fraction of the Emacs community as Unix shell (dashed red line), exhibits much lower risk (black line). The high values for the difference between the two time series (dotted blue line), relatively stable in time, indicate that Python is commonly known by developers programming in popular languages. Indeed, $mi_{Python}(EmacsLisp) \simeq 0.46$, $mi_{Python}(Lisp) \simeq 0.44$, and $mi_{Python}(C) \simeq 0.23$. C (Figure 3.1d) is spoken by approximately half of the Emacs community and is also commonly known by developers programming in Lisp (0.39) and Emacs Lisp (0.38), resulting in very low risk. Emacs Lisp exhibits a similar pattern, with the difference that its low risk is mostly due to the large share of Emacs Lisp speakers within the community rather than high similarity with the other languages.

3.6 Conclusions

Inspired by linguistic diversity measures, we proposed a method to quantify the risk of not finding developers who can maintain code implemented in a certain programming language, and empirically illustrated it using a case study. Our method takes into account similarities between programming languages, for which we have proposed a novel measure based on shared knowledge of the developers participating in Stack Overflow. By tracking the evolution of such a risk measure as projects evolve (*e.g.*, in a dashboard-like application), risky languages can be discovered on time, and preventive action can be taken to ensure the maintainability of components implemented in those languages.

We believe the results obtained so far to be a promising start. Our new dimension to risk assessment, bordering software maintenance and the social sciences, *does* offer additional insights into the evolution of a software system, and *does* open up many avenues for future research. For example, to offer a more complete understanding of evolution, our risk model should be refined to incorporate the number of artifacts in a certain programming language (the risk seems higher if a large proportion of a system is implemented in a risky language), their role (examples may be less important than the core implementation), their stability as reflected in a version control system (files not changed for a long time seem less risky than recently changed ones, cf. [231]) or their algorithmic or linguistic complexity (files implementing more complex behaviour or using more exotic language features seem to be more risky). We also plan to include a more refined language-tag mapping such that tags corresponding to versions and dialects (*e.g.*, `c#-2.0` and `swi-prolog`) can be accounted for. A further refinement would include distinction between tags representing different technologies (*e.g.*, `ejb`, `hibernate` and `swing`). Apart from being all implemented in Java, such technologies share little in common and likely require different skills to maintain.

We also would like to introduce a project-level risk measure $risk_P$, being the maximum of $risk_\ell$ for all languages ℓ in the project P, indicating the highest risk of not having developers who can take over code implemented in a certain language. Similarly, for a given project P we can identify and rank developers that—should they decide to leave P—would contribute most to increase of $risk_P$. This ranking can be seen as an alternative interpretation of the "bus factor" [107] and a way to quantify the developers' contributions [46].

Beyond the boundaries of *linguistic diversity* is the general concept of diversity, and its measurement in several biological, physical, social, and management sciences [224]. Some of these techniques have recently been applied in context of software engineering as well [207, 262, 324]. A detailed comparison of these techniques with the risk measure proposed in the current paper is considered as future work.

Chapter 4

Working rhythms across communities

Stack Overflow *is a popular on-line programming question and answer community providing its participants with rapid access to knowledge and expertise of their peers, especially benefiting coders. Despite the popularity of* Stack Overflow*, its role in the work cycle of open-source developers is yet to be understood: on the one hand, participation in it has the potential to increase the knowledge of individual developers thus improving and speeding up the development process. On the other hand, participation in* Stack Overflow *may interrupt the regular working rhythm of the developer, hence also possibly slow down the development process.*

In this chapter we investigate the interplay between Stack Overflow *activities and the development process, reflected by code changes committed to the largest social coding repository,* GitHub*. Our study shows that active* GitHub *committers ask fewer questions and provide more answers than others. Moreover, we observe that active* Stack Overflow *askers distribute their work in a less uniform way than developers that do not ask questions. Finally, we show that despite the interruptions incurred, the* Stack Overflow *activity rate correlates with the code changing activity in* GitHub.

4.1 Introduction

Developers create and maintain software by standing on the shoulders of others [285]: they reuse components and libraries, and go *foraging* on the Web for information that will help them in their tasks [36]. For help with their code, developers often turn to programming question and answer (Q&A) communities, most visible of which is Stack Overflow[1] (SO) [191]. To engage its participants to contribute more, Stack Overflow employs gamification [74]: questions and answers are voted upon by members of the community; the number of votes is reflected in a person's *reputation* and *badges*; in turn, these can be seen as a measure of one's expertise by potential recruiters [45] and are known to

[1] `http://stackoverflow.com`

motivate users to contribute more [74]. By *asking questions* on Stack Overflow, developers can seek help and advice from their peers, *e.g.*, about their own code snippets or about undocumented technology features [223]. By *answering questions* posed by others, developers can share their knowledge and expertise, help and educate others, or compete in the "game" to achieve higher reputation.

The analogy of Stack Overflow as an effective educational institution asserts itself then. The extended effect of education, beyond the immediate edification, is to accelerate or catalyse societal advances. Does Stack Overflow have the same effect on software development communities? The connection between developer productivity and their using of Stack Overflow is not well-understood. On the one hand, Stack Overflow is known to provide good technical solutions [223] and to provide them fast [191], to the extent that closer integration between Q&A websites and modern IDEs is now advocated [18, 57]. On the other hand, as an exponent of social media, using Stack Overflow may lead to interruptions impairing the developers' performance [285], especially when gamification is factored in.

In this chapter we investigate the interplay between asking and answering questions on Stack Overflow and committing changes to open-source GitHub[2] repositories. GitHub is arguably the largest social coding site [296], hosting more than three million software projects maintained by over one million registered developers. The two platforms overlap in a knowledge-sharing ecosystem: GitHub developers can ask for help on Stack Overflow to solve their own technical challenges; similarly, they can engage in Stack Overflow to satisfy a demand for knowledge of others, perhaps less experienced than themselves. By identifying GitHub users active on Stack Overflow and studying their activities on both platforms, we can study if a connection exists between their participation in Stack Overflow and their productivity on GitHub. That is,

Goal: Is participation in Stack Overflow related to productivity of GitHub developers?

Here, following Adams *et al.* [5], we look at only one, but representative, facet of developer productivity: the number of commits made by developers in a given time period. Clearly, commits can be of different length and quality, and thus their number is insufficient to quantify the total contribution of a developer to a project, or even to quantify their energy expenditure while doing so. However, it is a reasonable representative, or sample, of the overall activities a developer undertakes while working on a project.

As complex relationships are best understood when looked at from different angles and at a range of resolutions, we examine the relationship between Stack Overflow and GitHub at three different levels. At the *macro-level*, we look at the time-aggregate (overall) activities over the two platforms (questions, answers and commits) of developers active on both. At this level we would like to identify differences between GitHub contributors in terms of their involvement in Stack Overflow, and understand whether some groups of developers benefit more from participation in SO than others. Indeed, GitHub users are a mix of novice and professional programmers [65]. While it is known that foraging is common for novices and experts alike [36], their *diets* are different [86], with potentially different impact on their performance. Similarly, different roles can be

[2] `http://github.com`

identified among Stack Overflow participants based on the quantity and quality of their questions, answers, and comments [234]. However, for both platforms, such roles are identified using only information about the activities of contributors *within* each platform. We would like to understand how such groups of developers relate *across* platforms, and to which extent activity (expertise) on one platform can be used as a proxy for activity (expertise) on the other. For example, such relations become immediately important when evaluating the reliability of social signals for career advancement [45].

> **RQ4.1:** (Macro level) How are the overall activity levels of developers related across the two platforms? *e.g.*, do active GitHub committers ask more or fewer questions on SO? Are more active answerers also committing more?

At the *intermediate level*, we attempt to capture the distribution of work units over time vis-a-vis people's participation on Stack Overflow. We are interested in understanding how developers distribute their time over commits, given their amount of Q&A activity. Do different groups of developers exhibit different working rhythms (*e.g.*, do active SO answerers, presumably more experienced, work differently than less active ones)? Bursts of intense commit activity followed by long periods of inactivity would suggest more focused attention at any given time, while a more egalitarian distribution of inter-commit time intervals would suggest a more steady but less focused working rhythm, where attention at any given time is divided between the two platforms. Working rhythms of developers are known to influence the quality of the software [87]. Specifically, we answer the following research question:

> **RQ4.2:** (Intermediate level) Are the commit (work) patterns on GitHub of developers more active on SO different than the commit patterns of those developers less active on Stack Overflow?

Lastly, at the *micro-level*, we associate GitHub commits and Stack Overflow questions and answers over time. We wish to understand whether activities in the two platforms show signs of coordination, *i.e.*, whether the rate of asking or answering questions on SO is related to the rate of commit activities in GitHub. Specifically, we ask:

> **RQ4.3:** (Micro level) Is there a functional interaction, or coordination between commit and question/answer activities? *i.e.*, when commits are close to Q&A in time, are they more frequent? How about vice-versa?

The remainder of this chapter is organised as follows. After reviewing the related work in Section 4.2, in Section 4.3 we discuss how the data has been obtained and prepared. We distinguish between the macroscopic (*e.g.*, are heavy GitHub committers also heavy Stack Overflow users?), intermediate (*e.g.*, how do Stack Overflow activities affect the working rhythm of the developers?) and microscopic (*e.g.*, are Stack Overflow activities occurring in lockstep with GitHub commits?) views and discuss them in Sections 4.4, 4.5 and 4.6, respectively. Finally, we summarise our contribution and sketch directions for further research in Section 4.7.

4.2 Related work

The abundance of information to which developers are exposed via social media is changing the way they collaborate, communicate, and learn [45, 65, 285], thus ultimately impacting the way they write software. Specifically, Stack Overflow is known to cover numerous software engineering topics and attract numerous software developers. Popularity of Stack Overflow among software developers has lead to increased interest from the research community as well [312]. However, the productivity implications of Stack Overflow remain unclear.

On the one hand, it can be argued that participating in Stack Overflow leads to *interruptions that could impair a developer's performance* [285]. Indeed, Bacchelli et al. [18] and Cordeiro et al. [57] argue that the current lack of integration between Q&A websites and modern IDEs forces developers to interrupt their flow and change context every time they need to deal with them, thus delaying their activity. Xuan et al. [346] argue that social communication activities (such as asking or answering Stack Overflow questions) may delay programming activities, since both of these activities compete for the time resources of developers. Indeed, it is well known that "a wealth of information creates a poverty of attention and a need to allocate that attention efficiently among the overabundance of information sources that might consume it" [271].

On the other hand, it can be argued that participating in Stack Overflow *speeds up development activities* as quick solutions to technical challenges are provided, thus saving the developers precious time. Mamykina et al. [191] show that most Stack Overflow questions are answered in a median time of 11 minutes. Parnin et al. [223] argue that Stack Overflow is a better source of API documentation, while Brandt et al. [36] propound that by relying on information and source code fragments from the Web, developers more effectively distribute their cognition, allowing them to devote more energy to higher-level tasks.

Moreover, both Stack Overflow participation and software development in public GitHub repositories can be seen as social activities. Stack Overflow evaluates the participant's contribution in terms of reputation points and badges, that allow participants to access new features and gain more control on others' postings. In this way, Stack Overflow encourages the participants to ask "good" questions and to give "good" answers [45]. Similarly, heavy GitHub users are aware of being watched by their peers, and this awareness influences how they behave and construct their actions, for example, by making changes less frequently [65].

4.3 Data preparation

To study the interplay between communication and code commit activities we integrate information extracted from two sources: Stack Overflow and GitHub. In this section we discuss how the data has been obtained and merged. All data used in this study as well as the tools developed are available for replication on `http://www.win.tue.nl/mdse/stackoverflow/`.

4.3.1 Extraction

All public data in Stack Overflow, including the list of members and the history of their activity, can be downloaded in XML format as part of the Stack Exchange data dump[3]. Data dumps are released every three months under the Creative Commons license. Here we explore the one released in August 2012, containing information about 1,295,622 registered users since July 2008 until August 2012.

The GitHub data comes from GHTorrent [112], a service that gathers event streams and data from GitHub and provides that data back to the community in the form of incremental MongoDB data dumps[4]. The GitHub data set contains information about 397,348 users and 10,323,714 commits, most from the July 2011 to April 2012 period.

4.3.2 Preprocessing

Git commits contain information about both the *author* (the person who originally changed the code) and the *committer* (the persons who last applied the change), each with their own timestamp. The two are not necessarily one and the same person (*e.g.*, they can differ when someone *rebases*[5] or *cherry picks*[6] a commit). In this chapter we consider only the commits which record the same person as both author and committer (97.8% of the commits in our data set), and record the date at which a commit was *authored* (rather than *committed*).

In addition, git allows commit metadata, including the authorship date, to be overwritten. For instance, we conjecture that commits with the 1969-12-31 or 2050-07-18 timestamps underwent such a history rewriting process. Therefore, we restrict our study to the period July 2011 to April 2012 (depicted in Figure 4.1) which contains the bulk of the commits in GHTorrent (approximately 99%).

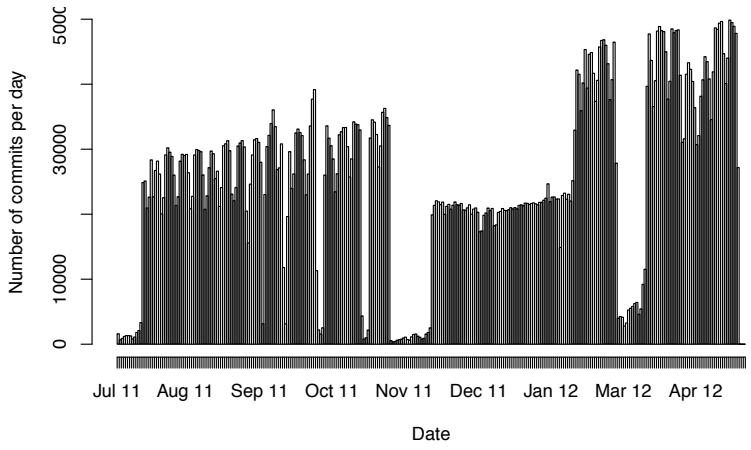


Figure 4.1: Number of commits per day in the GitHub data set between July 2011 and April 2012.

[3] http://www.clearbits.net/torrents/2076-aug-2012
[4] Accessible via https://github.com/gousiosg/github-mirror
[5] http://www.kernel.org/pub/software/scm/git/docs/git-rebase.html
[6] http://www.kernel.org/pub/software/scm/git/docs/git-cherry-pick.html

Table 4.1: Sizes of the original and intersection datasets.

Dataset	Number of users (in intersection; active)
GitHub	397,348 (23.6%; 11.8%)
Stack Overflow	1,295,622 (7.2%; 3.6%)
Intersection	93,771
Intersection (active)	46,967

Finally, the GHTorrent authors acknowledged that bugs in their extraction process led to some duplicate commits being recorded. We ignore duplicate commits, *i.e.*, commits authored by the same person and having the same timestamp.

4.3.3 Intersecting the two datasets

A key step in our process is merging the GitHub and Stack Overflow datasets, *i.e.*, identifying those contributors which are active on both platforms. Merging aliases used by the same person in different software repositories is a well-known problem [28, 106, 163] (Chapter 7). For example, Bird et al. [28] try to match full names or email addresses shared by different aliases, and use heuristics to "guess" email prefixes based on combinations of name parts (*e.g.*, *jsmith* and *John Smith*). Kouters *et al.* [163] (Chapter 7) use Latent Semantic Analysis (LSA), a popular information retrieval technique, and report better results in presence of very noisy data. However, all existing approaches are known to produce false positives and false negatives [106].

To limit the number of false positives[7], we follow a more conservative approach and make use of email addresses. In the GitHub data set email addresses are present. In the Stack Overflow data set email addresses are obscured, but their MD5 hashes are available. Therefore, we decide to merge (*i.e.*, link) a GitHub and a Stack Overflow user if the computed MD5 hash of the former's email address is identical to the MD5 email hash of the latter. Table 4.1 presents basic statistics about the two datasets, before and after intersecting. More advanced approaches to identity merging, *e.g.*, that also take into account names or email prefixes (cf. [320], Chapter 2), are considered as future work.

As a result of this process, approximately one quarter of the GitHub users (23.6%, or 93,771) are linked to Stack Overflow. However, it is possible that not all users in the GitHub & Stack Overflow intersection have authored at least one commit on GitHub between July 2011 and April 2012 (see the discussion above). Similarly, it is possible that not all users in the GitHub & Stack Overflow intersection have actively participated in Stack Overflow by asking or answering during the same period. Therefore, we further require users to have been active on both platforms, hence we filter out those users that neither authored any commits, nor asked or answered any question between July 2011 and April 2012. This further reduces the size of the intersection data set to 46,967 users (or 11.8% of the GitHub data set).

[7]The accuracy of the identity merging algorithm cannot be estimated in the absence of an "oracle" (*i.e.*, the absolute truth) against which to compare the results. Such an oracle does not exist for the two datasets.

4.4 Macroscopic view

To study how GitHub committing reflects Stack Overflow activities we start by taking the macroscopic view and studying distributions of the number of events of each type (C for commit, Q for question, A for answer). For each ordered pair of event types (*e.g.*, (C, Q)), with the data sorted along one of the dimensions (*e.g.*, C), we split the other dimension (*e.g.*, Q) into multiple groups and compare the resulting distributions. We performed experiments with our groups being quartiles and deciles but report only the results obtained for quartiles, since splitting into deciles yielded similar results.

This "split-and-compare" approach was chosen over the traditional statistical approaches of comparing correlation, like the correlation coefficient and regression modeling, because the latter are only capable of detecting monotonic relations (*e.g.*, "high number of commits corresponds to high number of questions and low number of commits corresponds to low number of questions", or "high number of commits corresponds to low number of answers and vice versa"). The "split-and-compare" approach, when used with an appropriate statistical testing procedure, as we do below, can *also* detect non-monotonic relationships (*e.g.*, "both the low and the high number of commits correspond to high number of questions, while if the number of commits is neither too high nor too low, the number of commits is low").

4.4.1 Comparing multiple distributions

Traditionally, comparison of multiple groups follows a two-step approach: first, a global null hypothesis is tested, and then multiple comparisons are used to test sub-hypotheses pertaining to each pair of groups. The first step is commonly carried out by means of ANOVA or its non-parametric counterpart, the Kruskal-Wallis one-way analysis of variance by ranks. The second step uses the t-test or the rank-based Wilcoxon-Mann-Whitney test, with Bonferroni correction. Unfortunately, the global test null hypothesis may be rejected while none of the sub-hypotheses are rejected, or vice versa [95]. Moreover, simulation studies suggest that the Wilcoxon-Mann-Whitney test is not robust to unequal population variances, especially in the case of unequal sample sizes [40]. Therefore, one-step approaches are preferred: these should produce confidence intervals which always lead to the same test decisions as the multiple comparisons.

To this end, we employ the recently-proposed multiple contrast test procedure $\widetilde{\mathbf{T}}$ [158] using the traditional 5% family-wise error rate. $\widetilde{\mathbf{T}}$ is robust against unequal population variances and is applicable to different types of contrasts, including comparisons of all pairs of distributions, the so called *Tukey-type contrasts*. For Tukey-type contrasts, we summarise the results of $\widetilde{\mathbf{T}}$ by means of $\widetilde{\mathbf{T}}$-graphs [320] (Chapter 2). In such a directed acyclic graph, nodes correspond to the different groups being compared, and edges to the results of the pairwise comparisons. There is an edge from A to B if A tends to have higher values for a given metric than B (*i.e.*, for the comparison A–B, $\widetilde{\mathbf{T}}$ reports $p < 0.05$). Since $\widetilde{\mathbf{T}}$ respects transitivity, in a $\widetilde{\mathbf{T}}$-graph we omit direct edges between A and B if there is a path from A to B passing through at least one other node. Consider the example $\widetilde{\mathbf{T}}$-graph from Figure 4.2, summarising the results of the $\widetilde{\mathbf{T}}$ procedure applied to four groups of values A, B, C and D: D tends to have higher values than

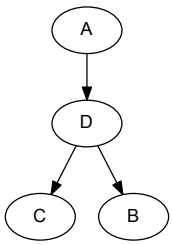
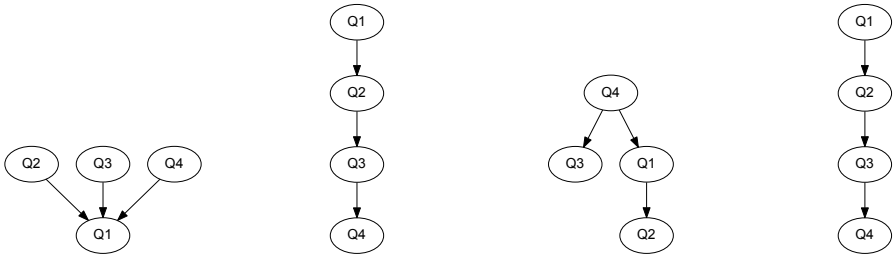


Figure 4.2: Example $\widetilde{\mathbf{T}}$-graph.



(a) Q1 committers ask fewer questions on SO than any of the others.

(b) More active committers answer more questions on SO.

(c) Q4 askers commit more than any of the others. Q1 askers commit more than Q2 ones.

(d) More active answerers author more commits on GitHub.

Figure 4.3: Macroscopic view of activity levels of users active on both GitHub and Stack Overflow [July 2011–April 2012].

both B and C, but lower than A; A tends to have higher values than all other groups (D directly, B and C transitively).

4.4.2 Results

Are heavy committers also heavy question askers?

With the data sorted along the C dimension in decreasing order, we split the Q dimension into quartiles and compare the resulting groups pairwise. The results (Figure 4.3a) reveal that the most active 25% of the committers (Q1) ask fewer questions on Stack Overflow than any of the other quartiles, but Q2, Q3 and Q4 are virtually indistinguishable from each other in terms of their Q activity. This suggests that active GitHub committers are experienced developers that do not need much technical advice: they perform numerous commits without asking much for help on Stack Overflow.

> Active GitHub committers ask fewer questions on Stack Overflow than others.

Are heavy committers also heavy answer givers?

With the data sorted along the C dimension in decreasing order, we split the A dimension into quartiles and compare the resulting groups pairwise. The results (Figure 4.3b) reveal a perfect ordering: more active committers provide more answers (*e.g.*, Q2 developers answer more questions than any of Q3 or Q4, but fewer than Q1). This suggests that GitHub activity can be seen as a proxy for one's willingness to answer technical questions on Stack Overflow, or one's level of expertise. When further put into the context of gamification, this finding suggests that top users on Stack Overflow are "superstars" rather than "slackers" [56]: they don't just compete for reputation and badges, but are actually active software developers.

More active GitHub committers provide more answers on Stack Overflow.

Are heavy question askers also heavy committers?

With the data sorted along the Q dimension in decreasing order, we split the C dimension into quartiles and compare the resulting groups pairwise. The results (Figure 4.3c) reveal a non-monotonic relation between Q and C that could not have been revealed by traditional correlation techniques. One the one hand, the least active askers (the Q4 users with the fewest questions asked) author more commits than any of the others. This observation is in line with its complement above: the most active committers ask the least. On the other hand, Q1 askers (most active) commit more than Q2 ones, suggesting an active learning process in which seeking answers to technical challenges is accompanied by experimenting with the proposed solutions, and committing the changes to GitHub. However, this conjecture will have to be further investigated.

The least active askers author more commits than others. However, more active askers are not indistinguishable in terms of their commit activity: the most active askers commit more than the second most active ones.

Are heavy answer givers also heavy committers?

With the data sorted along the A dimension in decreasing order, we split the C dimension into quartiles and compare the resulting groups pairwise. The results (Figure 4.3d) reveal another perfect ordering: more active answerers commit more. This observation is in line with its complement above: more active committers provide more answers. This suggests that answering questions on Stack Overflow can be seen as a proxy for one's commit activity.

More active Stack Overflow answerers make more commits on GitHub.

Summary

We find a *direct relationship between* GitHub *commit activity and* Stack Overflow *question answering activity*: the more active a committer, the more answers she gives; similarly, the more active an answerer, the more commits she authors. In contrast, we find an *inverse relationship between* GitHub *commit activity and* Stack Overflow *question asking activity*: active GitHub committers ask fewer questions than others; less active question askers produce more commits. Overall, these findings suggest that an activity-based ranking of Stack Overflow contributors reflects one extracted from their open-source contributions to GitHub, increasing the confidence in the reliability of SO-based social signals (*e.g.*, heavy SO answerers tend to be also very active GitHub committers).

4.5 Intermediate view

In the macroscopic view we have ignored when commits, questions and answer occur, restricting our attention solely to the number of events. Next we refine the approach and include information about the time intervals separating subsequent events. Following Xuan *et al.* [346], we define a *working rhythm* of an individual in a given activity (committing, asking/answering questions) as determined by a series of interactivity times: $\Delta t_i = t_{i+1} - t_i$, where t_i is a time-stamp of the i'th activity instance (commit, question, answer). Specifically, in this section we focus on committing rhythms.

4.5.1 Methodology

We are interested in understanding how developers distribute their time over commits, *i.e.*, whether or not they are following a steady working rhythm. To evaluate the committing rhythm of a developer, we calculate the Gini index over the lengths of her inter-commit time intervals. The Gini index is a popular econometric measure designed to study inequality of income or wealth distributions; it is often being used to aggregate software metrics, *e.g.*, [311, 324]. The Gini index values range over $[0; 1 - \frac{1}{n}]$, where n is the number of values being aggregated: Gini index equal to zero would indicate an egalitarian distribution of developers' time over commits, *i.e.*, developers following a steady working rhythm; Gini index close to the maximum would correspond to one big inter-commit time interval and numerous small inter-commit time intervals. Since the number of inter-commit intervals significantly varies from one developer to another, we normalise the Gini index values by dividing them with $1 - \frac{1}{n}$.

Similarly to Section 4.4, we split the individuals into quartiles depending on the number of questions asked or answers given on Stack Overflow. Then, we compare the normalised Gini values computed for the time series of GitHub inter-commit intervals, for the individuals associated with each quartile. This helps us understand whether different groups of developers exhibit different working rhythms. For example, active Stack Overflow answerers (shown above to be also active committers), presumably more experienced, might work differently than less active ones. To compare multiple distributions (each quartile generates a distribution of Gini index values), we follow the methodology described in Section 4.4.1.

4.5.2 Results

We have first used the number of questions as a basis for grouping the developers into quartiles. However, the median equals 0, *i.e.*, half of the developers did not ask any questions, and we can no longer distinguish between $Q1$ and $Q2$. Hence, we compare three distributions of GitHub inter-commit time intervals: normalized Gini index values of the individuals that do not ask questions $Q12$, that ask few questions $Q3$, and that ask the most questions $Q4$. Using the $\widetilde{\mathbf{T}}$ procedure we conclude that the normalized Gini index values are higher for active askers than for developers that do not ask questions ($p = 0.013$), *i.e.*, active askers distribute their effort in a less egalitarian way than developers that do not ask questions. In other words, developers who ask many questions on Stack Overflow commit changes to GitHub in bursts of intense activity followed by longer periods of inactivity, *i.e.*, they focus their attention at any given time. Specialization (or focus) of developers has also been noted previously in the context of activity types (*e.g.*,

coding versus translating) or files touched as part of a shared project [232, 320] (see also Chapter 2).

On the other hand, no differences can be observed between the normalised Gini index values for individuals grouped into quartiles based on the number of answers given on Stack Overflow. Therefore, *asking* questions on Stack Overflow influences how developers distribute their time over commits on GitHub, while *answering* questions does not seem to have the same effect. This observation is in line with our previous conjecture on developers learning from Stack Overflow and committing their experiences to GitHub, as well as the literature on the impact of social media on software development [65, 285].

> Active Stack Overflow askers distribute their work in a less egalitarian way (*i.e.*, focus their attention more) than developers that do not ask questions.

4.6 Microscopic view

So far we have ignored the ordering between commits, questions and answers. The microscopic view takes this temporal aspect into account by considering committing, asking and answering as time-series. To study the interaction between activities we follow the approach proposed by Xuan et al. [346].

4.6.1 Interaction between the activities

Consider the timeline of GitHub and Stack Overflow activities of a particular developer (Figure 4.4a). Let $\mathcal{A}$ and $\mathcal{B}$ be two activities we would like compare (*e.g.*, C and Q). For every event $t_i^{\mathcal{A}}$ of $\mathcal{A}$ we measure the *evaluation latency*[8] $\epsilon_i^{\mathcal{B}}$ as the difference between the earliest event of $\mathcal{B}$ following $t_i^{\mathcal{A}}$ and $t_i^{\mathcal{A}}$, and the *response latency* $\rho_i^{\mathcal{B}}$ as the difference between $t_{i+1}^{\mathcal{A}}$ and the latest event of $\mathcal{B}$ preceding $t_{i+1}^{\mathcal{A}}$ (Figure 4.4b). The sequences $\epsilon^{\mathcal{B}}$ and $\rho^{\mathcal{B}}$ characterise the relationship between $\mathcal{A}$ and $\mathcal{B}$. Next, to study whether the sequence of $\mathcal{B}$ events for this particular developer could have occurred by chance, we create m random permutations of $\mathcal{B}$ events[9] $(\mathcal{B}_1, \ldots, \mathcal{B}_m)$. Reshuffling is done in such a way that the durations of the "idling periods" between two consecutive events of activity $\mathcal{B}$ are preserved, but the order of the "idling periods" is randomised (Figures 4.4c,d). Let $\epsilon^{\mathcal{B}_1}, \ldots, \epsilon^{\mathcal{B}_m}$ and $\rho^{\mathcal{B}_1}, \ldots, \rho^{\mathcal{B}_m}$ be series of evaluation and response latencies corresponding to $\mathcal{B}_1, \ldots, \mathcal{B}_m$ (Figure 4.4e).

Finally, we aggregate all the sequences $\epsilon^{\mathcal{B}}$ for the different developers into $E^{\mathcal{B}}$, all $\rho^{\mathcal{B}}$ into $P^{\mathcal{B}}$, etc. Then, we compare $E^{\mathcal{B}}$ with each one of $E^{\mathcal{B}_1}, \ldots, E^{\mathcal{B}_m}$ and $P^{\mathcal{B}}$ with each of $P^{\mathcal{B}_1}, \ldots, P^{\mathcal{B}_m}$. However, as opposed to Section 4.4.1, we are no longer interested in performing all pairwise comparisons between different groups, but in comparing one of the distributions ("control") against multiple alternatives ("treatments"). This kind of comparisons is known as a *Dunnett-type contrast*, and it is also supported by $\widetilde{\mathbf{T}}$. Hence, we apply $\widetilde{\mathbf{T}}$ for Dunnett type contrasts with $E^{\mathcal{B}}$ and $P^{\mathcal{B}}$ as control groups, and the traditional 5% family-wise error rate:

[8] For the sake of readability we use a lightly different notation than in the original paper [346].

[9] In our experiments we chose $m = 10$.

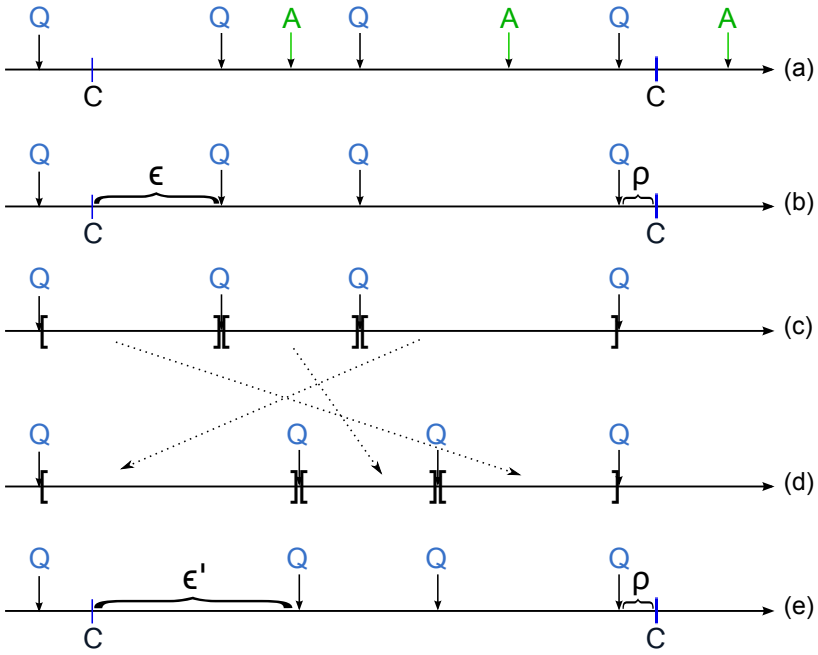


Figure 4.4: The steps to generate a simulated time-series of Stack Overflow activities.

- If $\mathcal{A}$ and $\mathcal{B}$ are independent from each other, $E^{\mathcal{B}}$ and $P^{\mathcal{B}}$ will be statistically indistinguishable from their simulated counterparts.
- If $\mathcal{A}$ delays $\mathcal{B}$, $E^{\mathcal{B}}$ will be statistically longer than the simulated evaluation latencies. Similarly, if $\mathcal{B}$ delays $\mathcal{A}$, $P^{\mathcal{B}}$ will be statistically longer than the simulated response latencies.
- If $\mathcal{A}$ accelerates $\mathcal{B}$, $E^{\mathcal{B}}$ will be statistically shorter than the simulated evaluation latencies. Similarly, if $\mathcal{B}$ accelerates $\mathcal{A}$, $P^{\mathcal{B}}$ will be statistically shorter than the simulated response latencies.

To address the potential inconsistencies between the $\widetilde{\mathbf{T}}$ results for the m randomisations, we apply the following schema. We say that two activities *do not influence* each other (denoted "none") if at most one of the simulations resulted in a statistically significant comparison. Otherwise, we speak of *acceleration* (*delay*) if at least 80% of the simulations have been found to indicate acceleration (delay). In all other cases we say that the influence is *inconclusive*.

Since we focus on the impact of Stack Overflow activities (Q, A) on GitHub committing (C) and vice versa, we always choose GitHub committing as one of the activities and vary a different Stack Overflow activity as the other one.

4.6.2 Results

To investigate whether Stack Overflow activities impact only specific groups of committers (*e.g.*, those very active), we split the committers into quartiles based on their

Table 4.2: Mutual influence of Stack Overflow activities and GitHub committing, for different committers (from least active $Q1$, to most active $Q4$).

Q	Influence of			
	asking on committing	committing on asking	answering on committing	committing on answering
$Q1$	none	none	none	none
$Q2$	none	inconclusive	inconclusive	inconclusive
$Q3$	accelerates	accelerates	accelerates	accelerates
$Q4$	accelerates	accelerates	accelerates	accelerates

Table 4.3: Mutual influence of Stack Overflow activities and GitHub committing, for different answerers (from least active $Q1$, to most active $Q4$). Individuals in $Q1$ do not give answers.

Q	Influence of			
	asking on committing	committing on asking	answering on committing	committing on answering
$Q1$	accelerates	accelerates	n/a	n/a
$Q2$	none	none	none	none
$Q3$	accelerates	accelerates	none	none
$Q4$	accelerates	accelerates	accelerates	accelerates

total *number of commits* (as in the previous sections). Results of our investigation are summarised in Table 4.2. First of all, we observe that the results are consistent. Moreover, for the most active half of the committers ($Q3$ and $Q4$), the real latencies consistently tend to be lower than the simulated ones. This suggests that for these developers, committing and asking questions accelerate each other, as well as committing and answering questions accelerate each other.

For active committers, asking questions on Stack Overflow catalyses committing on GitHub. Similarly, for active committers, answering questions on Stack Overflow catalyses committing on GitHub.

Similar differences in influence of Stack Overflow activities on GitHub committing between more and less active developers can be observed after grouping by *length of involvement* in GitHub (the catalysis is more visible for individuals who have been involved in GitHub for sufficiently long time), or *number of questions* asked on SO (the catalysis is more visible for active askers).

Finally, when grouping is done according to the *number of answers* given on SO, slightly different results are obtained (Table 4.3). By answering questions, there is a benefit

only for the most active answerers ($Q4$): answering questions on Stack Overflow and committing changes to GitHub accelerate each other. In contrast, by asking questions, acceleration is visible for both the active answerers ($Q3$ and $Q4$) as well as for the developers that do not answer any questions at all (exclusive askers or exclusive knowledge seekers; $Q1$): asking questions on Stack Overflow and committing changes to GitHub accelerate each other.

> For the most active answerers as well as for developers that do not answer any questions at all, their Stack Overflow activities accelerate their GitHub committing.

4.7 Conclusions

In this chapter we studied the relationship between question & answer activities carried out by individuals on Stack Overflow and their contributions to GitHub repositories. Our findings are based on the data for 46,967 users active both on Stack Overflow and GitHub.

First, we focussed on differences in Stack Overflow involvement of the GitHub developers (RQ4.1). We observed that individuals who tend to ask few questions tend to have a high number of commits, and individuals with a high number of commits tend to ask few questions. Moreover, individuals that tend to answer many questions tend to have a high number of commits, and individuals that have a high number of commits tend to answer many questions. This suggests that highly productive (in terms of GitHub commits) individuals tend to take the role of a "teacher" more actively involved in providing answers rather than asking questions.

Next, we studied whether the working rhythm of the GitHub contributors is related to their Stack Overflow activities (RQ4.2). We observed that individuals that tend to ask many questions distribute their work in a less uniform way than developers that do not ask questions at all. No differences were observed between the work distributions for individuals grouped based on the number of answers given.

Then, we showed that despite interruptions incurred, for active GitHub developers Stack Overflow activities are positively associated with the social coding in GitHub (RQ4.3). Similar observations hold for active askers as well as individuals who have been involved in GitHub for sufficiently long time. Finally, Stack Overflow activities accelerate GitHub committing also for the most active answerers as well as for developers that do not answer any questions at all.

To deepen our understanding of the impact Stack Overflow has on GitHub we intend to expand the research presented as follows. First of all, we would like to refine the classification of activities: we plan to distinguish between questions and answers pertaining to different subjects (as expressed by Stack Overflow tags) and commits pertaining to different projects. Then, using information retrieval techniques we intend to classify questions and answers as being related, or not, to a given commit. For instance, we expect to observe a closer relation between commits and the topics of questions asked compared to the relation between commits and the topics of answers given, as answers are more likely to pertain to the general knowledge of the individual. As a continuation of

our work on the impact of the number of questions on the working rhythms, we intend to study to what extent can the inequality in the inter-commit time intervals' distribution be explained by different aspects of the GitHub projects and their developers (including Stack Overflow activities of the latter ones). To measure the explanation we intend to employ the Theil index [59, 262]. Moreover, we plan to investigate the impact of the committing rhythm of the individual on her activity rhythm on Stack Overflow: are questions being asked or answered when no committing is done, or rather interleaved with commits? We also would like to augment our study of the intermediate view by applying further models of inter-event time distribution [20, 345] to our data. Finally, to obtain additional insights in the combined Stack Overflow & GitHub activities we would like to apply process mining techniques originally developed for information systems [338] and successfully applied to traditional software repositories such as version control systems, mail archives and bug trackers [231].

Chapter 5

Transition to gamified environments

Historically, mailing lists have been the preferred means for coordinating development and user support activities. With the emergence and popularity growth of social Q&A sites such as the Stack Exchange *network (*e.g.*,* Stack Overflow*), this is beginning to change. Such sites offer different socio-technical incentives to their participants than mailing lists do,* e.g.*, rich web environments to store and manage content collaboratively, or a place to showcase their knowledge and expertise more vividly to peers or potential recruiters. A key difference between* Stack Exchange *and mailing lists is gamification,* i.e.*,* Stack Exchange *participants compete to obtain reputation points and badges. In this chapter, we use a case study of R (a widely-used tool for data analysis) to investigate how mailing list participation has evolved since the launch of* Stack Exchange*. Our main contribution is the assembly of a joint data set from the two sources, in which participants in both the* `r-help` *mailing list and* Stack Exchange *are identifiable. This permits their activities to be linked across the two resources and also over time. With this data set we found that user support activities show a strong shift away from* `r-help`*. In particular, mailing list experts are migrating to* Stack Exchange*, where their behaviour is different. First, participants active both on* `r-help` *and on* Stack Exchange *are more active than those who focus exclusively on only one of the two. Second, they provide faster answers on* Stack Exchange *than on* `r-help`*, suggesting they are motivated by the* gamified *environment. To our knowledge, our study is the first to directly chart the changes in behaviour of specific contributors as they migrate into gamified environments, and has important implications for knowledge management in software engineering.*

5.1 Introduction

Historically, mailing lists have been the preferred medium for coordinating development and user support activities [118, 273, 277]. In particular, mailing lists have been viewed as the *de facto* communication medium between *knowledge seekers* (*e.g.*, users of the software asking for support) and *knowledge providers* (*e.g.*, other users, more knowledgeable about

the topic, or the developers themselves) in models of knowledge sharing in open source [277]. The two categories of knowledge actors have been reported to co-exist in a symbiotic relationship, wherein "the community learns from its participants, and each individual learns from the community" [277]. However, their motivations for participation may differ. For instance, knowledge seekers may directly benefit from having their problems solved, while knowledge providers may be motivated intrinsically (*e.g.*, by altruism), or by learning about the problems experienced by other users [170, 277].

Recent years have witnessed the emergence and growing popularity of software-development-related social media sites, such as GitHub[1] (coding), Jira[2] (issue tracking), or the Stack Exchange network (question and answer websites, *e.g.*, Stack Overflow for "professional and enthusiast programmers,"[3] or Cross Validated for "statisticians, data analysts, data miners and data visualization experts"[4]). Such sites are rapidly changing the ways in which developers collaborate, learn, and communicate among themselves and with their users [24, 45, 65, 272, 285]. Moreover, they are offering different socio-technical incentives to their participants, *e.g.*, rich Web 2.0 platforms to store and manage content collaboratively, or a place to showcase their knowledge and expertise more vividly to peers and potential recruiters [45]. In addition, Stack Exchange sites employ *gamification* [75] to engage users more: questions and answers are voted upon by the community; the number of votes is reflected in the poster's *reputation* and *badges*; exceeding various reputation thresholds grants access to additional features (*e.g.*, moderation rights on topics and posts); reputation and badges can also be seen as a measure of one's expertise by potential recruiters [45], and are known to motivate users to contribute more [11, 12, 74, 354]. Activity on Stack Exchange sites can also elevate one to celebrity status within the developer community (see, *e.g.*, the discussion around Jon Skeet[5], the most prolific contributor to Stack Overflow).

Naturally, the richer user interfaces, wider audiences, or different incentives and motivations for participation inherent in social Q&A sites are challenging the supremacy of mailing lists as the *de facto* communication medium between knowledge seekers and knowledge providers. For example, Stack Overflow is known to provide good technical solutions [223] and to provide them fast [191]. At the same time, mailing list participants are signalling the need for more modern support,[6,7] and are even promoting a transition to Stack Exchange.[8] Our goal here is to study in detail the effects of such a transition on contributors and their work. Are mailing list participants transitioning to Stack Exchange? If so, do they behave differently on Stack Exchange than on the mailing list?

To study the phenomena associated with such a transition, we need a *longitudinal data set* combining mailing list and Stack Exchange activity, wherein participants overlap. In this chapter we create such a data set for R [238], a popular data analysis software, by integrating mailing list activity and Stack Exchange activity (the latter under the `[r]` tag). Using this data set, we find that:

[1] `https://github.com`
[2] `http://www.atlassian.com/software/jira`
[3] `http://stackoverflow.com`
[4] `http://stats.stackexchange.com/`
[5] `http://meta.stackoverflow.com/q/9134`
[6] `http://tolstoy.newcastle.edu.au/R/e10/help/10/06/8782.html`
[7] `http://tolstoy.newcastle.edu.au/R/e16/help/11/11/0902.html`
[8] `http://benmazzotta.wordpress.com/2010/07/06/r-goes-to-stackexchange/`

- activity on `r-help` (the main user support mailing list for R) has been consistently decreasing since around 2010 (*i.e.*, participants are asking fewer and fewer questions), while at the same time the number of R-related questions asked on the two main Stack Exchange sites for R (Cross Validated and Stack Overflow) has been accelerating.
- participants in the two media overlap, but different categories of `r-help` contributors are "attracted" differentially by Stack Exchange. For example, the proportion of mailing list users active on Stack Exchange is much higher among R developers than non-developers.
- the levels of activity for `r-help` participants who are also active on Stack Exchange differ relative to those who restrict themselves to either the mailing list or to Stack Exchange: those participating in *both* `r-help` and Stack Exchange are more active.
- knowledge providers active on both media answer questions significantly faster on Stack Exchange than on `r-help`, and their total output increases after the transition to Stack Exchange.

Apart from uncovering interesting phenomena, these findings reveal that knowledge management in open source is changing, and that the different socio-technical incentives offered by Q&A sites such as Stack Overflow enhance participation in these sites, facilitate user contributions and foster productivity. Therefore, our findings could inspire start-up open source or commercial projects looking to establish user support platforms, or stakeholders interested in knowledge management in general.

The rest of the paper is organised as follows. We first focus on the background underlying knowledge-sharing in open source and present our research questions. Then, we describe the data set and data gathering process, followed by our methods, results, and concluding sections.

5.1.1 Background

The importance of user support for the adoption, growth, and success of open source projects is well-recognized [19, 170, 289]. Traditionally, user support was organised through mailing lists, forums, user groups, etc. However, as new venues of information and tools for information access emerge (*e.g.*, weblogs, wikis, social Q&A sites), people's online information seeking behaviour is also evolving [97, 264]. When it comes to user support activities around open source software, though, what is common among both traditional and new venues of information is that this "mundane but necessary task" [170] is typically carried out by unpaid volunteers. Often, the developers themselves take part in these activities, but thriving open source projects also succeed in enlisting some of their users to offer assistance to peers. But what motivates knowledge providers to answer other people's questions? Together with online information seeking behaviours, these motives are also evolving.

In traditional information-sharing venues, *knowledge-providers* participate for reasons related to, albeit different, than those of *developers* contributing to open source. Developers contribute for reasons such as: a direct need for the software; enjoyment of the work itself; or the enhanced reputation arising from high-quality contributions [170]. Knowledge-providers, on the other hand, are reportedly motivated by: learning about problems other users are experiencing; an enhanced feeling of being part of a community; personal benefits

of learning through teaching; an enhanced likelihood of receiving help in the future; or a sense of obligation from having received help from others in the past [170, 220].

As social Q&A sites, *e.g.*, the Stack Exchange network, are becoming more popular, instances of what the economic literature calls "signalling incentives" start to better explain what motivates knowledge providers to help others [174], in addition to the previous reasons. Career concerns (*e.g.*, online activities in social Q&A sites are more visible to employers and recruiters, who may use them to find qualified people [45]), or a desire for peer recognition (*e.g.*, reputation building) are among the listed motives [170, 174, 220]. However, such signalling incentives may not be equally applicable to all knowledge providers. For example, we can expect that more knowledgeable providers would draw greater benefit from signalling, and thus signal more. Therefore, to obtain a more fine-grained understanding of the transition from traditional information venues to social Q&A, it is important to distinguish between different groups of participants (*e.g.*, developers, with likely more knowledge about the software, versus non-developers).

Another dimension of social Q&A participation incentives arises from *gamification*, *i.e.*, using elements of game design in non-game contexts [74, 75]. Gamification has been widely used in online platforms in recent years, where it was shown to motivate users to contribute more [11, 12, 74, 354], and there are many psychological theories that can explain why [327]. The blueprint[9] that all current implementations of gamification (including all Stack Exchange sites) follow assumes stimulating users to perform a desirable activity by awarding them points. Specifically, users (i) are rewarded with points to encourage the desired behaviour (and may be subtracted points to sanction undesired behaviour); (ii) are awarded badges after collecting sufficiently many points or when performing certain activities; and (iii) have their progress tracked and their achievements displayed publicly in a leaderboard, to create competition between them. On Stack Exchange sites, the goal of "the game" is to have participants teaching and learning from each other[10] by asking relevant questions and providing helpful, well-documented and clear answers. Users receive reputation points when fellow users vote up their questions and answers. In turn, reputation points or directly generating *good* content translate to badges (*e.g.*, "Famous Question", if a question received 10,000 views, or "Legendary", if the user earned 200 reputation points daily 150 times). Top performing users in terms of their reputation count (either at week, month, quarter, year, or overall level) are displayed in public leaderboards. Exceeding various reputation thresholds grants users additional privileges on Stack Exchange sites, including at the higher end, the privilege to help moderate the site.

Therefore, coming back to our study of mailing lists, it seems that the different incentives, inherent to social Q&A sites, have the potential to "disrupt" mailing list activity and catalyze a transition to social Q&A, for knowledge seekers and knowledge providers alike. Seekers may turn to Stack Exchange expecting faster answers and a wider expert audience. Providers may transition to it to satisfy a rising demand for information, or pursuing signaling incentives and recognition. The latter is, for instance, supported by "alpha-male" behaviours self-reported by Apache help forum knowledge providers interviewed by Lakhani and von Hippel [170]: those wanting to be known as "the" expert in a particular aspect of Apache "would strive to answer all questions associated with their area" and seek to "drive out all other information providers from [that] chosen

[9] http://codingconduct.cc/Meaningful-Play

[10] http://bcove.me/34kz5iao

field of expertise". In the presence of gamification, such as on Stack Exchange sites, such behaviours may be even further accentuated. For example, Stack Overflow is said to suffer from the *fastest gun in the West problem*:[11] to maximise their chances of collecting up-votes from their peers, participants would race to answer questions as quickly as possible, rather than as correctly or as exhaustively as possible. The said problem is based on the anecdotal belief that given two answers of comparable quality, it is the earliest that would typically receive the most votes, or that one might refrain from answering altogether if someone else already offered a similar solution.

5.1.2 Research Questions

In this chapter, our goal is to analyze a mature and vibrant community where knowledge-transfer is transitioning from an older, mailing list modality to a new, social Q&A modality, and understand some of the effects of that transition. We chose the R software community.

> **Research Question 5.1:** Can we find evidence in the R community for a decreasing popularity of the mailing list and an increasing popularity of Stack Exchange?

RQ5.1 Discussion. R [238], a popular data analysis software, makes for an interesting case study since there have been initiatives by members of its community to promote a transition to Stack Exchange.[12,13] However, to quantitatively assess this phenomenon, we first need to construct a historical data set combining mailing list and Stack Exchange activity for R, and identify participant overlap (if any). Given such a data set, we can then evaluate activity in the two media, and look for transition phenomena. To the best of our knowledge, we are the only ones creating and analysing such overlapping data sets between Stack Exchange and open source communities [318] (Chapter 4).

In addition to the potentially different motivations driving participants, other factors may influence these phenomena. The R mailing lists have been around since 1997, while Stack Overflow, the first of the Stack Exchange sites, was only launched in 2008. The relative maturity of the former compared to the latter, as well as the direct contact with R developers, may mean that mailing lists are seen as well-established "educational institutions" by R users, *i.e.*, the default go-to place for requesting R support. Moreover, not all mailing list discussions are equally as well suited for Stack Exchange, hence not all are at risk of being subsumed. For example, Stack Exchange discourages questions that are too broad or primarily opinion-based.[14] Such questions may therefore only be suitable for the mailing lists. In other words, despite addressing some mailing lists limitations (*e.g.*, pertaining to user interface), Stack Exchange may not be a complete substitute.

On the other hand, R users asking questions on Stack Overflow may be given answers referring to earlier posts from `r-help`.[15,16] This suggests that mailing lists may not be as well indexed by search engines (hence solutions posted there may not appear

[11] http://meta.stackoverflow.com/q/9731/182512
[12] http://tolstoy.newcastle.edu.au/R/e10/help/10/06/8782.html
[13] http://benmazzotta.wordpress.com/2010/07/06/r-goes-to-stackexchange
[14] http://meta.stackoverflow.com/a/10582
[15] http://stackoverflow.com/a/1996404
[16] http://stackoverflow.com/a/4947528

as high up in the results list), or that the mindset of users (asking directly on a Stack Exchange site) is altogether changing in disfavour of the mailing lists. In addition to the factors above, the two knowledge-sharing media also show signs of symbiosis. For example, Stack Overflow discussions are being followed by R developers and bug reporters, who use the feedback received from this channel to stir up development discussions on `r-devel`,[17,18] the other main mailing list for R, dedicated to developers. The varying and synergistic activities of developers and knowledge-providers on mailing lists and Q&A sites suggest that some developers and knowledge-providers in R are splitting their time between the two, while others, perhaps, are not. Thus, R, a project with a knowledge-exchange in transition, includes some participants who stay with the old, some who are in transition, and some who go entirely with the new. It would be important to understand if this transition is in fact taking place, and to what degree. Community processes for answering user questions on-line are a vital component of the industry, and it is important to understand and promote this important process.

Research Question 5.2: How do the contributors active both on the mailing list and on Stack Exchange differ from those focused on a single medium (either of the two) in terms of their activity levels?

RQ5.2 Discussion. The different socio-technical incentives inherent in social Q&A (*e.g.*, related to user interface, potentially wider audiences, and gamification) may result in Stack Exchange becoming more attractive, *i.e.*, taking over (part of) the mailing list activity, and engaging (some of) the mailing list participants. Previous studies (*e.g.*, [45]) argue that one's activity in social media can be used as a signal of qualifications and, similarly, one's reputation in social media as a signal of references from peers. Are there mailing list participants active also on Stack Exchange sites? Who are they and how do they differ (in terms of their activity levels) from other mailing list or Stack Exchange participants, who choose to focus solely on the mailing list or solely on Stack Exchange? Are developers more likely to be active both on `r-help` and on Stack Exchange (*i.e.*, to "signal" more) than non-developers?

Research Question 5.3: For the contributors active both on the mailing list and on Stack Exchange, can we find differences in their behaviour in one platform versus the other?

RQ5.3 Discussion. If Stack Exchange sites are attracting mailing list participants (*e.g.*, knowledge seekers looking for faster answers or a wider expert audience, or knowledge providers looking to satisfy a demand for knowledge, "signal" their "fitness" as experts, or engage in the game for reputation and badges), we should observe a decreasing trend in the mailing list activity concomitant with an increasing trend in Stack Exchange activity for those participants (especially knowledge providers) active on both. Moreover, gamification, one of the key differences between `r-help` and Stack Exchange sites, might influence `r-help` participants active on Stack Exchange to adopt a different

[17] `http://tolstoy.newcastle.edu.au/R/e12/devel/10/10/0002.html`
[18] `http://stackoverflow.com/a/1321491`

Source	Period	#messages #threads	#participants #unique	Multiple aliases
r-help	4/1997–	344,854	33,338	≃15%
	3/2013	97,125	28,096	
Stack Overflow [r]	9/2008–	67,248	10,534	≃2%
	3/2013	24,957	10,284	
Cross Validated [r]	7/2010–	7,351	2,312	≃1%
	3/2013	3,208	2,285	

Table 5.1: Basic statistics for the two data sources.

behaviour on Stack Exchange [11, 12, 74, 354]. For example, in order to "survive" in the game for reputation and badges, they might have to provide faster answers than they do on the mailing list. Can we observe such a trend? Do participants active both on r-help and on Stack Exchange behave differently when on the mailing list than when on Stack Exchange sites? If answers are indeed quicker, and the speed is somewhat attributable to "game-playing", this might confirm that gamification is a useful adjunct in the design of Q&A fora.

5.2 Methodology

As case for our case study we selected R [238], a popular data analysis software, for several reasons: First, R is a typical example of an open-source software ecosystem, comprising a (relatively) closed *core of developers providing the basic functionality* and coordinating new releases, various developers *contributing patches and bug fixes*, numerous developers *contributing packages* (plugins) that extend the functionality beyond that provided with each release, and a *plethora of users*. Other examples of such ecosystems include the Eclipse IDE and its third-party plugins, or the Python/Ruby/LaTeX programming languages and their various contributed packages/gems. Second, R has been evolving for almost 20 years, and its entire history of mailing list communication is archived and publicly available. Third, R has been the recent subject of an extensive study of its evolution [100]. Fourth, R promises to provide broader relevance outside the software developers' community, as many users and contributors are data analysts from different domains such as economics or biology, with none or limited software engineering experience.

5.2.1 Data extraction

We integrated data from two different sources: the community behind the main R user support mailing list (r-help), and the Stack Exchange R subcommunities behind Stack Overflow and Cross Validated, the Stack Exchange websites containing the most R-related ([r]-tagged) questions and answers.

r-help is the principal mailing list for user support in R. It hosts discussions about problems and solutions using R, as well as announcements about the availability of new functionality and documentation. Moreover, it mirrors announcements from r-packages on new or enhanced contributed packages, or major developments from r-announce. The other major mailing list for R is r-devel, targeting developers, testers and bug

reporters, with topics that are considered "too technical" for `r-help`'s audience. The `r-help` archives can be downloaded in standard mbox format.[19] We wrote Python scripts to automatically download, extract, and parse the archives, which date as far back as April 1997.

For each `r-help` participant we recorded their role: (i) *core developer*, *i.e.*, the 22 developers with "write access to the R source" since its inception;[20] (ii) *peripheral developer*, *i.e.*, the 43 developers who "contributed by donating code, bug fixes and documentation";[20] (iii) *package author/maintainer*, *i.e.*, the 2617 developers maintaining or having authored packages on CRAN, the largest R package repository;[21] (iv) *user*, *i.e.*, those not fitting in any of the previous categories.

Stack Exchange is a network of Q&A sites started in 2008, now comprising more than 100 sites on different topics, such as English language, video gaming, photography, or parenting. All have similar look and feel, and function by the same principles: participants can ask and answer questions; questions are organised by tags (*e.g.*, `[r]` for R-related questions) and voted upon by users, with votes translating to reputation points and badges; questions and answers can be edited and improved by other members with sufficiently high reputation; each participant has a dedicated profile page, combining the representation of oneself (self-determined, *e.g.*, name, location, or personal website) with activity-related information (automatically provided, *e.g.*, a list of the answers provided, the total reputation count, or a list of badges); users can subscribe to tags and receive email notifications when new questions are being asked (less practical for high-volume tags such as `[c#]`, `[java]`, or even `[r]`, the latter receiving around 500 questions per week at the time of writing), or simply browse the website.

Stack Exchange releases quarterly data dumps in XML format from all the websites under its umbrella. We explored the one dated March 2013,[22] using Python scripts to parse the XML archives. We restricted our analysis to Stack Overflow (a generic programming Q&A site, and the first and largest Stack Exchange member) and Cross Validated (dedicated to statistics). This left out a number of other Stack Exchange websites where questions tagged `[r]` occasionally (*i.e.*, very infrequently) pop up, such as TeX or GIS, as well as all the activity on Stack Overflow and Cross Validated which did not occur within the `[r]` tag. The oldest Stack Overflow questions tagged `[r]` are dated September 2008. Cross Validated hosts `[r]`-tagged questions dating as far back as July 2010. The `[r]` activity on both websites mainly targets R users as opposed to R developers (hence the comparison to `r-help` is sound), although R developers may also follow it, as discussed in the introduction.

The organization of Stack Overflow and Cross Validated in terms of questions, answers, tags, badges and reputation points is the same, with the only difference being the target audience, programmers in the former vs. data analysts in the latter. Therefore, we expect that differences between Stack Overflow and Cross Validated should not affect data collection, analysis and interpretation. Table 5.1 lists some basic statistics about the data sources.

[19] `http://www.r-project.org/mail.html`

[20] `http://www.r-project.org/contributors.html`

[21] Comprehensive R Archive Network, `http://cran.r-project.org/web/packages/available_packages_by_date.html`

[22] `http://www.clearbits.net/torrents/2121-mar-2013`

5.2.2 Identity merging

One of the biggest challenges when mining software repositories is *identity merging* [28, 106, 163, 320] (Chapter 7). Both within a single repository (*e.g.*, mailing lists), as well as across repositories (*e.g.*, mailing lists and Stack Exchange), the same person may use different aliases, *i.e.*, different (*name, email*) tuples. For instance, *John Smith* may go by (*John Smith, johnsmith@gmail.com*), (*John, john@smith.com*), etc. The extent of the problem is unpredictable, *e.g.*, one of the Gnome developers reportedly used 168 different aliases in the source code repository [320] (Chapter 2).

A solution is to merge identities, and existing approaches are very diverse. For example, Bird *et al.* [28] try to match full names or email addresses shared by different aliases, and use heuristics to "guess" email prefixes based on combinations of name parts (*e.g.*, *jsmith* is likely to belong to *John Smith*). Kouters *et al.* [163] (Chapter 7) use Latent Semantic Analysis (LSA), a popular information retrieval technique, and report better results in presence of very noisy data. However, all existing approaches are known to produce false positives and false negatives [106]. We followed different approaches for `r-help` and for Stack Exchange.

Stack Exchange The email addresses of the Stack Exchange participants are not publicly available for privacy reasons, but their MD5 hashed versions are offered instead. Since it is highly unlikely for two different email addresses to share an MD5 hash, we performed identity merging on the Stack Exchange data if two MD5 hashes coincided. This process resulted in a reduction in the number of participants of approximately 2% on Stack Overflow and 1% on Cross Validated.

We decided to limit the identity merging *solely* to the case above due to the following reasons. First of all, on Stack Exchange as opposed to the mailing lists users have *accounts* and can log in using a password. This suggests that the incidence of multiple aliases (*i.e.*, accounts) by the same person should be much lower than on the mailing lists, where one is more likely to send an email, *e.g.*, from the account which is most at hand at any given time. While it is common for people to own multiple email accounts (*e.g.*, employer-related, open-source-related, private, etc.), we believe it to be less common for the same person to own multiple accounts on the same Stack Exchange site (at least because one cannot integrate the activity and reputation points earned using each). Moreover, since the email addresses are not publicly available, identity merging would rely solely on names, increasing the likelihood of false positives.

Mailing list For `r-help` we performed a fixed-point computation: for each email address, we (i) collected all other email addresses having the same prefix (*e.g.*, for *john.smith@gmail.com* we collected *john.smith@hotmail.com*), as long as the prefix did not consist of a single word (*i.e.*, contained either a dot or an underscore character); and (ii) collected all the different names associated with it (*e.g.*, *John Smith* and *John* if both used the email address *john.smith@gmail.com*). Then, for each of these names we collected the different email addresses with which they were associated (*e.g.*, *John Smith* might be associated with both *john.smith@gmail.com* and *johnny@gmail.com*), and repeated the process until a fixed-point was reached. Checks for a minimal length of the email prefixes and names were used to limit the number of false positives. Applying this technique resulted in a reduction in the number of participants of approximately 15%. Among the users with the most aliases were, *e.g.*, an active participant with ten

different emails under the same name, or one of the peripheral R developers with two email addresses and nine different variations of his name.

5.2.3 Intersecting the two data sets.

A prerequisite for studying migration phenomena from the mailing lists to Stack Exchange was computing the overlap between the communities of participants in the two platforms. First, we made use of the fact that email addresses are available for the mailing list participants, as opposed to only MD5 hashes for the Stack Exchange users. As a result, we merged a mailing list user with one on Stack Exchange if the MD5 hash computed for any of the former's email addresses (there might have been multiple after identity merging on the mailing list) was identical to the MD5 email hash of the latter. Then, to expand the merges beyond only matching email addresses, we followed a fixed-point approach similar to the one described above, using names and email address prefixes (*e.g.*, *John Smith* from Stack Exchange with *John Smith* from `r-help`, despite the latter not having an email address that matches the former's MD5 hash). As a side effect, at this step two Stack Exchange users could have been merged to the same `r-help` individual, hence also among themselves (in addition to the merges using MD5 hashes discussed in the previous paragraph), if their MD5 hashes matched email addresses known to belong to this `r-help` individual.

Using this approach we found that approximately 15% (or 3,894) of the `r-help` unique participants (*i.e.*, after identity merging) have Stack Overflow accounts, and 5% (or 1,159) have Cross Validated accounts.[23] These numbers are influenced by the difference in age between the two media, since `r-help` started in 1997, while Stack Overflow only in 2008 and Cross Validated in 2010: part of the overall mailing list population is not active anymore, as observed also for other open source projects [29]. The size of the overlap increases (*e.g.*, to slightly over 20% for Stack Overflow) when considering only the `r-help` participants active starting with September 2008. Variations may also occur between different user groups (*e.g.*, knowledge seekers and knowledge providers). For example, the activity of open source contributors typically follows very skewed distributions [320] (Chapter 2), *i.e.*, most have very few contributions. Hence, it seems more likely that active knowledge providers would participate in Stack Exchange than one-time knowledge seekers.

5.2.4 Comparing apples and oranges.

Our goals include understanding how the amount of activity differs between mailing lists and Stack Exchange, and whether mailing list participants are migrating to Stack Exchange. However, *activity* is organised in different ways in the two media. On Stack Exchange users ask questions and typically receive several answers from other members of the community. Answerers "compete" with each other at offering good answers, as reflected by the up votes received from other users. If a question was ill-formulated, *e.g.*, as signalled by other users through comments, the original poster has the option of editing and improving it any number of times. Similarly, if the answers were incomplete, the answerers or any other users have the option of updating them. Questions may remain unanswered. Acknowledgements for answers that solve the original problem can be made

[23] Having a Stack Exchange account does not guarantee having engaged in Q&A there, hence a smaller fraction will have actually been *active* on Stack Exchange.

by the original poster by *accepting* one answer using a dedicated button, and/or *posting comments* to the others (we do not consider comments). The standard structure of a Q&A post is therefore *one question followed by zero or more answers, typically offered by different people.*

The equivalent communication structure on mailing lists is represented by *discussion threads.* The first new message with a particular topic (subject) is the one starting the thread (*i.e.*, the question). Similarly to Stack Exchange, it can remain unanswered, or it can receive responses identifiable as messages sent In-Reply-To the original message, or to other messages within the same discussion thread. However, as opposed to Stack Exchange, the original poster and the answerers can all appear multiple times within a thread.

Therefore, to avoid comparing apples to oranges when comparing activity on the two communication media, we (i) grouped related email messages into discussion threads, operation known as *threading*; and (ii) discounted multiple answers by the same person, and discounted the original poster as answerer within the same thread. Due to numerous caveats, threading is a non-trivial operation, typically taken lightly in the literature. We used a slightly modified version of Zawinski's algorithm[24] (based on Kuchling's Python implementation),[25] to the best of our knowledge the leading publicly-available threading algorithm. The number of distinct threads obtained for `r-help` is displayed in Table 5.1. For the Stack Exchange websites, the number of threads is the same as the number of questions, by definition.

5.2.5 User survey

To better understand the context of our research, related to changes in information seeking and information providing behaviours in the R community, we augmented the quantitative study with a user survey,[26] used to triangulate the quantitative findings. We asked the participants for: background information about their occupation, experience with R and involvement in the development of R (*e.g.*, as core developers, package maintainers, users); their information seeking behaviour (*e.g.*, how frequently they need information, where and how they search for it, how satisfactory what they find is, what their preferred information seeking medium is, and whether anything has changed in their seeking behaviour over the years); and their information providing behaviour (how often and by what means they share information with others, and what motivates them to do so). Survey participants were recruited by posting an ad about the questionnaire on various channels, such as `r-help`, the Stack Overflow Meta (the site for meta-discussion of the Stack Exchange family of Q&A websites, where the maintainers of Stack Exchange also hang out), the Stack Overflow R chat room, Google groups, other French, Russian or German fora (to include also predominantly non-English-speaking R subcommunities), our own LinkedIn, Twitter, Google+ or Facebook profiles, or by contacting R developers directly.

[24] `http://www.jwz.org/doc/threading.html`

[25] `https://github.com/akuchling/jwzthreading`

[26] `https://docs.google.com/forms/d/14dXaSdy2NJ94Fq50fAbmEr3_GK8XfbRdys9qPLMMZ-c/viewform`

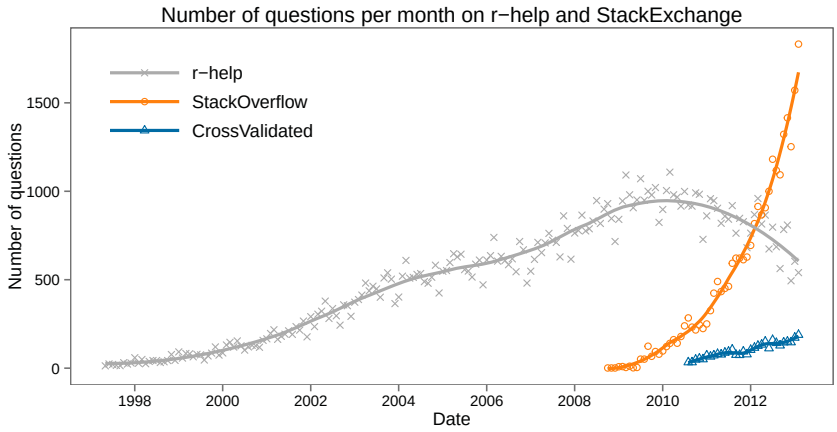


Figure 5.1: Number of questions asked (threads started) each month on `r-help` and Stack Exchange (Stack Overflow and Cross Validated) in the `[r]`-tag. The trend curves are Loess curves [54] with 0.5 span.

5.3 Results

5.3.1 Overall knowledge seeking activity

We start by analysing the activity on `r-help` (Figure 5.1). After an initial increase in the number of threads started (questions asked) each month, we observe drops in recent activity (since 2010). It is interesting to note that the 2010 inflection point is synchronised with initiatives to promote Stack Exchange among R users,[27] as illustrated by this excerpt from a blog entry:[28] "*Young experts don't want to have to monitor email all day to be part of the discussion. Their answers belong on a website with a normal content management system, with good search functions and user interactions. Go* [to Stack Exchange and] *sign up.*" On the other hand, `[r]` activity on Stack Exchange and in particular Stack Overflow (Figure 5.1) grows at an increasing rate. The fraction of `[r]` questions relative to all questions grows linearly[29] (*i.e.*, `[r]` gains popularity).

RQ5.1. Knowledge seeking on `r-help` decreases sharply since around 2010. At the same time, the number of R-related questions asked on Stack Overflow and Cross Validated grows at an increasing rate.

5.3.2 Structure of the community

Recall that we have distinguished between different roles within the `r-help` population. Here we wish to understand the knowledge seeking and knowledge providing activities of the different roles. Developers (mostly package authors/maintainers) are responsible

[27] `http://tolstoy.newcastle.edu.au/R/e10/help/10/06/8782.html`
[28] `http://benmazzotta.wordpress.com/2010/07/06/r-goes-to-stackexchange/`
[29] `http://hewgill.com/~greg/stackoverflow/stack_overflow/tags/#!r`

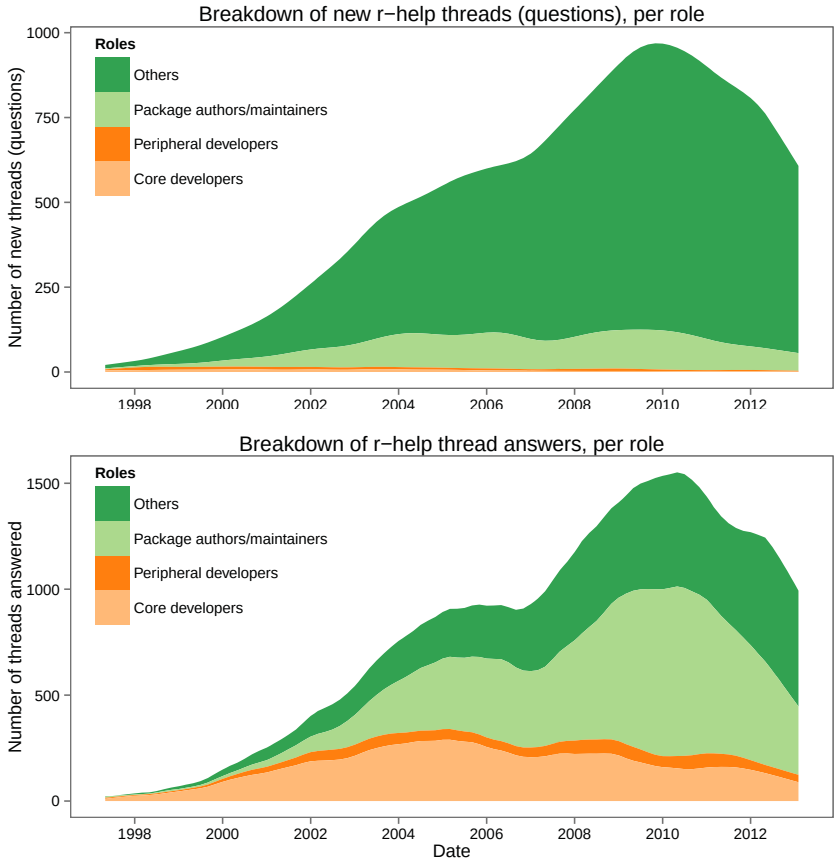


Figure 5.2: Breakdown of new threads (top) and thread answers (bottom) on `r-help` by initiator role. Only the *trend* component of each time series is displayed, after using a seasonal-trend decomposition procedure based on Loess [54]. The top plot corresponds to Figure 5.1.

for starting only a small fraction of the discussion threads (approximately one tenth in recent years), as depicted in Figure 5.2, top. Non-developers, the vast majority of question askers, are most responsible for the decreasing activity on `r-help` since 2010. The situation is reversed for thread answerers (Figure 5.2 bottom). The R core developers and the package authors/maintainers are responsible for most replies to threads started on `r-help`. However, since the decrease in new `r-help` threads started in 2010, the answering activity of developers (package authors/maintainers in particular) has also decreased the most. Both users and developers may be tempted by Stack Exchange, *e.g.*, one in search of better or faster answers, the other—of recognition.

Outcome	Devs (D)	Users (U)	χ^2 (p-val)	D-U (CI)	RR (CI)
Have SE accounts	331/938 (35.28%)	3,003/15,631 (19.21%)	141.28 (<2.2E-16)	.16 (.13, .19)	1.83 (1.67, 2.01)
Active on SE	183/331 (55.3%)	1,110/3,003 (36.9%)	41.39 (1.2E-10)	.18 (.12, .24)	1.49 (1.34, 1.66)

Table 5.2: Results of statistical testing for the Figure 5.3 cases. D-U (CI): Estimate and 95% confidence intervals for the difference of proportions. RR (CI): Risk ratio by unconditional maximum likelihood estimation and 95% confidence intervals using normal approximation.

5.3.3 Activity of contributors to both channels

In this section we focus on those mailing list participants who were also *active* on Stack Exchange. To control for the difference in age between `r-help` (started in April 1997) and Stack Exchange (started by Stack Overflow in September 2008), we only consider the mailing list participants starting with September 2008. While we were able to link approximately 20% (from September 2008, or 15% overall) of the `r-help` participants with Stack Exchange accounts (either Stack Overflow, Cross Validated, or both), we also observed that not all of them have actually engaged in `[r]`-tagged Q&A on the two Stack Exchange sites. Indeed, only 7.8% (or 1,293 out of 16,569) have asked or answered at least one `[r]`-tagged question on Stack Overflow or Cross Validated.

Different population subgroups again exhibit different behaviour (Figure 5.3): the fraction of participants with Stack Exchange accounts is much higher among developers (core, peripheral, or package authors/maintainers) than non-developers (left); within those with Stack Exchange accounts, developers are also more likely to have been active (*i.e.*, to have engaged in Q&A) than non-developers (right). The groups of core and peripheral developers are too small to enable statistically significant comparisons (*e.g.*, we linked only 8 out the 18 core developers active on `r-help` since September 2008 with Stack Exchange accounts; out of these, only 5 asked or answered at least one question on Stack Exchange). However, when considering the core, peripheral, and package developers together, the differences in joining and contributing to Stack Exchange between developers and the users become statistically significant and the effect sizes practically significant (Table 5.2). For example, from the second row in Table 5.2 we can see that 55.3% (or 183 out of 331) of the developers with Stack Exchange accounts are also active on Stack Exchange, as opposed to 36.9% (or 1,110 out of 3,003) of the users with Stack Exchange accounts. We are 95% confident that the population proportions of developers and users with Stack Exchange accounts who are also active on Stack Exchange differ by between 0.12 and 0.24 (the D-U column), with a lower proportion for users. Furthermore we are 95% confident that developers have between a 1.34 and 1.66 times higher chance of being active on Stack Exchange than users (the RR column).

5.3.4 The more things you do, the more you can do

Actively contributing to both media requires dividing one's time between them. Do mailing list participants contribute more or less to `r-help` if they are also active on Stack Exchange? To answer this question, we recorded for each `r-help` participant a

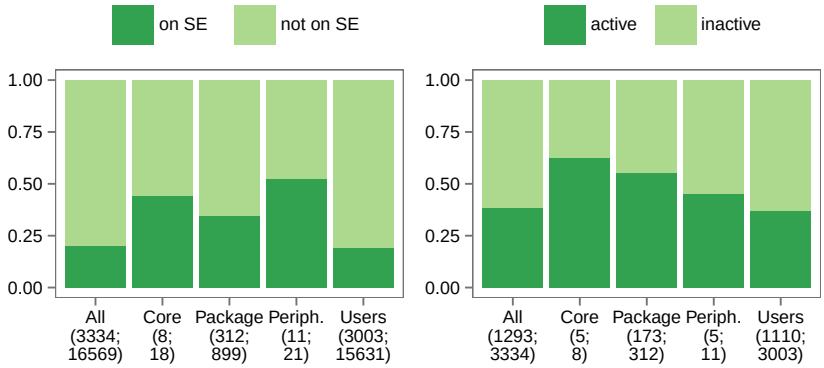


Figure 5.3: *Left:* Fraction of `r-help` participants (starting with September 2008) with Stack Exchange accounts, by role. *Right:* Among those with Stack Exchange accounts, the fraction that asked or answered at least one question. The absolute values are displayed under each label.

flag indicating whether or not he or she was active on Stack Exchange in the `[r]` tag, the number of threads started and the number of threads answered on the mailing list. To control for differences in age, we again restricted the counts to posts after September 2008. Depending on their `r-help` activity types, three groups of participants emerged (Table 5.3): (i) those who only started threads; (ii) those who only replied to threads; and (iii) those who both started and replied to threads. For each group we compared the amount of activity (number of threads started for (i) and (iii), and number of threads answered for (ii) and (iii)) between those active and those inactive on Stack Exchange using the Wilcoxon-Mann-Whitney test (since activity is not normally distributed). In all cases, the results reveal that the mailing list participants who also contributed to Stack Exchange are more active than those who did not. For example, the median number of threads answered is 4 among those who ask and answer on `r-help` *and* engage on Stack Exchange, compared to 1 for the others.

Next we compared the Stack Exchange activity for `r-help` participants also active there to that of the rest of the Stack Exchange `[r]` community: which were more prolific? Similarly, we distinguished between exclusive askers, exclusive answerers, and users who both asked and answered questions (Table 5.4). Similar comparisons using Wilcoxon-Mann-Whitney tests show that among the Stack Exchange `[r]` population, those who also participate in `r-help` are more active. For example, the median number of `[r]` answers for those who ask and answer on Stack Exchange *and* contribute to the mailing list is 3, as opposed to just 1 for the others.

These results show that the most prominent mailing list participants "signal" the most both on the mailing list and on Stack Exchange. The group of users who participate in the R mailing lists *and* engage in `[r]` Q&A on Stack Exchange are the most active contributors relative to the other mailing list participants and, similarly, the most active contributors relative to the other Stack Exchange participants. This result is in line with a recent observation made by Posnett *et al.* [234] in their study of expertise on Stack Exchange: "*expertise is present from the beginning* [of one's participation]*, and doesn't increase with time spent with the community. [...] In other words, experts join the*

Activity on r-help\SE	No account or inactive (N)	WMW comparison (p-val)	Active (A)
Only ask	11,724	Asking: A>N (4.7E-14)	790
Only answer	1,413	Answering: A>N (4.2E-3)	123
Ask, answer	2,139	Asking: A>N (2.83E-8)	380
		Answering: A>N (<2.2E-16)	
Total	15,276		1,293

Table 5.3: Activity comparisons for three groups of r-help participants also active on Stack Exchange relative to the other r-help participants (Wilcoxon-Mann-Whitney tests with Benjamini-Hochberg correction). Stack Exchange = Stack Overflow + Cross Validated.

Activity on SE \r-help	Inactive (N)	WMW comparison (p-val)	Active (A)
Only ask	5,464	Asking: A>N (5.25E-5)	794
Only answer	2,824	Answering: A>N (<2.2E-16)	338
Ask, answer	1,470	Asking: A>N (3.24E-12)	564
		Answering: A>N (<2.2E-16)	
Total	9,758		1,696

Table 5.4: Activity comparisons for three groups of Stack Exchange contributors also active on r-help relative to the other Stack Exchange participants (Wilcoxon-Mann-Whitney tests with Benjamini-Hochberg correction). Stack Exchange = Stack Overflow + Cross Validated.

community as experts, and provide good answers immediately." Assuming the number of answers one provides to be a proxy for their expertise, we showed, *e.g.*, that Stack Exchange experts (*i.e.*, heavy answerers) stem from mailing list experts. Recently we made a similar observation about GitHub developers active on Stack Overflow [318] (Chapter 4): those who provide the most answers are also those who perform the most commits.

RQ5.2. The more things you do, the more you can do: Contributors to both platforms (more likely developers than non-developers) are more active than those who focus on just one.

5.3.5 Behavioural differences

We compared the question answering activity on r-help for two groups of knowledge providers: those with (denoted "r-help and Stack Exchange") and those without (denoted "only r-help") Stack Exchange accounts (Figure 5.4). For each group we plotted the total number of answers given to r-help threads each month (denoted "On r-help"); for the "r-help and Stack Exchange" group, we also plotted the

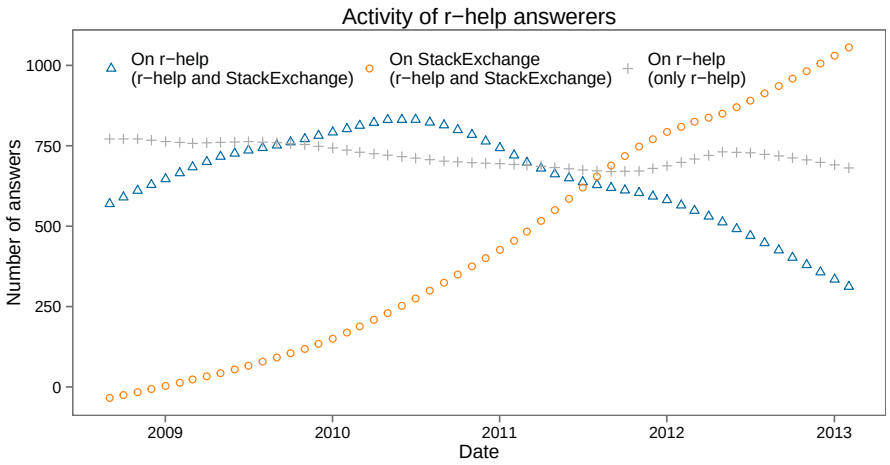


Figure 5.4: Number of questions answered on `r-help` (after September 2008) and Stack Exchange each month: participants exclusive to the mailing list versus those also active on Stack Exchange. Only the *trend* component of each time series is displayed, after using a seasonal-trend decomposition procedure based on Loess [54].

number of answers given to Stack Exchange questions each month (denoted "On Stack Exchange").

We draw the following observations. The inflection point (mid 2010) in the number of answers given to `r-help` threads by the "`r-help` and Stack Exchange" group coincides with the inflection point in general activity trend on `r-help`, depicted in Figure 5.1 top. This should not come as a surprise. In the previous sections we have seen that it is those mailing list participants that are also active on Stack Exchange that are most active, *i.e.*, the activity drivers or trend setters (both relative to the other mailing list participants as well as to the other Stack Exchange participants). Therefore, it is natural that they also exhibit a decreasing trend in activity since mid 2010. However, it is interesting to observe that the activity trend of the `r-help` knowledge providers without Stack Exchange accounts remains relatively constant (or decreases much slower). Corroborated with the increasing trend in answering activity on Stack Exchange by mailing list participants active on Stack Exchange, this suggests that R mailing list experts are migrating to Stack Exchange.

However, not all members of the "`r-help` and Stack Exchange" group exhibit similar patterns (Figure 5.5). For example, participant 3513 (topmost subfigure) *is* the norm: singlehandedly responsible for approximately one fifth of the total number of answers, he exhibits a similar decreasing trend in activity as the entire group. In contrast, participant 9440 (second topmost subfigure) has migrated entirely to Stack Exchange in the second half of 2010. Participant 209 (second bottommost subfigure) has been active on both `r-help` and Stack Exchange ever since the beginning, albeit not as intensively on Stack Exchange; over the years he has become increasingly disinterested in `r-help`, without becoming more interested in Stack Exchange. Finally, participant 9859

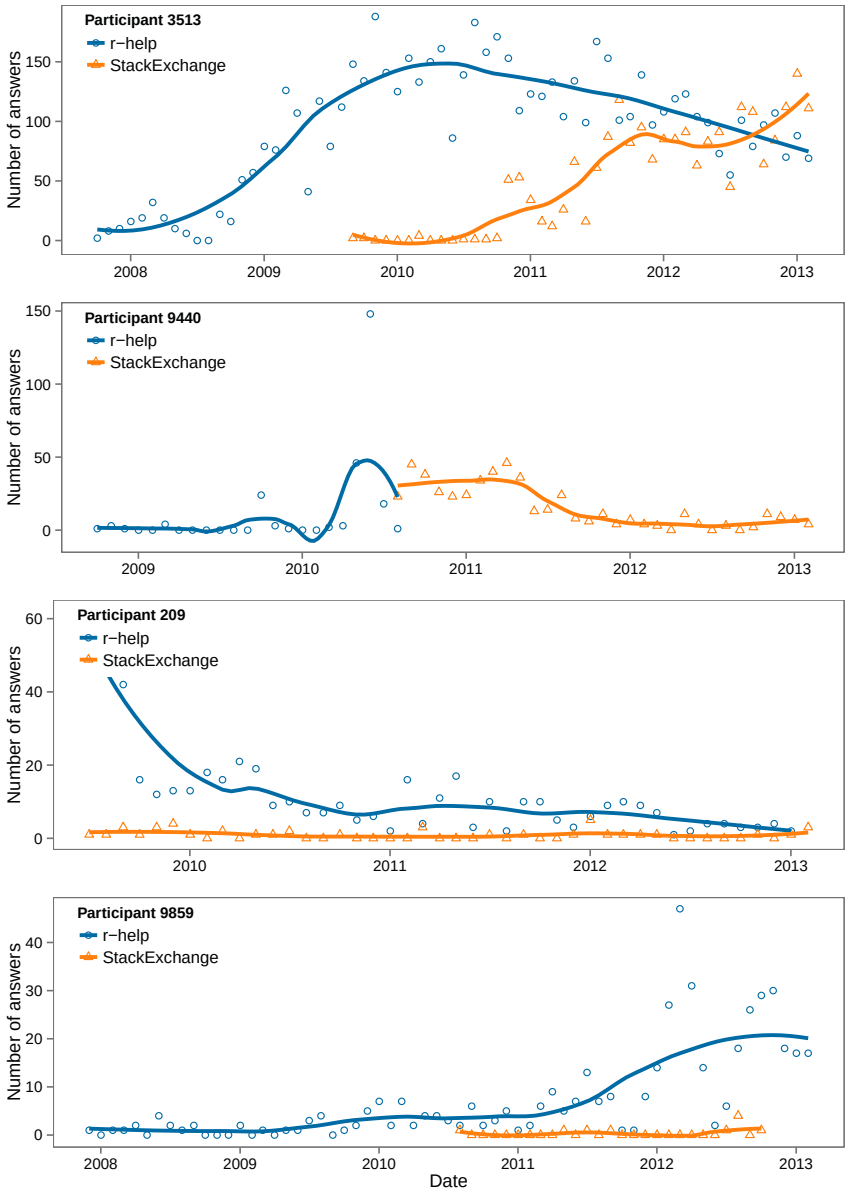


Figure 5.5: Different patterns of co-participation in `r-help` and Stack Exchange for knowledge providers (Loess curves [54] with 0.5 span).

(bottommost subfigure) is barely active on StackExchage, but is becoming increasingly more active on `r-help`.

Next we investigated whether there are significant differences between the speed with which `r-help` participants also active on Stack Exchange answer questions on `r-help` versus on Stack Exchange. If the incentives on Stack Exchange (*e.g.*, gamification,

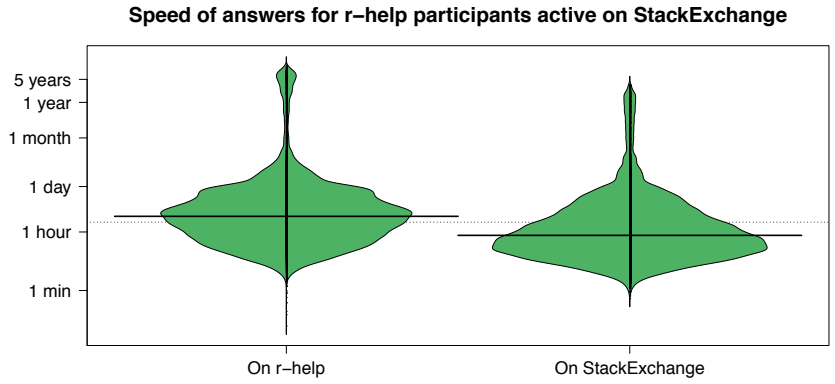


Figure 5.6: Answer speed for `r-help` users active on Stack Exchange. Beans: corresponding density shapes. Longer horizontal lines: medians per group. Dashed line: median over all groups.

more attractive user interface) do not influence behaviour, then we should not observe any meaningful differences. If instead users are engaging in the race for reputation and badges, then they might try to answer more questions (we have already seen this in the previous section) as well as answer questions faster on Stack Exchange than on the mailing list, where they are not "rewarded" for their haste. To test this hypothesis, we compute for each member of the "`r-help` and Stack Exchange" group the time intervals between their first answer within a thread and the thread start (for all threads for which they provide answers), on the one hand, and between their first answer to a Stack Exchange question and the question date (for all questions they answer), on the other hand. Then, using a Wilcoxon-Mann-Whitney test we compare two large groups of `r-help` and Stack Exchange time deltas obtained by concatenating the intervals for all "`r-help` and Stack Exchange" members (Figure 5.6).

Our results show that the Stack Exchange answers are significantly faster ($p <$ 2.2e-16), with a median of 47 minutes versus 3 hours on `r-help`. This confirms that `r-help` participants behave differently when on Stack Exchange, where they are rewarded for their efforts more. To further put these results into context, note that on mailing lists a knowledge provider is *passively* (automatically) provided with opportunities to answer questions (by receiving emails with new questions directly, or in the form of a digest); on Stack Exchange, although subscribing to tags (to receive notifications of new questions) is possible, knowledge providers will frequently *actively* pursue new questions being asked by browsing the site,[30] and will rush to answer them.

RQ5.3. Knowledge providers active both on `r-help` and on Stack Exchange (*i.e.*, the mailing list experts) are migrating to Stack Exchange, where they answer questions significantly faster than on the mailing list.

[30] `http://meta.stackoverflow.com/q/40927/182512`

5.3.6 Triangulation

In this subsection we compare the quantitative findings with the user survey results. We received 115 responses. One respondent participated twice, and two respondents did not consent to participate in the survey, leaving 112 valid responses, mostly from academics (32% of the respondents), statisticians (35%), students (14%), and software engineers (6%). Most respondents described themselves as R users (51%), or authors or maintainers of R packages (35%). We also received two responses from the core developers. We stress that the survey was not intended for quantitative analysis; rather, it served to triangulate the earlier findings.

Our first empirical finding was the evidence of a transition in seeking support from the mailing list to Stack Exchange: while activity on `r-help` decreases sharply since around 2010, the number of R-related questions asked on Stack Exchange grows at an increasing rate. In order not to impose our perception of transition on the survey participants, we opted for an open question, and asked whether the participants have experienced changes in their information seeking behaviour over the years. On the one hand, we observed that all survey participants reporting changes related to the mailing lists indicated disengagement, *e.g.*, "*Google is getting better at finding answers related to R so I use it more. I rely less on going directly to mailing lists now*" (user and documentation contributor, statistician/data analyst, 6 years of experience), or "*`r-help` used to be very helpful. But as the number of posts has gone up, I find that reading it is not as useful as it had been*" (package maintainer, academic, 8 years of experience). On the other hand, when the survey participants reported changes related to Stack Exchange sites, they are much more positive, *e.g.*, "Stack Exchange *has become more prevalent and more useful in the last two years*" (user, academic, 5 years of experience), or "*I started using RSeek.org, but currently I prefer stackoverflow.com, whose question database is increasing*" (user, statistician/data analyst, 6 years of experience). Still, these observations should be placed in the right context: more than half of the respondents are satisfied with the quality of the answers found in the mailing list archives, and more than a third will occasionally share their knowledge through mailing list discussions.

Our next empirical finding was that developers are more likely to be active on Stack Exchange than non-developers. To confirm it, we asked the survey participants what motivates them to contribute their knowledge. While numerous reasons such as reciprocity have been named both by the users and by the developers, developers are the only ones that mentioned Stack Exchange and gamification explicitly: "*In case of* Stack Exchange*, the reputation ratings are a nice little incentive,*" (academic, package maintainer, 6 years of experience), or "[I am motivated by] *peer recognition/gamification within* Stack Overflow" (academic, contributes to documentation, submits bugs, 5 years of experience). Moreover, developers put more stress on being motivated by the desire to enhance one's own reputation ("*wanting to be seen as a relative expert in some sub-domain*"; academic, contributes to documentation, submits bugs, 5 years of experience) and on evangelisation ("*I like to promote R because I think it's a great tool to learn about the principles of statistics and programming*"; academic, submits bugs, 4 years of experience). We believe that Stack Exchange is a better platform for these goals than mailing lists, since Stack Exchange quantifies one's reputation and makes it visible to everyone (see previous discussion of the gamification blueprint), as opposed to the mailing lists lacking commonly agreed upon and readily accessible representations of

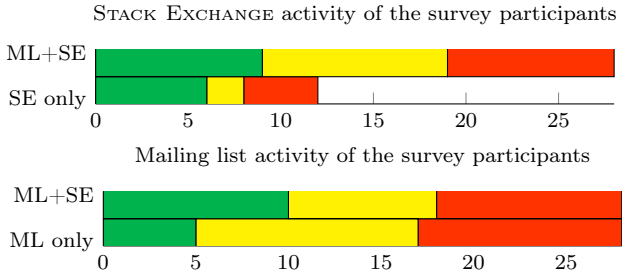


Figure 5.7: Number of survey participants frequently (green), occasionally (yellow) and rarely (red) sharing knowledge about R on Stack Exchange (SE) and mailing lists (ML).

reputation. Moreover, Stack Exchange is a multi-topic community, hence better suited to evangelisation than "preaching to the choir" on a mailing list.

The empirical study also showed that contributors active both on the mailing list and on Stack Exchange are more active than those focused on only one of the two. To triangulate this finding, we asked the survey participants how often they share information about R by replying to threads on R mailing lists, and by answering questions on Stack Exchange. Figure 5.7 shows that survey participants active on both media are more actively participating in the mailing list discussions than those active only on the mailing lists, *i.e.*, it supports the findings of the empirical study. A similar observation can be made for the Stack Exchange activity, although disparity between the numbers of survey respondents active only on Stack Exchange (12) and active on both media (28) hinders interpretation in this case.

Finally, in the empirical study we observed that knowledge providers active both on `r-help` and on Stack Exchange (*i.e.*, the mailing list experts) are migrating to Stack Exchange and are answering questions faster there than on the mailing list. To confirm, we revisited the survey questions about changes in information seeking behaviour and motivations for sharing knowledge. Half of the survey participants reporting changes related to the mailing lists explicitly indicated their transition to Stack Exchange websites, *e.g.*, due to an easier user interface, a friendlier community, or the sites being better indexed by the search engines. In addition, although not explicitly commenting on the speed with which they provide answers, respondents who contribute their knowledge to Stack Exchange acknowledged gamification (*i.e.*, reputation building) as important, *e.g.*, "[I am motivated to answer questions on Stack Exchange because] *it's a game, which also serves a good purpose*" (academic, package maintainer, contributes to documentation, submits bugs, 6 years of experience). Not everyone is attracted by the Stack Exchange incentives, though. R users might still prefer to offer support via the mailing lists, *e.g.*, "*mailing lists are very nice to read and to reply to, due to their text-only policy*" (academic, package maintainer, 11 years of experience).

5.4 Implications

Implications of our work for *Q&A site designers* are twofold. First of all, we have observed that the movement to social, gamified Q&A is correlated with an increase in the engagement of knowledge providers, and the rapidity of response. This finding suggests that *Q&A site designers* should consider gamification elements to increase engagement of the participants, and, indirectly, popularity of their sites. Second, we have seen that users and developers exhibit different behaviour (*e.g.*, developers are more likely to be active on Stack Exchange). This means that the Q&A site designers should cater for different groups of community members (*e.g.*, developers and users) having different needs and expectations, by providing different knowledge sharing channels involving different participation and reward mechanisms. This is also why we do not believe that `r-help` will eventually die off, as Stack Overflow and `r-help` users have explained: "*If you have a problem and you are* completely stuck*, ask a question on the mailing list*"[31] and "*Although many people there gave very detailed answers, I have the feeling that there is much more wisdom on the subject that is still only available in this mailing list*".[32]

For *knowledge communities* our findings suggest that a move to gamified social Q&A can be a beneficial strategy when searching for a better visibility and contributors' activity. We believe that gamification may be of particular interest in closed environments, such as commercial corporations, where knowledge is proprietary, and the set of potentially knowledge providers is closed. Furthermore, the aforementioned distinction between different groups of community members, and different knowledge sharing channels required, suggests that knowledge communities might prefer to combine different channels.

Finally, our findings provide new insights for *researchers* of collaborative software engineering. While there is a significant body of work on why gamification features should positively influence participation in knowledge communities, facilitate user contributions and foster productivity, our study adds concrete evidence that gamification features or community design affect productivity of the contributors.

5.5 Threats to validity

Despite our detailed efforts in experimental design, data gathering and data analysis, we do note several threats to the validity of our methodology and conclusions. The core step in the data gathering, extracting information from the Stack Exchange data dump and the mail archives, is subject to threats to validity akin to those identified for digital trace data [109, 134]. Following the classification of Howison [134], *system and practice issues* in our work may be related to communication through other mailing lists than `r-help` and other Stack Exchange sites than Stack Overflow and Cross Validated. Indeed, advanced R packages such as `lme4` have separate mailing lists,[33] and experts working with these packages might prefer not to migrate to Stack Exchange. However, `r-help`, Stack Overflow and Cross Validated are the biggest platforms of their kind. We also knowingly omitted other venues where knowledge exchange happens, *e.g.*, Google+ groups or individual blogs. Our results are very strong, and give us confidence that Stack Exchange is a good representative of the missing Q&A communities.

[31] `http://stackoverflow.com/a/3382477`

[32] `http://tolstoy.newcastle.edu.au/R/e9/help/10/02/5810.html`

[33] `http://goo.gl/ofUT3L`

Moreover, the data extracted is subject to potential *reliability* threats arising from missing messages from mail archives, and questions and answers being deleted from Stack Exchange. Indeed, as indicated by one of the survey participants, Stack Exchange sites encourage the participants "*to ask well-formed questions, leading to well-formed answers*", while ill-formed questions are being removed. Question or answer removal might lead to underestimating the activity of the Stack Exchange contributors, as well as the overlap between Stack Exchange and `r-help`. Similarly, removal of Stack Exchange accounts might lead to underestimation of the overlap. Another reliability threat is related to multiple system representations of a single individual, which again might incur an underestimate of the overlap between Stack Exchange and `r-help`. We have explicitly addressed this threat when discussing identity merging in the methodology section: while no data gathering is perfect, our efforts are particularly detailed, and have been shown elsewhere to work satisfactorily. Finally, reliability might be threatened by noise in the data, such as spam messages stored in the mail archives.

The next group of threats to validity refers to *representation of the data extracted in the model* of one question followed by multiple answers. While the distinction and relation between questions and answers on Stack Exchange is explicit in the data organisation, the natural counterpart in the mailing lists is the discussion thread (see "Comparing apples and oranges" in the methodology section). Recognition of the discussion thread starter as the asker, and other thread participants as answerers can, however, be threatened by the thread starter posting an announcement rather than a question, as well as by thread participants trying to clarify the intention of the thread starter rather than answering her questions. We have performed an informal evaluation of the discussion threads in `r-help` and observed that the lion's share of threads adhere to the "question and answers" model similar to Stack Exchange.

Temporal aggregation threats arise when aggregating events that occur at different points in time (cf. Figures 5.1 and 5.4). Indeed, inappropriate choice of the time granularity might have made our work subject to ecological fallacy [233]. However, our choice for month as the basic time unit stems from the way traditional mail archives are presented (per month) as well as the literature.

5.6 Conclusions

Using an *aligned, longitudinal* data set which connects the same people over time in different knowledge-transfer venues, we have been able to examine the phenomena associated with the transition to Stack Exchange from the mailing lists.

Future research directions would involve designing and setting up follow-up interviews with R developers and users to address the causality of user migration between different mailing lists and Stack Exchange: which incentives such as more modern user interface, potentially wider audiences, and gamification are perceived as most important for different subgroups within the R community? Understanding causality could result in quantitative predictive models of user behaviour in the presence of various incentives.

Chapter 6

Gender issues

Online communities are flourishing as social meeting web-spaces for users and peer community members. Different online communities require different levels of competence for participants to join, and scattered evidence suggests that female and minorities as participants can be under-represented. Additional anecdotal evidence suggests that women withdraw from unfriendly online communities.

Due to the limited amount of empirical evidence on the matter, this chapter provides a quantitative study of the phenomenon, in order to assess the representation and social impact of gender in online communities. This study positions itself within recent and focused international initiatives, launched by the European Commission in order to encourage women in the field of sciences and technology.

*Focusing on technical support networks around web content management tools (*e.g.*, Drupal and WordPress) and on question & answer websites (*e.g.*, Stack Overflow*), this chapter unearths a spectrum of online communities, in which women participate to various degrees.*

6.1 Introduction

Online communities and social sites represent an extension of the very well known "open source" phenomenon: individuals use their own free time to gather around a common web space, to discuss, socialize or request support from other contributors. Current online communities target a wide spectrum of diverse users, ranging from the general audience (*e.g.*, Facebook), professionals (*e.g.*, LinkedIn), or IT experts specifically (*e.g.*, Stack Overflow). The purposes of these sites can also be very diverse: some provide the ability to share content such as source code fragments (*e.g.*, snipplr), entire projects (*e.g.*, GitHub, bitbucket) or images (*e.g.*, Flickr). Others support knowledge sharing by means of questions and answers (*e.g.*, Stack Overflow) or news postings (*e.g.*, reddit).

Software developers (*e.g.*, professionals working in a company, or voluntaries active in open-source) create and maintain software by standing on the shoulders of others [285]:

they reuse components and libraries, and go *foraging* on the Web for information that will help them in their tasks [36]. When foraging online, developers often turn to mailing lists, or programming question and answer (Q&A) communities such as Stack Overflow[1] (SO). Stack Overflow is becoming one of the most visible venues for sharing knowledge about software development [191]. Participation in such communities is a two-way process: by *asking* (*e.g.*, sending emails to a mailing list, or posing questions on a Q&A platform), developers can seek help and advice from their peers about undocumented technology features [223], report bugs, or discuss future changes to the software; by *answering* (*e.g.*, Q&A questions posed by others), developers can share their knowledge and expertise, and educate their peers. Therefore, such dedicated online communities are an integral part of the working lives of software developers, exponents of Science, Technology, Engineering, and Mathematics (STEM) professionals.

The gender representation in STEM-related subjects raises significant attention of researchers and academics [52, 89, 128, 202, 241], as well as of policy-makers [136], all noting a significant under-representation of women. The main reasons behind such under-representation have been studied mostly qualitatively, to delineate the issue, formulate the main reasons of the imbalance between the number of female and male participants, and propose initiatives to attract more women to sciences, in terms of both majoring in STEM-related studies as well as choosing STEM-related career paths. In particular, encouraging more women to participate in Information Technology, Computer Science, and Computer Engineering is seen as having potential benefits not only to women, but also to society [339, p.235].

In addition to gender and STEM-related studies and careers in general, there is also the issue of representation of women in the use of technology and online communities. The use of Internet technologies is not as unbalanced as the access to careers and vocational studies [26]. Still, software development remains a predominantly male activity, especially for Open Source: all surveys reviewed by David and Shapiro [70] agree that only 1-5% of the open source developers are women. This is in sharp contrast with the 28% female employees with computer and mathematical occupations reported by the National Science Foundation [212].

The focus on gender under-representation in online communities is further motivated by anecdotal observations: it has been suggested that the Q&A website Stack Overflow strongly promotes one-upmanship; fosters flame-wars and the down-voting of individuals; and it is based on earning prizes, reputation and badges, that allow participants to access new features and gain more control on others' postings [306, 307]. Experience suggests that this results in a lesser participation by female users, who do not engage with this sort of communities. In a similar vein Turkle [304, p.194] observed that the focus of hackers in winning and achieving recognition "*makes their world peculiarly male in spirit, peculiarly unfriendly to women*". Moreover, some female Stack Overflow users prefer to use gender-neutral names to be accepted by the mostly-male audiences: similar feeling of a strong pressure to make themselves invisible as women has been reported by Nafus *et al.* [208]. Male Stack Overflow users also masquerade as females sometimes, with the belief that other (male) users would be less aggressive towards them and their questions. Similar "gender swapping" has been observed (and for similar purposes) in an online poker community [343].

[1] http://stackoverflow.com

This chapter is an attempt to quantitatively evaluate the presence of women in specific software-development-related online communities, and to compare their levels and duration of engagement (as compared to the male counterparts). The main rationale for this study stems from the scarcity of empirical studies so far on how gender plays a role in highly-skilled software-development-related online communities, while most of the evidence remains at the anecdotal level. This chapter analyses a sample of female and male users from the Q&A website Stack Overflow, and compares its gender engagement with the interaction on the Drupal and WordPress mailing lists, whose topics (web Content Management Systems) are associated with skills subject to gendered evolution [160].

This chapter is organised as follows: Section 6.2 reports on relevant insights from the existing literature on gender and technology as well as on publications related to the kind of data we analyse and analysis techniques we apply. Section 6.3 deals with the research design of this study following the *Goal-Question-Metric* approach [21]. Section 6.4 discusses gender and self-representation in Stack Overflow, Drupal and WordPress; Section 6.5 summarises the issues with collecting and analysing the necessary data, and includes our gender resolution approach; Section 6.6 presents the results of a pilot survey conducted to obtain insights in the demographics of Stack Overflow, and to test the accuracy of the gender resolution algorithm; Section 6.7 shows and compares the results of the experiments on the analysed communities; Section 6.9 identifies the threats to validity. Finally, Section 6.10 summarises the paper and concludes that two types of communities emerged: while women are still a minority in both, in the first type of community women and men show broadly similar contribution patterns; while in the second type of community, women's levels of participation are significantly lower than men's.

6.2 Related work

This chapter is an extended and revised version of our previous conference paper [315] that focussed on Stack Overflow (SO). There we have observed that: (i) gender of many SO users could not be identified; (ii) the percentage of users engaged in SO and presenting themselves as feminine was greatly imbalanced in comparison to the percentage of users presenting themselves as masculine; (iii) not only do male-perceived users represent the vast majority of contributors, but they also participate more, earn more reputation points, and engage in the "game" imposed by SO more than female-perceived users do. However, the question remained open on *whether a lesser participation of the latter users in SO can be attributed to specifics of SO (such as prizes, reputation and badges), or it is a common phenomenon in online developer communities.* Therefore, in the current work we compare SO with two additional online communities, Drupal and Wordpress.

As the question we address is at the intersection of technology studies, studies of online communities and gender studies, in Subsection 6.2.1 we discuss relevant insights from the literature on gender and technology (technology in general, and gender and information technology/online communities specifically), while in Subsection 6.2.2 we focus on publications related to the kind of data we analyse and gender resolution technique we apply.

6.2.1 Gender and technology

Our work builds upon the existing studies of the relation between gender and technology in general [176] and specifically, gender and information technology [3].

6.2.1.1 Feminist scholarship

Several qualitative studies have focused on the reasons why women are not willing to embark in STEM-related subjects and careers [89, 128, 241]. Various reasons have been given to the under-representation of women in STEM subjects: a general lack of interest in STEM subjects [128], stereotyped thinking by family and teachers [114], lack of role models [194], and most often a combination of various causes together [297]. These approaches frequently adhere to the liberal feminism vision [182]: technology is inherently gender-neutral but controlled by men and misrepresented in the classroom [147, 148]. The gap between women and technology should, hence, be addressed by adapting aspirations, values and behaviour of women [330]. Alternative approaches to conceptualization of the link between gender and technology have been provided by ecofeminist thinking and socialist feminism [117, 132]. Ecofeminists associate women and nature, on the one hand, and men and mechanisms, on the other hand. They reject the reductionist approach characteristic for modern science, as subjugating both nature and women, in favour of holism [200, pp.290–295]. Socialist feminism, also known as "technology as masculine culture", shares with radical feminism the idea of gendered technology but rejects the essentialism of ecofeminism [117, 330, p.8]. This rejection of essentialism by "technology as masculine culture" is close to the concept of "gender as performance" advocated by [43, p.33]. Technofeminism [330] asserts that gender is integral to the sociotechnical process: technology affords or inhibits the doing of particular gender power relations. This also means that technofeminist approaches can expose concrete technological practices leading to exclusion of women.

We subscribe to the latter stance. The ultimate goal of our research consists, therefore, in understanding communication practices in software engineering leading to exclusion of women. Our previous study [315] has shown that communication in SO leads to lesser participation by women. Understanding whether this observation is SO-specific, however, calls for a comparative study, which is the subject of this chapter. In our chapter, we also follow the liberal-feminist approach: we inherently assume that communities and their activity are gender neutral, and we test whether this assumption is correct or not. The Stack Overflow community, as well as the Drupal and Wordpress ones, are technical in nature, so devoted to proposing solutions to technical problems. Gender should not play a role in such communities: an answer to a question is not "more correct" based on the gender of the contributor who solves the problem.

6.2.1.2 Gender and information technology

Different paths of feminist thinking have been also applied to information technology [304, 358, pp.111–114]. For instance, it has been pointed out that the use of Internet technologies is not as unbalanced as the access to careers and vocational studies [26]. Moreover, the "hacker" culture tends to be male-dominated [292] and unfriendly to women [304, p.194], although the number of female hackers is not easy to evaluate [4] and in some countries, such as Malaysia, women account for about half of all students in higher computer science education [169]. Conversely, a major advocate of the open source phenomenon posits that

the hacker culture does not favor a specific gender, but rather is asexual when dealing with technology-oriented problems [239]. This has been questioned by recent results showing an "active discrimination" towards women [208].

The alienation of women and, in general, the unwillingness observed in groups of minorities to participate in online communities have created the so-called "impostor syndrome" amongst women: in spite of having good knowledge and being professionally well-settled, women believe they are disqualified or are doing a fraud by fooling others [51]. Sullivan [287] recommends "Do not feed the Trolls" that can become discouragement amongst the users or members. Apart from the "Science: It's a Girl Thing" initiative launched in June 2012 [136] by the EU commission, there are also some other gender specific online communities who encourage girls to engage in STEM vocational fields [257] and support women in computing and also provide them with a private space to take advice from other members in the same field[2].

Relating to this second sector of related work, our approach is to produce a quantitative experiment of how gender engages with computing technology, without assuming that the "hacker" culture is more or less biased towards one or the other gender, or that there are relevant differences in the behaviour of male and female users, when dealing with technical content, and the production of user-generated content.

The ultimate application of our research consists in creating gender-neutral communities supporting software development. However, before this application can be realised, the gender implications of current communication practices in software engineering should be understood.

6.2.1.3 Gender and online communities

A number of studies targeted a broader question of differences in the online behaviour of men and women [126, 222]. Specifically, impact of gender on participation in online communities has been studied, *e.g.*, in communities targeting cancer [103] and travel [333]. Women have been more actively involved in cancer communities than men [103], despite the common observation that computer-mediated communication, in general, is a male-dominant technology which privileges men [267]. In the online travel community [333] it has been found that men, holding age and educational level constant, have been community members for a longer period of time.

The ways participants of online communities chose to perform their gender through the choice of names, locations and avatars (discussed in detail in Section 6.4) are related to the issues of self-representation and identity display in online communities [115, 309]. Indeed, while the online communities we study cannot be considered social networking sites as defined in [35], both these communities and social networking sites allow the user to construct a public or a semi-public profile. Flexibility of self-presentation in online communities means that by constructing such a profile the users can practice forms of gender vagueness and impersonation [16, 195] such as "gender swapping", *i.e.*, self presentation as the opposite gender [243, 305]. "Gender swapping" is well-studied in the context of multiplayer real-time virtual worlds (known as MUDs [305]), and has been furthermore observed both in SO and in poker communities [343]. "Gender swapping" can

[2] *e.g.*, the Anita Borg Institute for Women and Technology, `http://anitaborg.org`, The Women's Foundation of Colorado, `https://www.wfco.org/` but also LinuxChix, GrrrTalk, Ada Initiative, among others; cf. [179]

be seen as an attempt to *queer SO*, *i.e.*, cross the prescribed norms, or go in an adverse or opposite direction [178].

Our work on participation of women and men in online software engineering communities is at the intersection of gender studies in information technology and studies of computer-mediated communication. Given the male dominance in both domains, we expect to observe male dominance in online software engineering communities as well. However, it is not clear how big the disparity between men and women would be in these communities, and especially whether this disparity would be also visible at the level of engagement.

6.2.2 Methodology

Our empirical analysis is based on analysing data from mail archives and Stack Overflow, and the core step of the empirical analysis is gender resolution.

6.2.2.1 Studies of mail archives and Stack Overflow

Our analysis is based on the data from mail archives of Open Source projects and Stack Overflow (SO). Mail archives have been extensively studied in software engineering research (*e.g.*, cf. [280]), while Q&A websites, and specifically SO, are gaining more and more interest from the research community: since 2010, more than forty research papers were based on the Stack Overflow data [312], *e.g.*, [191, 298]. However, these studies do not consider gender issues, with the aforementioned analysis of gender in mail archives [166] being the only notable exception.

6.2.2.2 Gender resolution

The core step of our empirical analysis is gender resolution, *i.e.*, inferring the gender (of the participants active in the chosen online communities), otherwise undisclosed, from other pieces of information which are publicly available (such as their names or profile pictures). As opposed to using interviews in the sociological studies of genders [32], researchers in information science and software engineering propose automatic or semi-automatic gender resolution approaches. By using quantitative methods one can consider a much broader group of participants. Name-based gender resolution has been attempted before: Herdağdelen and Baroni [125] and Kuechler *et al.* [166] base their decisions on the data from the U.S. Census.

Alternatively, genders can be recognised based on the style of writing [14]. Style-based gender resolution involves counting so called markers that are more frequently used by writers of a certain gender, *e.g.*, pronouns "I", "you" and "she" are significantly more often used by females, while "of"-phrases ("garden of roses") are more typical for male writers [14]. The authors report accuracy of gender resolution as high as 80% [159]. An obvious advantage of the style-based gender-resolution is its robustness against "gender swapping" or individuals presenting themselves in gender-neutral or gender-ambiguous ways. Unfortunately, style-based gender resolution is challenged by a statistical relationship between the gender of the author and the absence of gendered stylistic language features, as described in the study of blogs by Herring and Paolillo [127]. Moreover, style-based gender-resolution is likely to be affected by the writing style, *i.e.*, the existing style-based gender resolution approaches may not be applicable to SO

questions and answers, as SO questions and answers are neither similar to fiction nor to non-fiction documents (*i.e.*, scientific papers) considered by Argamon *et al.* [14], nor to blogs considered by Herring and Paolillo [127]. Finally, gender-resolution accuracy will be affected by errors made by non-native speakers.

Complementary approaches to gender resolution have been proposed by the image processing community [186]. Ideally, these approaches could have simplified or even replaced the manual avatar analysis. Unfortunately, many avatars cannot be regarded as facial images (symbols, cartoons, body parts). Moreover, application of image processing approaches such as [186] would require a manual preprocessing step involving cropping, resizing and rotation of images. Moreover, in online communication users can guess each other's gender by discussing the kinds of things an off-line person of that gender could be expected to know, *e.g.*, related to period, ring sizes and pantihose [286]. This approach, however, cannot be applied to communication in online software development communities. First of all, it does not transfer well across different cultures [195], hence it is not applicable in a multicultural context, such as the one present in these communities. Second, communication in online developer communities is topic-restricted and does not cover more personal topics as mentioned above.

Based on the discussion above, similarly to Kuechler *et al.* [125, 166], *in this article* we opt for a name-based gender resolution approach, augmented with manual analysis. However, while Herdağdelen and Baroni [125] and Kuechler *et al.* [166] report using name lists for USA only, we employ a much broader search across 30 countries, and use additional heuristics going beyond name frequency (Section 6.5.2).

6.3 Research design

This section presents the research design of this study following the *Goal-Question-Metric* (GQM) approach [21].

6.3.1 Goal

The aim of this work is to produce a comprehensive study on how and when women engage in software-development-related online communities (hereafter referred to as "online communities"), or whether specific communities perform better at attracting and retaining women.

Rationale—What is currently known about this "disengagement" phenomenon is still at the anecdotal level, and qualitative and quantitative studies are needed for three reasons: first, to produce a solid and reproducible understanding of the main cause of this disengagement; second, to evaluate the possible long-term implications of such online behaviour; third, to focus on gender-balanced online communities, in order to understand the reasons (if any) of gender-balanced communities and to feed them back to less welcoming communities.

6.3.2 Questions

This chapter addresses the following research questions:

RQ6.1: **What are the challenges with identifying gender in online communities?**

Rationale — The identification of gender in online activities is complicated by several factors: some communities do not record the gender of their participants; users often choose gender-neutral names, or opposite-sex names to cope with a male-dominated environment; in specific countries, certain names are "unisex", therefore the names–to–gender translation has to be country-specific.

RQ6.2: **What is the rate of participation by women in online communities?**

Rationale — While the sharp decline of women in STEM-related subjects is well known, a quantitative study of how many women participate in online communities has not been achieved yet. Before trying to understand the possible reasons behind a possible under-representation of women, it is necessary to first delineate the issue.

RQ6.3: **What is the level of engagement of women in online communities?**

Rationale — Even in case of extreme imbalance in the representation of gender in online communities, it is important to define whether women and men follow similar patterns of contribution and engagement. Showing that women engage less than men in these communities, but achieve similar levels of contribution, would produce a picture of a (relatively) "healthy" community. On the other hand, a community with an imbalanced representation of gender, where the levels of participation vary substantially with gender, would suggest a gender-specific community.

6.3.3 Metrics

The representation of gender and their levels of engagement are measured using various attributes:

- The *number of women and men* participating in online activities. In the case of mailing lists, the participation event occurs when users post messages to the appropriate list; in the case of SO, when a user proposes a new question, or attempts to answer an existing one[3].

[3]Other events are possible in SO, such as commenting/editing posts. We only analyze participation related to posing/answering questions, considered the core activities for a Q&A website.

Table 6.1: Null hypotheses to be tested, and their relation to the research questions

Null (H_0)	H_1	RQ
$H_{1,0}$: women and men are similarly represented	$H_{1,1}$: the number of women in online communities is under-represented, and it reflects the current trend of women enrolling in STEM-related subjects	RQ6.1, RQ6.2
$H_{2,0}$: women engage for a length of time statistically similar to men's	$H_{2,1}$: men engage for longer	RQ6.2, RQ6.3
$H_{3,0}$: women formulate a number of questions statistically similar to men's	$H_{3,1}$: men formulate more questions	RQ6.2, RQ6.3
$H_{4,0}$: women provide a number of answers statistically similar to men's	$H_{4,1}$: men provide more answers	RQ6.2, RQ6.3
$H_{5,0}$: women and men achieve similar levels of reputation	$H_{5,1}$: men achieve higher reputation levels	RQ6.3

- The *number of questions* posed by an individual to the online community. In the case of mailing lists, these are the messages with "no precedence", i.e. just posted, and not written "in reply to" any other messages. In the case of SO, these are the questions that are posted to the community, and that can be voted upon by members, edited, or even closed and deleted.
- The *number of answers* given by an individual. In the mailing lists, this is the number of messages (per individual) "in reply-to" other messages; in SO, it is the number of answers (that can be voted, down-voted, or edited/deleted) given by a user to pending questions.
- The *length of engagement* in an online community. Regarding the mailing lists, this was counted, per individual, as the number of days between the first and the latest message to the mailing list. For SO participants, the number of days between the first question or answer, and the latest question or answer given by a user.

Based on the questions and metrics formulated above, we posit a number of null hypotheses (reported in Table 6.1), to be tested via statistical testing (the exact machinery used is described in Section 6.5.4). The alternative hypotheses test whether the communities under study are gender-specific and biased towards men, and they are drawn from the collected anecdotal evidence.

6.4 Self representation in Stack Overflow, Drupal and WordPress

In this section we discuss how the Stack Overflow, Drupal and WordPress participants present themselves, in general, and their gender identity in particular.[4] Individuals

[4]Following Harding [120, pp. 52–57] we distinguish between symbolic, structural and individual gender. In this and the forthcoming sections we focus on the individual gender, *i.e.*, what counts as masculine and feminine identity and behaviour.

construct their gender, *e.g.*, as feminine or masculine in various ways, and they might present it, consciously or not, in a fashion that is contrary to one's sex. Moreover, individuals might be less eager to disclose their gender or reject the traditional gender binary.

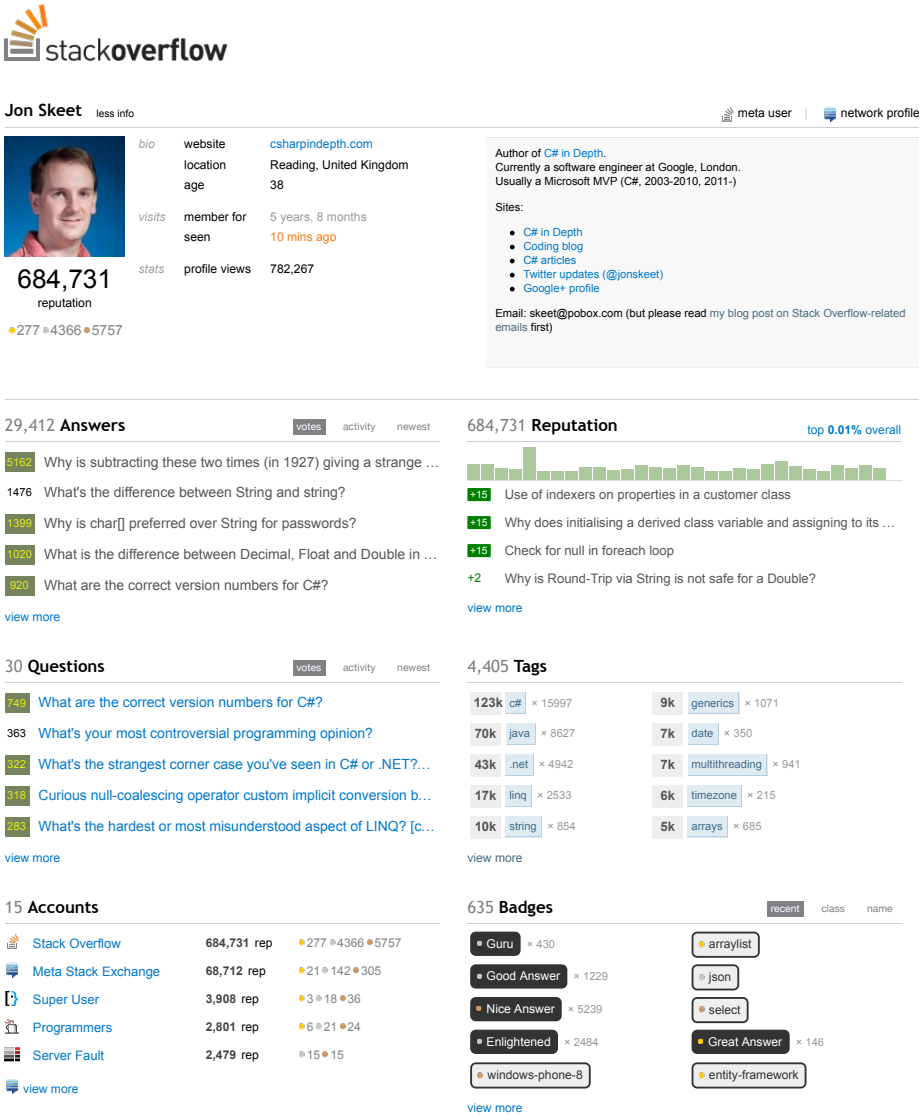


Figure 6.1: A SO user page combines the representation of oneself with the activity-related information.

6.4.1 Stack Overflow

Stack Overflow (SO) is a programming questions and answers (Q&A) website collaboratively built and maintained by programmers, and owned by Stack Exchange, Inc. Created in 2008, SO is the first and largest Q&A website in the Stack Exchange network, and was followed by more than 100 "siblings" on different topics, such as English language, video gaming, photography, or parenting. Moreover, SO has been followed by a number of international clones such as a German-language CodeKicker (started in 2009)[5] and a Russian-language HashCode (started in 2010)[6].

The SO user profile page (Figure 6.1) combines the representation of oneself (self-determined) with activity-related information (automatically provided by SO). Self-representation is expressed through the choice of a user name, the user's real name, personal web-site, email, location and age, as well as a free text self description. A user's gender is not explicitly recorded, but can be constructed here through name, location and avatar choice.

Activity-based information (in the lower part of Figure 6.1) includes the reputation score, badges, questions asked and answers given, as well as tags, *i.e.*, topics, frequent in the user's questions or answers. SO uses *gamification* and an activity-based reputation system: users receive "badges" for different actions performed (*e.g.*, resurrecting and editing posts that were inactive for long periods, up-voting competing answers, or sharing links to questions in order to attract more viewers); similarly, users earn "reputation" points by posting interesting questions and answers (as reflected by the up-votes received from the SO community). The higher the reputation and the more badges one has, the more control she has over SO and other members' postings (*e.g.*, users having earned certain badges can be elected to help moderate the site).

Registration is not obligatory in order to participate in SO: as indicated by a female programmer, referred to as B^{***}, the users might even prefer to select a different name for each question or answer [17]. The SO user page for unregistered users contains the same information as for regular users, with the additional indication *Unregistered* accompanying the user name.

6.4.2 Drupal and WordPress

Drupal[7] and WordPress[8] are free and open source content management platforms used as back-end systems for many websites, ranging from personal blogs to the Eclipse marketplace[9] or the Ubuntu homepage[10]. Similarly to SO, the Drupal and WordPress user pages combine self-representation with activity indication (Figures 6.2 and 6.3). On Drupal gender is recorded explicitly, although this field is not mandatory: should the user be interested in indicating her gender, she can choose between *male*, *female*, *transgender* and *other*. On both Drupal and WordPress gender is shown through the choice of name and avatar. In both the WordPress and Drupal *mailing lists*, produced by the respective communities, full names and emails are available for inspection.

[5] http://codekicker.de/
[6] http://hashcode.ru/
[7] http://drupal.org
[8] http://wordpress.org
[9] http://marketplace.eclipse.org
[10] http://ubuntu.com

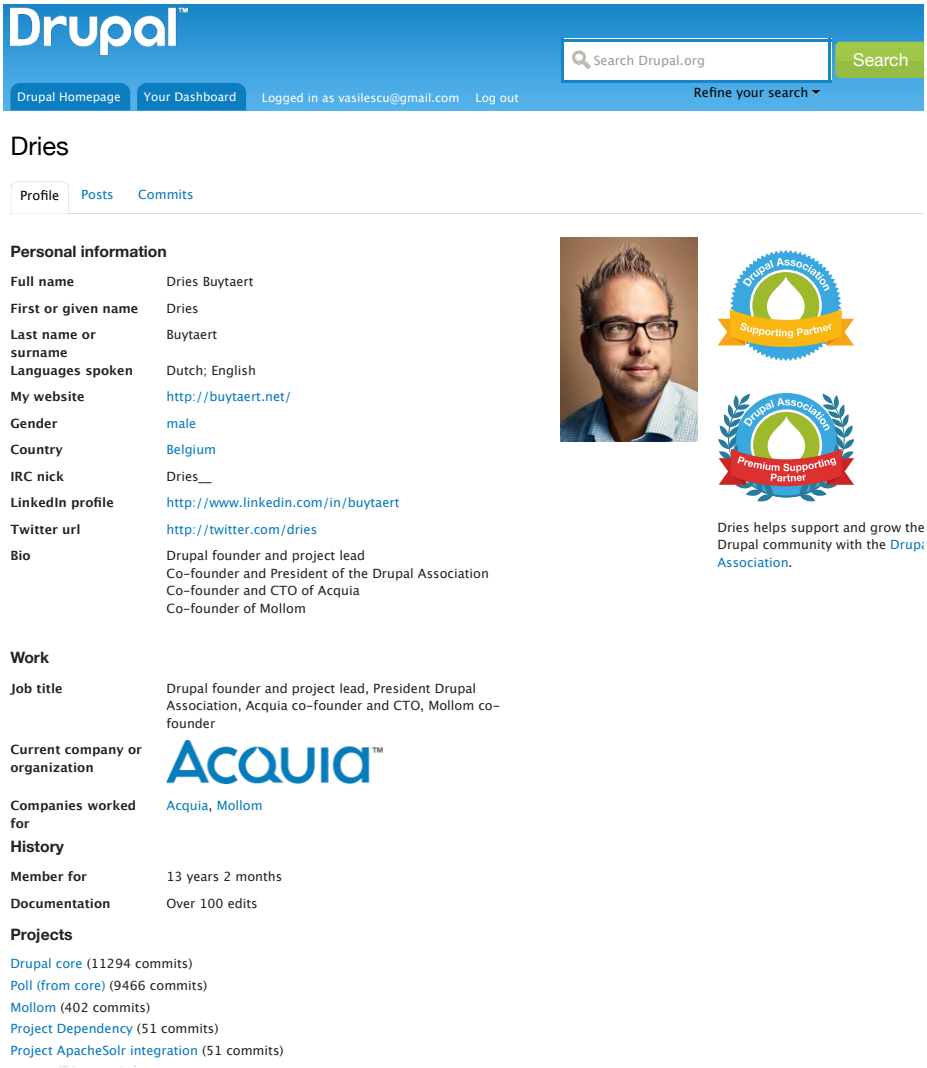


Figure 6.2: A Drupal user page records gender, but the field is optional.

6.5 Empirical approach

As discussed in Section 6.4, gender of participants in online communities is not always recorded. The Stack Overflow and WordPress communities do not record gender at all, while Drupal records gender-related information only for the community members who decide to do so. However, as detailed below, it was found that only a fraction of "registered users" are also "participants" in the Drupal mailing lists. Therefore, one needs to *infer* the gender of participants (as opposed to "registered users") in all three online communities.

The empirical approach followed involves *automatic* and *manual* steps. The *automatic* process comprises inferring gender *based on a person's name* and, if available, their location.

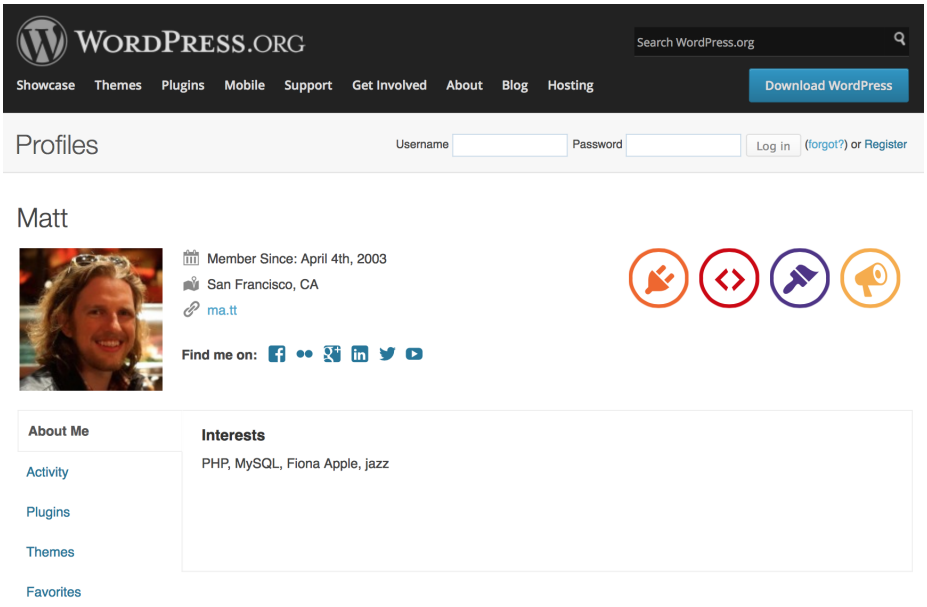


Figure 6.3: WordPress user pages do not reveal gender explicitly.

Despite the cases of "gender swapping" discussed in Section 6.2 and anecdotal evidence that "gender swapping" occurs in Stack Overflow, Armentor-Cota [15] indicated that, *e.g.*, on chat forums there were many more performances of dominant gender identities than there were transgressive acts. Moreover, given the masculine culture of technology [330] and our focus on online communities focusing on technology, we expect male names to predominantly correspond to masculine identities. We can be even more confident in genders reconstructed for Drupal and WordPress mailing lists: Marshall [195] conjectures impersonation to be less frequent when people are identified by e-mail addresses rather than usernames.

The *manual* process is based on inferring gender *based on a person's avatar picture* (only available for Stack Overflow participants), or *based on additional data sources*, such as GitHub, Twitter, Flickr, Facebook, GMail or LinkedIn. The manual process was performed only for those participants for which the automatic process did not result in a gender inference. All results were manually reviewed. The details about data extraction, as well as specific challenges pertaining to the datasets are described below. The accuracy of the gender resolution process is discussed in Section 6.6.

6.5.1 Obtaining the data

6.5.1.1 Stack Overflow

All public data in SO (including the list of members and data about their activity) can be downloaded as part of the Stack Exchange data dump[11]. In this article we explore the data dump dated April 2012, containing information about 1,078,708 registered users.

[11] From `http://www.clearbits.net/torrents/2017-apr-2012`

Table 6.2: Characteristics of the studied mailing lists. The value in the "overall" row is not the sum of the preceding rows, since the same person may have participated in multiple lists.

Drupal	first	last	messages	participants
consulting	12/2005	05/2012	5,790	746
development	04/2005	05/2012	38,702	1,541
support	04/2005	05/2012	22,308	1,627
themes	12/2005	05/2012	1,279	309
translations	12/2005	05/2012	920	154
overall				3,342
WordPress	first	last	messages	participants
wp-accessibility	08/2009	08/2011	29	18
wp-docs	02/2005	05/2012	3,133	281
wp-edu	11/2008	05/2012	578	127
wp-hackers	02/2005	05/2012	43,744	2,128
wp-testers	03/2005	05/2012	14,504	1,451
wp-ui	01/2010	05/2012	279	105
wp-xmlrpc	06/2007	05/2012	633	80
overall				3,611

The SO data set starts in August 2008, which corresponds to the creation date of the website. Since our gender-resolution process is only partly automatic, we decided to sample a smaller set: to obtain a 2% margin error and 99% confidence, a random sample of 4,144 SO users was extracted. Hereafter, whenever we discuss SO results, they have been obtained on this sample.

6.5.1.2 Drupal and WordPress Mailing Lists

The mailing lists analysed for this study are the public lists appearing in the Drupal [78] and WordPress [344] websites. The data from the mailing lists and their participants was analysed using the *mlstats* tool [247]. Data on messages, dates, senders and receivers are recorded and stored automatically. Table 6.2 presents basic facts of the studied mailing lists. Note that the mailing lists are generally older than Stack Overflow (which started in 2008). Note also that the owners of the Drupal and WordPress mailing lists further partitioned these lists into specific sublists, to keep topics clearly separated. For example, WordPress separates discussions between members of the community who want to contribute to the documentation (hosted in the "wp-docs" list), from discussions between people interested in extending WordPress through plugins or improvements to the core code (hosted in the "wp-hackers" list). Such separation is often also indicative of the differences in expected vocabulary (or skills) of the participants: for example, "wp-docs" states that "no coding skills [are] required, just a lot of patience", while "wp-hackers" assumes "a certain level of working knowledge of WordPress and PHP" from its participants [344].

Two additional public sublists for WordPress have been excluded from the analysis, since the vast majority of their traffic is automatically generated: "wp-trac" mirrors

changes to the issue tracking system of WordPress, while "wp-svn" mirrors updates to the WordPress SVN repository. While in both cases these changes have likely been originally performed by humans, their mirrored versions, automatically generated, lose this authorship information. No such lists exist for Drupal.

Since the two communities are relatively small (3,342 participants for Drupal, and 3,611 for WordPress, respectively), no sampling was required to perform the analysis.

6.5.2 Automatic gender resolution

A person's name is often indicative of their gender. For example, John is a common male English first name, while Claire is a common female English first name. Corroborated with location information, even more accurate inferences can be made about one's gender based on their name. For example, Andrea is a common male first name in Italy, but a common female one in Germany. To implement this approach, we iteratively built lookup tables with first names for countries where this information was accessible online. Whenever available (*e.g.*, when the data came from national statistics institutes), we also record the name usage frequency per gender[12]. Both our gender inference tool as well as the data on which it is based are available on GitHub.[13]

6.5.2.1 Preprocessing

The goal of the preprocessing step consists in obtaining the *(name, country)* tuples whenever possible. The first activity to perform is preprocessing the names, and purge the accents, special characters and symbols. We also converted the names from Leet [252] to Latin: for example, w35l3y is converted to Wesley.

Next, we tried to identify the real names of those participants who choose not to disclose them, but instead use *nicknames* (*e.g.*, Carrotman, CoffeeCode) or, in case of SO, standard SO-assigned usernames (*e.g.*, user4106). To determine the real names of such SO users we crawled and parsed their personal webpages, if linked from their SO profile pages. Moreover, if one person has multiple SO accounts, information obtained from one of the accounts can be used to infer gender for another one. To identify accounts belonging to the same person, we made use of email hashes[14]: accounts associated with the same hash have the same email address, hence belong to the same person. For example, user357032 shares the same email hash with Thomas Johnson, so gender can be inferred for the former using the latter's name.

Location information is only available for the Stack Overflow users that choose to describe it in their SO profiles, and is not available for mailing list participants. Even for SO, only a fraction of the users in our sample specify location (821 out of 4,144, or 19.8%), and not all user-specified addresses refer to geographic locations (*e.g.*, The Matrix). Whenever available, the locations were fed into the Google Maps geocoding service[15], and the relative country (if available) was recorded.

[12] We have compiled lists for Albania, Australia*, Belgium*, Brazil, Canada*, Czech Republic, Finland, France, Greece, Hungary, India, Iran, Ireland*, Israel, Italy*, Latvia, Norway*, Poland, Romania, Russia, Slovenia*, Somalia, Spain*, Sweden*, The Netherlands, Turkey, UK*, Ukraine, USA*, and Vietnam. The asterisk denotes countries with frequency information.

[13] `https://github.com/tue-mdse/genderComputer`.

[14] The actual email addresses are not publicly available, for privacy reasons; MD5 hashes, however, are.

[15] `https://developers.google.com/maps/documentation/geocoding/`

6.5.2.2 Gender resolution process

In order to resolve the gender of a participant, we developed a Python tool that resolves a name using the lookup tables discussed above, and a number of heuristics. The tool, available on GitHub at `https://github.com/tue-mdse/genderComputer`, takes a *(name, country)* tuple as input, and returns one of "feminine", "masculine", or "x" (*i.e.*, no gender can be inferred). The resolution algorithm starts with the identification of the first and the last name, and continues with gender detection based on gender-specific last name forms (*e.g.*, *-ova* in Russian), country-specific lookup tables, cross-country lookup and diminutive resolution.

For example, given *(Anna Akhmatova, Russia)* the tool infers "feminine" due to a gender-specific last name form. For *(Jean Delville, Belgium)* the tool chooses "masculine" since Jean is much more frequent as a male name in Belgium (90,211 occurrences as a male name, as opposed to only 98 as a female name). In general we require the name to be at least twice more frequently used for men as for women to infer "masculine", and to be at least twice more frequently used for women as for men to infer "feminine". For *(Bogdan Lalić, Croatia)* the tool again chooses "masculine" despite the fact that we do not have data for Croatia: Bogdan is recorded only as male in all lookup tables that include this name. Observe, however, that we cannot (reliably) infer gender for *(Andrea Demirović, Montenegro)*, since we do not have data for Montenegro and different countries list Andrea both as a male and a female name.

If none of the above results in a resolved gender, and the name contains a single name part (*i.e.*, it resembles a username), we assume it is formatted according to common naming conventions for usernames [28] (*e.g.*, johns for John Smith), and restart the process (*e.g.*, with john derived from johns). We evaluate the accuracy of our automatic gender resolution algorithm in Section 6.6.3, by comparing against the results of a small survey conducted among the SO-users, in which the respondents indicated gender.

6.5.3 Manual gender resolution

Considering the SO names, not all participants choose names that the automatic name-based gender resolution algorithm can successfully parse (*e.g.*, because they prefer nicknames such as CoffeeCode). To improve the accuracy of the automatic gender inference process, whenever it does not result in an inferred gender (*i.e.*, one of either masculine or feminine), we manually inspect additional sources of information: avatar pictures (available for some of the SO users), and websites such as GitHub, Twitter, Flickr, or LinkedIn, which may help reveal a person's real name.

SO users have the option of displaying an avatar picture on their profile pages. We have manually inspected the avatar pictures of the SO users in our sample, and tried to infer gender. However, not all users upload pictures of themselves (*e.g.*, some use default geometric patterns, celebrity stock photos, cartoons or symbols). We do not infer gender from geometric patterns or symbols. Moreover, some of the SO users choose to display pictures of their babies or children, and some display photos of animals, places, or artificial symbols. We chose not to infer gender from such avatars. Ultimately we rely on heuristics to infer gender from the gender of the person or character depicted in the photo. For example, we infer "masculine" from a picture of Kenny McCormick, the South Park character, and "feminine" from a picture of Angelina Jolie. Clearly, the manual gender identification is a subjective process that can be put in jeopardy by "gender swapping"

discussed in Subsection 6.2.1.3; this and additional threats to the validity of our results are discussed in Section 6.9.

We have also observed that people often use the same way to identify themselves (*e.g.*, the same avatar picture, or the same nickname) in other online communities where they are participating (*e.g.*,GitHub, Twitter, Flickr, or LinkedIn). However, the level of personal information available for a given person in each of these communities may differ. For example, a person's Twitter account may also display her full name, or a person's avatar picture may also be used when she comments on blog posts, where she signs with her full name. Whenever we could not directly infer gender using the approaches above, we manually investigated the information available from one's participation in other online communities, and tried to infer gender from full names, as discussed above.

As mentioned above, we also evaluated the accuracy of the combination of the automatic and manual gender resolution on the pilot survey data (for which the respondents themselves have indicated gender), described in Section 6.6.3.

6.5.4 Statistical analysis

Testing for significance the hypotheses reported in Table 6.1 can be viewed as a comparison of multiple distributions: for example, given the total number of questions asked on a mailing list, we had to compare three distributions, two generated by the masculine and feminine gender, and one by the unknowns, *i.e.*, those users for which we could not infer gender.

Traditionally, comparison of multiple groups follows a two-step approach: first, a global null hypothesis is tested, and then multiple comparisons are used to test sub-hypotheses pertaining to each pair of groups. The first step is commonly carried out by means of ANOVA or its non-parametric counterpart, the Kruskal-Wallis one-way analysis of variance by ranks [133]. The second step uses the t-test or the rank-based Wilcoxon-Mann-Whitney test [340], with Bonferroni correction [80, 268]. Unfortunately, the global test null hypothesis may be rejected while none of the sub-hypotheses are rejected, or vice versa [95].

Therefore, one-step approaches are preferred: these should produce confidence intervals which always lead to the same test decisions as the multiple comparisons. To this end, we employ the recently-proposed multiple contrast test procedure $\widetilde{\mathbf{T}}$ [158] using the traditional 5% family-wise error rate (equivalent to a level of significance of 95%). $\widetilde{\mathbf{T}}$ is robust against unequal population variances and is applicable to different types of contrasts, including comparisons of all pairs of distributions (using the so-called *Tukey-type contrast* [303]).

However, not all all combinations of mailing list and metric lead to comparisons of three groups (*e.g.*, there are no unknowns in the "wp-accessibility" sublist of WordPress). In such cases we fall back on the non-parametric Wilcoxon-Mann-Whitney test [340] for assessing whether one of two samples of independent observations (*i.e.*, "masculine" or "feminine") tends to have larger values than the other for a given metric (*e.g.*, number of questions asked). We selected the Mann-Whitney test rather than the t-test since we observed that the distribution of values in each sample is highly skewed (*i.e.*, not normal), while the t-test requires the distributions of values to be normal. Indeed, few participants contribute a lot (*e.g.*, in terms of number of messages or answers), and many others participate very little (*e.g.*, sending very few messages). The Wilcoxon-Mann-Whitney test consists of calculating a test statistic U and comparing its distribution with a distribution which is known under the null hypothesis. The result of this comparison is a p-value. If

the p-value is lower than a predefined threshold (we use the traditional threshold of 0.05, equivalent to a level of significance of 95%), then we can reject the null hypothesis and accept the alternative hypothesis.

6.6 Accuracy of gender resolution

To obtain insights in the demographics of SO, and to test the accuracy of the gender resolution algorithm (discussed in Sections 6.5.2 and 6.5.3), we conducted a pilot survey. The survey was not used to evaluate the gender of all the participants of the SO network; rather, it served as a proof of concept of the assumptions used in the semi-automated approach.

6.6.1 Design and participation

We asked the respondents to indicate their SO userid, gender, age, country of birth, country of residence, highest education level obtained and years of professional experience, as well as involvement in open-source and proprietary software development. We obtained 141 responses, including 127 valid ones (*e.g.*, a unique SO userid mapped to an individual). Since the responses were obtained voluntarily, composition of the sample is likely to be affected by a selection bias. However, this data was only used to derive qualitative conclusions.

6.6.2 Results

Our first observation is that the majority of respondents are men: only 12 respondents from 127 have identified themselves as women. Using the 'score' method of Wilson [341] as described by Newcombe [217] and implemented by Lowry[16], with a 95% confidence interval we estimated the proportion of women in SO in the $[5.5\%, 15.8\%]$ interval.

We also noted that the respondents are predominantly involved either exclusively in proprietary software (47 respondents) or both in proprietary and open source software (47), while the number of exclusively open source developers is lower (17)[17]. This means that *a priori* one could have expected the share of women on Stack Overflow to be between 1-5% reported for open source projects [70] and 28% reported for proprietary software [212].

The 95% confidence interval calculated based on the survey data fits within the expected range. Still, it could be the case that women were less inclined to participate in the survey as the information about age, country of birth or country of residence can be considered private, and they might prefer not to disclose it. Hence, we verified this expectation on a larger scale in Section 6.7.

Finally, we observed that a significant group of respondents (25 out of 127) no longer resides in the countries of their birth due to personal, professional or educational reasons.

[16] `http://www.vassarstats.net/prop1.html`

[17] The remaining respondents are either not involved in software development at all or they indicated a more elaborate answer than "yes"/"no".

Table 6.3: Accuracy of the automatic gender resolution approach described in Section 6.5.2 on the pilot survey data. Precision $\simeq 0.93$, Recall $\simeq 0.58$, Accuracy $\simeq 0.55$.

Gender as described by 117 respondents	**Gender as inferred by automatic approach (name-based)**			
	masculine	feminine	unknown	total
masculine	60	3	43	106
feminine	2	5	4	11
total	62	8	47	117

Table 6.4: Accuracy of enhanced gender resolution approach, combining the automatic steps described in Section 6.5.2 with the manual steps described in Section 6.5.3, on the pilot survey data. Precision $\simeq 0.95$, Recall $\simeq 0.88$, Accuracy $\simeq 0.85$.

Gender as described by 117 respondents	**Gender as inferred by the enhanced approach**			
	masculine	feminine	unknown	total
masculine	90	3	13	106
feminine	2	9	0	11
total	92	12	13	117

6.6.3 Accuracy in the sampled users

Since gender is one of the entries the participants in the pilot survey filled in, we can use this data to evaluate the accuracy of the gender resolution approaches. However, 10 out of 127 valid responses (1 woman, 9 men) correspond to users who joined SO after the data dump was released, so we cannot verify whether their "inferred" gender matches their "described" gender. Therefore, we restrict our accuracy analysis to the remaining 117 SO users.

Tables 6.3 and 6.4 display the number of participants with correctly- and incorrectly-inferred gender, the number of persons with unresolved gender, as well as precision, recall, and accuracy for the automatic and combined automatic+manual gender resolution approaches. Recall from the previous section that the manual approaches (based on avatar pictures or information from other online communities) were only used as an enhancement to the automatic name-based approach, whenever it could not infer a gender. We observe that although the automatic approach has high precision (91%), it has only average recall (58%), due to many SO users preferring usernames or nicknames over real names.

Furthermore, 5 users are misclassified (3 users described themselves as male while the automatic approach inferred "feminine", and 2 described themselves as female and the approach inferred "masculine") due to (parts of) their usernames corresponding to either male or female first names in some countries.

On the other hand, the enhanced approach, combining the automatic steps with the analysis of avatar pictures and additional data sources, correctly identified gender of 34 further individuals labeled as "unknown" by the automatic analysis (cf. Table 6.4). In this way, the enhanced gender resolution approach achieves higher recall (88%) with high

precision (94%), although gender for 13 out of 117 users (or 11%) still cannot be inferred. Failure to infer any gender is due to absence of avatar pictures and use of common words as usernames (*e.g.*, Ubiquité, phant0m, or bitmask), which cannot be confidently assumed to belong to the same individual in other online communities.

Given the sample size, and with a 95% confidence interval, we estimate the precision and recall within the $[84\%, 97\%]$ and $[49\%, 67\%]$ ranges (respectively) for the automatic approach; similarly for the enhanced approach, the precision and recall are in the $[89\%, 98\%]$ and $[81\%, 93\%]$ ranges, respectively.

(RQ6.1) Gender of SO participants is typically not recorded. To infer gender one can use both automatic and manual techniques, with different accuracies. Give a person's *name* and, if available, their *location*, gender can be *automatically* inferred with high precision ($\simeq 95\%$). However, the method has relatively low recall ($\simeq 60\%$), largely due to the fact that not all participants in online communities disclose their real names, and even fewer reveal their locations. However, recall can be further improved (up to $\simeq 90\%$) by *manually* inspecting other data sources, such as *avatar pictures*, or profile pages of the same users in *other online communities* (*e.g.*, GitHub, Twitter, Flickr, or LinkedIn). Such sources may provide additional clues as to a person's real name and location, if not directly their gender.

6.7 Results

This section is articulated in two parts: we start with the mailing list results in 6.7.1, then discuss the Stack Overflow results in 6.7.2.

6.7.1 Mailing Lists: Drupal and Wordpress

We explore the representation (*what is the fraction of mailing list participants in each gender?*) and engagement of gender (*how does the length of engagement with the community and the amount of activity vary with gender?*) along two dimensions: (i) granularity level (*i.e.*, considering all the sublists together and analysing the mailing lists at large, or at the level of each sublist); and (ii) type of engagement (*i.e.*, distinguishing between *asking* and *answering*, since not all participants are necessarily involved in both). Moreover, we test the robustness of the analysis against the "unknowns" (those participants for which we could not infer gender).

6.7.1.1 Hypothesis testing – Gender Representation

When analysing the Drupal mailing lists at large (considering all the sublists together), we inferred "masculine" as gender for 2,879 (3,043 in Wordpress) participants, "feminine" for 328 (282) participants, and "unknown" for 135 (286) participants[18]. These numbers show a first result that is shared for both Drupal and WordPress: some 8-10% of the overall participants construct their gender as feminine, while the vast majority of participants

[18]For most of these unknowns, we observed a very poor engagement to the lists, and often a spamming activity to the same lists.

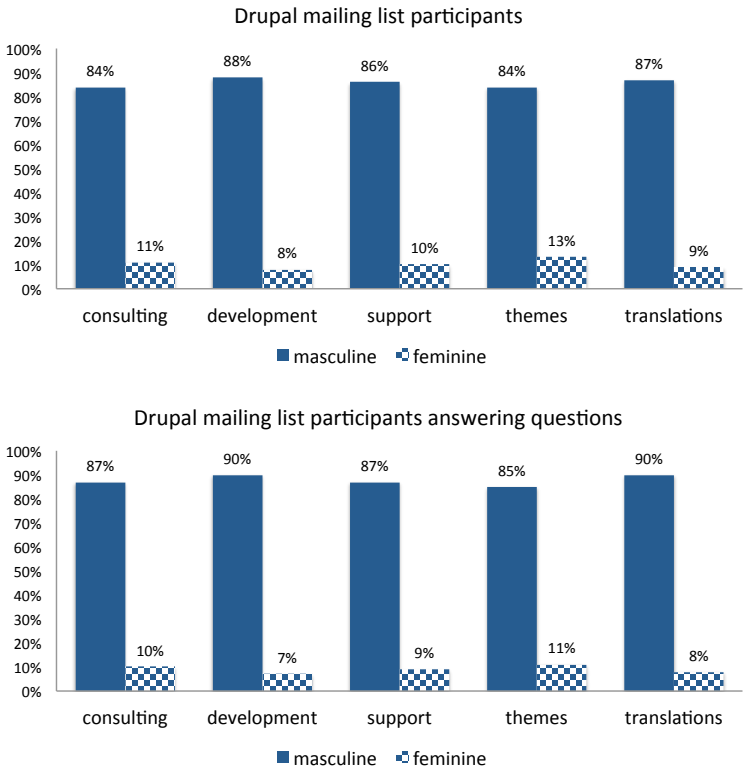


Figure 6.4: Gender of participants in the Drupal mailing lists. Overall (top) and participants answering questions only (bottom). "Unknowns" (not shown) account for the remainder up to 100%.

construct their gender as masculine (some 85%). The fraction of participants labelled "unknown" is approximately 4% in Drupal and 8% in WordPress.

Repeating the analysis at the level of individual sublists (i), on the one hand, and only for the populations of askers and answerers (ii), on the other hand, yields similar results: as an example, Figure 6.4 shows the gender ratios of participants in the individual sub-lists of Drupal (top), and of participants answering questions (bottom) in the same community. Bar chart labels correspond to the names of the Drupal mailing lists (cf. Table 6.2). Similar proportions are found in the number of participants posing questions in Drupal (not shown), and in all the sublists of the Wordpress community (also not shown).

Variation across sublists

The ratio of women participating in these sublists appears quite stable. However, small differences can be observed between different lists or within certain populations (*e.g.*, from Figure 6.4 it seems that there are fewer women in the "development" sublist than on other sublists, both when considering the entire population, as well as only the

population of participants answering questions). To check whether these differences are statistically significant, we applied the $\widetilde{\mathbf{T}}$-procedure. For participants labelled "unknown", we considered three different scenarios:

[A] excluding unknowns from the sample,

[B] assuming all unknowns to be "men", and

[C] assuming all unknowns to be "women".

For the Drupal community, we found that the only significant differences were in the the population of *askers* in the "development" and "consulting" mailing lists. If the individuals labelled "unknown" are excluded (scenario [A]), or all of them are assumed to be men (scenario [B]), the number of women asking questions on the "development" mailing list is lower than on the "consulting" mailing list. In all the other scenarios, and mailing lists, no statistically significant differences exist. In the WordPress data set, statistical significance of the comparisons is more sensitive to the scenario (likely due to a higher proportion of "unknowns" than in Drupal).

However, what is common to both Drupal and WordPress is that the differences in gender participation occur mostly between mailing lists focussing on *designing* technology ("development", "wp-hackers" and "wp-xmlrc") and *using* technology ("consulting", "wp-docs" and "wp-edu"). This observation confirms the anecdotal evidence that "the more technical the role, the less likely a woman will be involved" [190]. Moreover, it concurs with the old adage "boys invent things and girls use things boys invent" [69] which, as observed by Margolis and Fisher [194, p.2], seems to remain "uncomfortably true today".

In conclusion, and in terms of a formal test to ascertain whether gender is under-represented, the null hypothesis $H_{1,0}$ states that "women and men are similarly represented" in online communities. Given the disparity in numbers in both Drupal and WordPress, we had to reject the null hypothesis $H_{1,0}$ regarding the gender representation in the mailing lists, while not being to able to completely reject the alternative hypothesis $H_{1,1}$ with an adequate significance.

Active versus Passive Users

The results on gender representation can be complemented with additional observations, regarding active and passive users of the WordPress and Drupal communities:

- A larger layer of "users" of these communities is composed of visitors, potentially not engaged in the discussions, nor registered as users. The collected data for the US audience provided by the QuantCast company[19] for the two communities[20], and related to the gender of the visitors, shows even less divide between the genders: 44% of the traffic on the Drupal website comes from female users (and, similarly, 39% for WordPress).
- A different ratio is also observed in the number of "interested" users: when screen-scraping the users registered on the `Drupal.org` website (*i.e.*, the "interested" users), and classified as male or female members[21], it is found that some 12,000

[19] `http://www.quantcast.com`

[20] `http://www.quantcast.com/drupal.org#!demo&anchor=panel-GENDER`, `http://www.quantcast.com/wordpress.org#!demo&anchor=panel-GENDER`

[21] `http://drupal.org/profile/profile_gender/male` and `http://drupal.org/profile/profile_gender/female`

users are registered as female, while 63,000 are registered as male, hence reflecting a 15/85 divide between genders (of registered users). Only 43 users indicated their gender as "transgender" and 209 as "other".

- The registered users do not always match the participants in the Drupal mailing lists: many of those users do not participate in the discussions, and various mailing list participants are not registered as Drupal users.

(RQ6.2) The numbers of active participants in the Drupal and WordPress mailing lists (at large, and in the sublists) show that indeed there is a disparity in the gender representation in those two online communities (women account for approximately 10% of the participants). Such disparity is even more acute in the presence of more focused and technical aspects of developing and programming these software systems. On the contrary, this imbalance is not visible in the number of interested (or "passive") users: the male÷female ratio of content users, or readers, shows a much more balanced divide between gender.

6.7.1.2 Hypothesis testing – Gender Engagement

The previous section shows clearly that the gender representation in the Drupal and WordPress communities is highly imbalanced. The interested audience (*i.e.*, the "readers"), the registered users and the mailing list participants get in fact more and more biased towards a male audience, as long as more active involvement is needed. However, these facts alone are not enough to quantify the level of engagement, and the differences between gender in these communities: given for granted that a larger number of masculine participants exist, is it true that their activity and length of engagement with the community are significantly different from the feminine ones?

To answer this question, for each participant and each mailing list we recorded (i) their gender, (ii) the number of questions that they posed to the mailing list audiences, (iii) the number of answers that they gave to any of the questions posed[22], and (iv) the number of days between the first and the last posted messages.

In order to discount for the experience of a user, we normalised the numbers of questions and answers by the number of days in which she was active in the mailing list (likely, a user being around for longer asked and answered more questions than a novice user).

We separated the analysis into two parts: first, we analysed the engagement within the community, and we tested the following null hypothesis:

$H_{2,0}$: women engage for a length of time statistically similar to men's.

This hypothesis corresponds to the observation of Kuechler *et al.* [166] based on six different open source projects: no statistically significant differences were observed between men and women.

Second, we compared activity (number of questions asked, and number of answers given), and we tested the following null hypotheses:

[22]The *mlstats* tool, used for the analysis of the mailing lists, can tag a message as an "answer" if it was sent as a response to an earlier question.

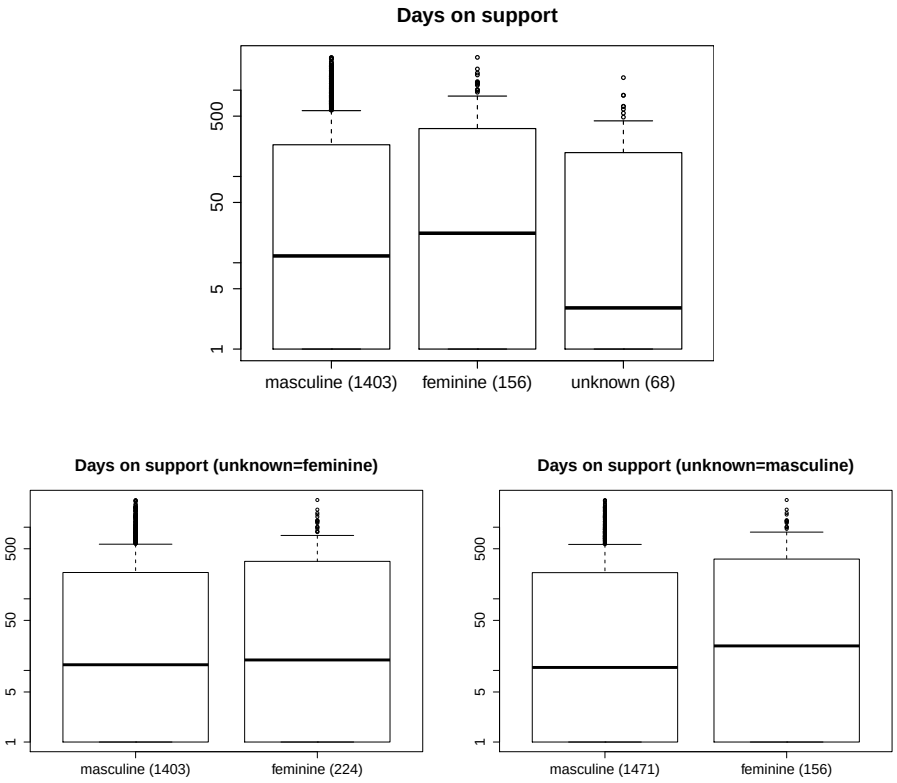


Figure 6.5: Top: length of engagement on the "support" list in Drupal, by gender. Bottom: hypothetical "what if" scenarios where the unknown gender is considered feminine and masculine, respectively. With each gender we display the number of users between parentheses.

$H_{3,0}$: women formulate a number of questions statistically similar to men's.

$H_{4,0}$: women provide a number of answers statistically similar to men's.

Length of engagement with the community

To illustrate the analysis, consider the example of the Drupal "support" sublist depicted in Figure 6.5. The boxplots capture the distributions of the length of engagement on this sublist (measured in days), for each gender and for the unknowns (those users for which our gender inference process did not produce a result). In the middle and right subfigures we depict hypothetical "what if" scenarios, wherein the unknown gender is considered feminine and masculine, respectively, to assess the sensitivity of the analysis to the gender inference process. By applying the $\widetilde{\mathbf{T}}$-procedure to the three groups (masculine, feminine and unknown; left subfigure), we do not find sufficient evidence to reject the null hypothesis

$H_{2,0}$: women engage for a length of time statistically similar to men's.

Table 6.5: Hypothesis testing results for length of engagement with the community. For each sublist we display the number of users tagged masculine (m), feminine (f) and unknown (u). ✓ - hypothesis not rejected at 95% confidence, X - hypothesis rejected

DAYS - DRUPAL MAILING LISTS					
List	m	f	u	$H_{2,0}$	"What if" scenarios
consulting	630	85	31	✓	$H_{2,0}$ is rejected at 95% confidence level in favour of $H_{2,1}$ (men are engaged for longer) if all the unknowns are feminine, or if all the unknowns are masculine.
development	1,362	120	59	✓	$H_{2,0}$ is rejected at 95% confidence level in favour of $H_{2,1}$ (men are engaged for longer) if all the unknowns are feminine.
support	1,403	156	68	✓	no change
themes	259	39	11	✓	no change
translations	134	14	6	✓	$H_{2,0}$ is rejected at 95% confidence level in favour of $H_{2,1}$ (men are engaged for longer) if all the unknowns are feminine.
DAYS - WORDPRESS MAILING LISTS					
List	m	f	u	$H_{2,0}$	"What if" scenarios
wp-accessibility	12	5	1	✓	no change
wp-docs	223	38	20	✓	no change
wp-edu	103	20	4	✓	no change
wp-hackers	1,870	130	128	✓	$H_{2,0}$ is rejected at 95% confidence level in favour of $H_{2,1}$ (men are engaged for longer) if all the unknowns are feminine, or if all the unknowns are masculine.
wp-testers	1,184	116	151	✓	$H_{2,0}$ is rejected at 95% confidence level in favour of $H_{2,1}$ (men are engaged for longer) if all the unknowns are feminine, or if all the unknowns are masculine.
wp-ui	105	11	7	✓	no change
wp-xmlrpc	73	1	6	✓	$H_{2,0}$ is rejected at 95% confidence level in favour of $H_{2,1}$ (men are engaged for longer) if all the unknowns are feminine.

at a 95% confidence level. Then, by applying the Wilcoxon-Mann-Whitney test to the two groups (masculine and feminine; middle and right subfigures) in the "what if" scenarios, we again did not find sufficient evidence to reject the $H_{2,0}$ with a 95% confidence level ($p = 0.89$ and $p = 0.45$, respectively).

Table 6.5 summarises the results of the statistical testing for the other sublists. For each sublist (row), we report the number of masculine, feminine and unknown-gender participants. The ticks show where the relative null hypotheses could not be rejected with a 95% confidence level. The "what if" column describes the three scenarios of treating unknowns. As an example, the line

> **development**; 1,362; 120; 59; ✓; $H_{2,0}$ is rejected at 95% confidence level in favour of $H_{2,1}$ (men are engaged for longer) if all the unknowns are feminine.

from Table 6.5 shows that on the "development" sublist of Drupal we have identified 1,362 users as masculine and 120 as feminine, we could not infer gender for 59 users, and we cannot reject the null hypothesis $H_{2,0}$: "women engage for a length of time statistically similar to men's". However, if all the unknowns were feminine, $H_{2,0}$ would be rejected in favour of $H_{2,1}$: "men engage for longer".

Note that for both Drupal and WordPress, the null hypothesis $H_{2,0}$ cannot be rejected for any of the sublists: there is no significant difference in the length of engagement with the community between feminine and masculine participants.

(RQ6.3) Both Drupal and WordPress have a larger number of masculine participants than feminine participants in their mailing lists. However, this imbalance does not describe the length of gender engagement with the community: both genders engage for lengths of time which are statistically similar.

Amount of activity

To compare the amount of activity on the mailing lists across genders, we repeat the procedure described above. The results of the statistical testing for the number of questions asked, and for the number of answers given on each list are presented in Table 6.6. The values have been normalised to account for the potentially confounding effect of the length of engagement. Note that the number of masculine, feminine and unknown-gender participants in Table 6.6 differs from the corresponding values in Table 6.5: indeed, as mentioned earlier, not all users engage in both asking *and* answering questions.

The null hypotheses ($H_{3,0}$: "women formulate a number of questions statistically similar to men's" and $H_{4,0}$: "women provide a number of answers statistically similar to men's") cannot be rejected for any of the sublists. Moreover, the results are remarkably stable with respect to the unknown-gender.

(RQ6.3) Both Drupal and WordPress come across as "healthy" communities, in which no discrimination is visible in terms of contributed activity. Although women are outnumbered, this imbalance does not describe the level of gender engagement: the contributing activities of men and women are in fact comparable, in number of questions asked, answers given and length of engagement.

6.7.2 Stack Overflow Q&A

Stack Overflow is similar to mailing lists as the core activities are asking and answering questions. Software developers can forage or actively engage in SO or the mailing lists to seek solutions to technical challenges or help others by sharing their knowledge and expertise. However, the Stack Overflow platform leaves room for additional actions besides Q&A (*e.g.*, moderating the website, or editing existing posts); in addition, its gamified environment has the potential to alter the behaviour of its users, who compete to achieve higher reputation and status by being more active.

Table 6.6: Hypothesis testing results for number of questions asked and number of answers given. For each sublist we display the number of users tagged masculine (m), feminine (f) and unknown (u). ✓ - hypothesis not rejected at 95% confidence, X - hypothesis rejected

NUMBER OF QUESTIONS - DRUPAL MAILING LISTS					
List	m	f	u	$H_{3,0}$	"What if" scenarios
consulting	299	51	16	✓	no change
development	884	81	42	✓	no change
support	1,054	129	55	✓	no change
themes	134	25	5	✓	no change
translations	94	10	4	✓	no change
NUMBER OF QUESTIONS - WORDPRESS MAILING LISTS					
List	m	f	u	$H_{3,0}$	"What if" scenarios
wp-accessibility	5	4	0	✓	no change
wp-docs	144	23	14	✓	no change
wp-edu	52	10	2	✓	no change
wp-hackers	1,384	102	98	✓	no change
wp-testers	760	77	101	✓	no change
wp-ui	39	7	5	✓	no change
wp-xmlrpc	56	1	4	✓	no change
NUMBER OF ANSWERS - DRUPAL MAILING LISTS					
List	m	f	u	$H_{4,0}$	"What if" scenarios
consulting	522	58	20	✓	no change
development	1,120	83	38	✓	no change
support	1,042	108	45	✓	$H_{4,0}$ is rejected at 95% confidence level in favour of $H_{4,1}$ (men ask more questions) if all the unknowns are feminine, or if all the unknowns are masculine.
themes	206	27	9	✓	no change
translations	103	9	3	✓	no change
NUMBER OF ANSWERS - WORDPRESS MAILING LISTS					
List	m	f	u	$H_{4,0}$	"What if" scenarios
wp-accessibility	9	4	1	✓	no change
wp-docs	174	27	12	✓	no change
wp-edu	75	15	3	✓	no change
wp-hackers	1,485	86	96	✓	no change
wp-testers	896	82	95	✓	no change
wp-ui	80	8	3	✓	no change
wp-xmlrpc	54	0	3	✓	no change

In particular, the fact that there can be registered users who neither asked nor answered any questions requires a more careful analysis than with the mailing lists, where the "visible" participants (*i.e.*, those encountered by mining the list archives) have either asked or answered at least one question (in other words, have sent at least one email message, otherwise they wouldn't leave any "trace" in the archives).

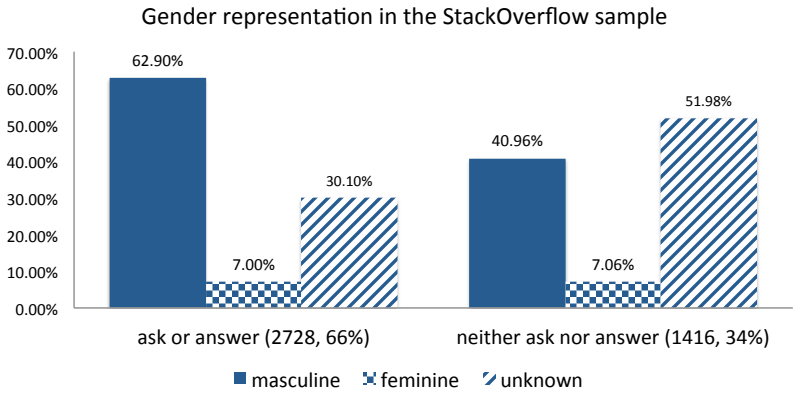


Figure 6.6: Gender representation in Stack Overflow, across two different groups of users: those who ask or answer at least one question (left), versus those who neither asked nor answered any question during the analysed period (right).

6.7.2.1 Hypothesis testing – Gender Representation

Overall, we observed 2,296 users with masculine identities, 291 users with feminine identities, and 1,557 users for which a gender could not be identified, and are not considered as either. The high percentage of unknown-gender users (37.5%) contrasts sharply with the previous situation of the mailing lists, where we could not infer gender for at most 5-8% of the users. For 616 out of these 1,557 (or 40%) it is impossible to infer gender, since they have standard SO-assigned user-names (*e.g.*, user1234), no avatar pictures, and no MD5 email hashes in common with other SO users. This suggests that as opposed to participating in the mailing lists, where not disclosing one's name and email address is more difficult (*e.g.*, it would require using "fake" email accounts created specifically for this purpose), SO users prefer to make use of the possibility offered by the platform to remain anonymous (since filling in personal information in SO profiles is not mandatory).

Another observation is that 34% of the users (or 1,416) in our SO sample have neither asked nor answered any question during the analysed period (Figure 6.6). A closer inspection reveals that only three of these have been engaged in other activities, such as editing posts, or commenting on existing ones. This suggests that virtually all of these inactive users create SO accounts but never use them. Note that SO does not require lurkers to create accounts in order to access existing posts, since all access is public. Moreover, we note that it is more likely for unknown-gender users in the inactive category to be "unresolvable" (*i.e.*, impossible to infer gender for)—51%, than for those in the active category—29%. However, even for the active category, the unknown-gender users generally have a very low reputation on SO, denoting very little activity.

Out of the 2,587 users for which we could infer gender, for 18.9% of them we had to resort to additional sources of information besides their name (in other words, 81.1% of the masculine and feminine users have been resolved automatically using the name-based technique, while the rest have been resolved manually using the enhanced technique): for

example, we relied on avatar pictures for 181 (or 37%) out of these manually resolved users.

The numbers in both groups (active and inactive users) show an overall representation of women at around 7% of the participants[23], and a vast majority of male users, again rejecting the null hypothesis $H_{1,0}$ regarding the balanced gender representation in SO, and accepting the alternative hypothesis $H_{1,1}$. The overrepresentation of males is less visible for the group of inactive users (it seems it is more likely for low-activity users to remain anonymous than for high-activity ones, based on the observation above).

On the other hand, Quantcast reports that the audience of the SO website (*i.e.*, people potentially not engaged in the discussions, nor registered as users) is more gender-balanced than the user community itself: 27% of the SO traffic comes from females[24].

(RQ6.2) There is disparity in the gender representation of Stack Overflow participants in our sample: approximately 55% present themselves as masculine and 7% as feminine, while gender could not be inferred for the remaining 38%. Inactive users are more likely to remain anonymous.

6.7.2.2 Hypothesis testing – Gender Engagement

The analysis of activity of Stack Overflow users follows a similar two-step approach as in the case of the mailing lists. First, we assert whether men and women engage for lengths of time which are statistically similar. Second, we compare their activity in terms of the number of questions asked and the number of answers given. These values are again normalised, to account for the confounding effect of the length of engagement (see the discussion above). Moreover, computations are performed only on the group of active users (those asking or answering at least one question), to enable the comparison with the mailing lists (length of engagement cannot be determined for users that neither asked nor answered any questions).

By applying the $\widetilde{\mathbf{T}}$-procedure to the three groups (masculine, feminine and unknown), we reject the null hypothesis $H_{2,0}$–"women engage for a length of time statistically similar to men's" at 95% confidence level ($p = 5.41 \times 10^{-6}$), thus accepting the alternative hypothesis $H_{2,1}$–"men engage for longer". A "what if" analysis reveals no changes if the unknown-gender users were all masculine or feminine, respectively.

In terms of their asking activity (Figure 6.7 top), the $\widetilde{\mathbf{T}}$-procedure reveals statistically significant differences between the feminine and masculine users, thus rejecting the null hypothesis $H_{3,0}$–"women formulate a number of questions statistically similar to men's" ($p = 1.4 \times 10^{-4}$). However, the alternative hypothesis is different than in the case of the mailing lists: after controlling for length of engagement, women formulate more questions than men. The result does not change when the unknown-gender users are considered all masculine or all feminine, respectively.

In terms of their answering activity (Figure 6.7 bottom), the $\widetilde{\mathbf{T}}$-procedure does not reveal statistically significant differences between the feminine and masculine users: women provide a number of answers statistically similar to men's (thus $H_{4,0}$ cannot be rejected

[23] Using the same methodology as in Section 6.6.2 we estimate the 95% confidence interval as [6.2%, 7.8%].

[24] http://www.quantcast.com/stackoverflow.com

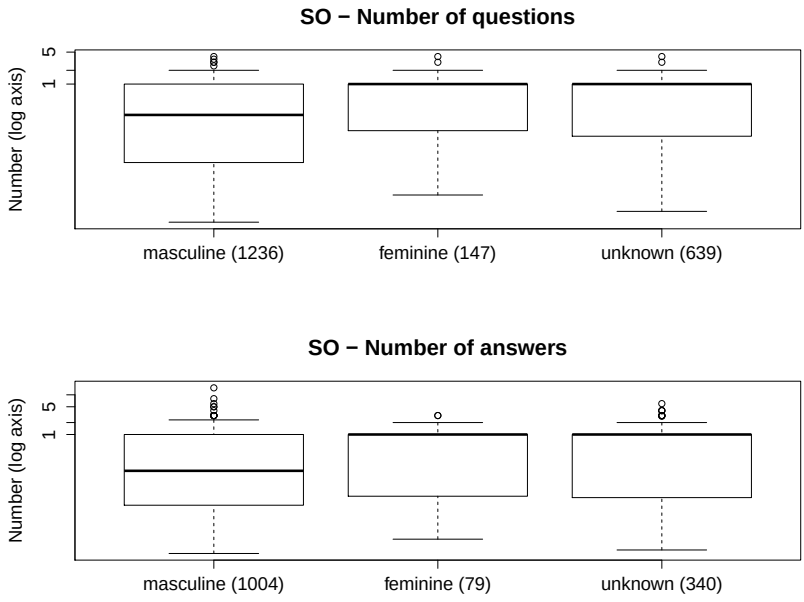


Figure 6.7: Gender engagement in Stack Overflow: number of questions asked (top), number of answers given (bottom). The number of users in each group is displayed in between parentheses. Only users that ask, respectively answer questions have been compared, similar to the mailing lists.

at 95% confidence level). The result would only change if all unknowns were feminine, in which case an alternative hypothesis "women answer more questions than men" cannot be rejected at 95% confidence level.

(RQ6.3) On Stack Overflow it is clear that men engage for longer periods of time than women. Relative to the duration of their engagement in SO, however, women formulate more questions than men, while providing a similar number of answers. This results in a relatively "unhealthy" community where women disengage sooner, although their activity levels are comparable to men's.

6.8 Discussion and implications

This chapter aims at quantifying the gender representation in online communities, by focusing on very different environments. While it is *a priori* clear that men will constitute the majority of participants in the online developer communities, in the same way they constitute the majority of computer professionals [70, 212], it was not *a priori* clear how big the disparity between men and women would be in these communities, and whether this disparity would be also visible at the level of engagement.

While studying the related work, we assumed a gender-neutral approach to the quantitative evaluation of the participation in the studied online communities, therefore we were not expecting any difference of participation and engagement between men and women in Stack Overflow, Drupal or WordPress.

We have observed that Drupal, WordPress and Stack Overflow are close in terms of gender representation: 7-10% of the participants in these communities construct their gender as feminine. This is in line with what has been discussed in the related literature, showing how gender is indeed under-represented in various parts of society: our finding is important to define what proportion of women *actively* engage with the studied communities, while it is still unclear whether a similar proportion could be found in a "passive" use of the online content.

In terms of gender engagement, two different kinds of communities seem to emerge. On the one hand, Drupal and WordPress come across as "healthy" communities, in which no discrimination is visible in terms of contributed activity. Although women are outnumbered, this imbalance does not describe the level of gender engagement: the contributing activities of men and women are in fact comparable, in number of questions asked, answers given and length of engagement. Stack Overflow, on the other hand, appears to be a relatively "unhealthy" community where women disengage sooner, although their activity levels are comparable to men's. From the liberal-feminist approach, discussed in the related work, the Drupal and WordPress communities appear as gender-neutral in terms of engagement, but not on the level of participation. Stack Overflow, on the other hand, seems to be better explained using a socialist-feminist framework, where technology is indeed linked to the masculine culture.

While a formal analysis of the reasons for the differences observed between Drupal and WordPress, on the one hand, and SO, on the other hand, goes beyond the scope of this chapter, we would like to make a couple of remarks on this point. As also noticed by the participants of StackOveflow [306, 307], the community around this Q&A site seems to create and maintain higher barriers to entry for women. It does so by designing an approach to collaboration based on earning prizes, achieving higher status and promoting and fostering extremely fast responses by the participants [191], in turn producing more reputation and status.

As opposed to Stack Overflow, communication on Drupal and WordPress mailing lists focusses on the essence, asking and answering questions, and is not being publicly quantified by means of prizes, statuses or reputation points, and as such might be more attractive for women.

Reflecting on the differences between performance of women and men in a more (Stack Overflow) and less (Drupal, WordPress) competitive environments, we conjecture that *the competitive nature of an online community not only creates higher entry barriers for women, but is also detrimental to their effectiveness.* This conjecture concurs with a series of laboratory experiments on gender differences in competitive environments. Gneezy *et al.* [105] have observed women to be less effective than men in competitive environments, even if they are able to perform similarly in non-competitive environments. Moreover, the observed effect has been stronger when women have to compete against men than in single-sex competitive environments. In a different experiment, Niederle and Vesterlund [218] have observed that women shy away from competition and men embrace it. We believe that quantitative verification of the conjecture above constitutes an important direction for future work.

Going beyond the differences we have observed between different online communities, we stress a number of unsolved challenges pertaining to gender-specific reluctance in equally participating to specific online communities: from encouraging women to enter the field of technology; to their engagement in online communities for expert help and advice; to their sharing of knowledge with other members.

6.9 Threats to validity

The validity of this study is subject to several threats. In the following, threats to *internal validity* (whether confounding factors can influence the findings), *external validity* (whether results can be generalized), and *construct validity* (relationship between theory and observation) are illustrated.

Internal validity – The following threats to internal validity have been individuated:

1. The observation (pilot survey) that a significant number of Stack Overflow users no longer reside in their birth country can affect the internal validity of the name-based gender resolution. Indeed, names associated with one gender in the birth country may be associated with a different gender in the residence country. Since Stack Overflow users indicate the country of residence (if anything) as their location, this means that the gender-resolution heuristics will make a wrong choice: *e.g.*, a male Andrea born in Italy but living in Germany will be identified as female.

2. Furthermore, information on gender has been constructed through name, avatar, and writing style as "masculine", "feminine" or something in between, and may be presented, consciously or not, in a fashion that is contrary to one's sex. This aspect is very relevant when considering the separation between male/female and masculine/feminine identities. We chose to treat names as dimensions of masculine and feminine gender performance.

External validity – The presented results are only valid for very specific communities: we suspect that other online communities act with similar gender barriers (*e.g.*, gaming communities), while others are more gender- and minorities-friendly (*e.g.*, those related to web technologies).

Construct validity – Below we report several points to be considered against the construct validity of this study:

1. Limitations of the automatic approach: as illustrated in the description of the automatic gender resolution tool in subsection 6.5.2.2, such algorithm cannot be expected to map *all* usernames to genders, nor can it be expected to *always* map names to genders correctly.

   Failure to map a username to gender can be related to inherent limitations of the resolution technique. For instance, we cannot resolve *(Norbertas, Lithuania)* since we do not have a name list for Lithuania, and none of the name lists we consider contains the name Norbertas. Furthermore, as explained above, we cannot resolve the gender for *(Andrea Demirović, Montenegro)* since we do not have data for Montenegro and different countries list Andrea both as a male and a female name. Moreover, some individuals might prefer not to disclose personal information and

use standard SO-assigned usernames (*e.g.*, *user1234*). Finally, location information, which is used to improve the accuracy of the technique, is not available for mailing list participants, hence could not be used when inferring gender for these users.

Gender resolution might produce incorrect answers, *e.g.*, if (parts of) the usernames correspond to either male or female first names in some countries. To illustrate this point consider the following artificial example: *(lajos, Belgium)* is recognised as "masculine" since Lajos is known as a male name in Belgium. However, this username could have also been used by a woman called *La*ura *Jos*ten. Moreover, we have chosen to infer "masculine" ("feminine") if the name is known to be at least twice more frequently used for men (women) as for women (men). While predominant popularity of a name can be seen as a strong indication of the gender, it might still produce wrong results for an individual user: some women might have first names more commonly associated with men and some men might have first names more commonly associated with women: *e.g.*, Coffey and McLaughlin [55] report 102 women called "John" (vs. 38,730 men) and 25 men called "Carolyn" (vs. 8,615 women).

2. Limitations of the manual approach: as any manual process, avatar-based gender recognition could have been affected by the evaluator's fatigue or failure to recognise an image, *e.g.*, a cartoon character, as being male or female. To address this threat, we have reviewed the genders inferred and corrected the few instances of misclassified genders.

3. Another threat to validity is formed by Stack Overflow users purposefully using as avatars images of the opposite gender, *e.g.*, male users with erotic stock photos of female models. While we have done our best to identify these cases and resolve them during the review, we cannot guarantee absolute correctness of the manual resolution approach.

4. Furthermore, we note potential human error when inferring gender from an avatar picture, or when deciding whether a certain profile in another data source (*e.g.*, Twitter) belongs to the same Stack Overflow or mailing list participant.

5. Another threat to the construct validity is formed by Stack Overflow users purposely using as avatars images of the opposite gender, *e.g.*, male users with erotic stock photos of female models. These cases have been identified and resolved during the manual review.

6.10 Conclusions

The issue of gender and STEM-related subjects has been studied for several years, and mostly from the point of view of "why" women do not engage with scientific studies or careers. Lesser attention has so far been given to quantify the phenomenon and representation of women in online communities (as technology-"users"), what are their levels of participation, and whether differences can be detected at the gender level. Only anecdotal evidence has been gathered on how specific communities actively encourage (or discourage) women from participating.

This study quantitatively investigated two different types of communities, the one gathered around the Stack Overflow Q&A website, and the mailing lists of two web Content Management Systems (Drupal and WordPress). The number of male and female participants, and the type and length of participation were measured to compare the two communities. The main objective of the study was to add facts to current anecdotal evidence, which suggests that Stack Overflow actively discourages the participation of women, and that web technologies tend to attract more female users.

Special tools were developed to infer gender based on name and nationality; however, since the gender inference was partly manual, the Stack Overflow data was sampled. In the analysis and attribution of gender to participants of online communities, it was found that a large proportion of users (Stack Overflow users in particular) are not identifiable. The large number of untrackable identities goes against the benefits that this community could produce: instead of uniquely identifying themselves, a large proportion of contributors prefer to remain anonymous, which could reflect an unfriendly community, or a way to abstract the solutions to problems from the people providing them.

It was found that the percentage of women engaged in the studied communities is greatly imbalanced, and men represent the vast majority of users and contributors. This finding is in line with the recent down-fall in number of graduates in STEM (and computing in particular) subjects. On the other hand, two types of communities emerged: in the first, the genders are not balanced, but women and men show broadly similar contribution patterns in terms of questions and answers, and length of engagement, when considering "traditional" mailing lists. Differences in participation were only detected in specific WordPress mailing lists, associated with "hacking" and more technical content.

In the second type of community, women are still a minority, but their levels of participation are significantly different from men's: when considering the specific case of Stack Overflow, men engage for longer periods of time than women. However, relative to the duration of their engagement in the community, women are even more active than men when asking questions, while providing a comparable number of answers. This result suggests a relatively "unhealthy" community, in which women disengage sooner, although their activity levels are comparable to men's.

This second set of findings, together with the tendency to remain anonymous, show that the Stack Overflow community in particular, beside producing excellent technical content, sets up higher barriers to entry to its participation, particularly to women. This further exacerbates the division in gender participation and engagement to online communities, which produce quality content at the expense of a generalised participation.

Future work should expand on the current notion of gender as a binary phenomenon (male/female), an approach that has been already criticised by some of the gender-technology students [32, 175]. Indeed, the conflation of gender and heterosexuality has been observed to complicate social relations in male-dominated domains like computing [282], and lesbian women may feel attracted to software development for the same reasons that heterosexual women may feel disinterested in this field [172]. Therefore, as a possible direction for future work we consider going beyond the gender binary and investigating how sexual orientation affects individual involvement in online software developer communities.

As stated in Section 6.8, a crucial direction for further investigation consists in verifying quantitatively whether the competitive nature of online communities creates higher entry barriers for women and is detrimental for their effectiveness (cf. laboratory experiments in [105, 218]).

In addition, based on the observations from different online communities, we would like to design a next generation online platform for software developers, a kind of Stack Overflow ++. Similarly to the study of spreadsheet software [42] the platform should take gender differences into account, without penalising either gender.

Ultimately, the data extrapolated in this study is purely quantitative. However, we acknowledge the value of qualitative research, and we plan to triangulate the reported results by doing qualitative analysis. Specific female-only, online communities (LinuxChix, GrrrTalk, Ada Initiative, among others; cf. [179]), devoted to technical and computer science aspects, have been contacted already for interest in the research and potential future interviews and questionnaires.

Chapter 7

Identity merging

*Understanding an individual's contribution to an ecosystem often necessitates integrating information from multiple repositories corresponding to different projects within the ecosystem or different kinds of repositories (*e.g.*, mail archives and version control systems). However, recognising that different contributions belong to the same contributor is challenging, since developers may use different aliases in different repositories, and even different aliases within the same repository.*

*It is known that existing identity merging algorithms (*i.e.*, algorithms that try to merge aliases belonging to the same contributor) are sensitive to large discrepancies between the aliases used by the same individual: the noisier the data, the worse their performance. To assess the scale of the problem for a large software ecosystem, we study all* GNOME *Git repositories, classify the differences in aliases, and discuss robustness of existing identity merging algorithms with respect to these types of differences.*

We then propose a new identity merging algorithm inspired by Latent Semantic Analysis (LSA), designed to be robust against more types of differences in aliases, and evaluate it empirically by means of cross-validation on GNOME *Git authors. Our results show a clear improvement over existing algorithms in terms of precision and recall on worst-case input data.*

7.1 Introduction

One of the challenges when mining software repositories is identity merging [106]. To study contributors to software projects or software ecosystems, one often tries to integrate information about their contributions in different software repositories, such as version control systems, bug trackers, or mailing lists. However, developers may use different aliases in different software repositories (*e.g.*, Bryan Clark authors Evince changes as *Bryan Clark* with the email address *clarkbw@domainA*[1], but participates in Evince

[1] Domain names obscured for privacy reasons.

mailing lists using *bclark@domainB*), and even different aliases in the same software repository (one of the Empathy developers sometimes uses the nickname *mrhappypants*). Correctly identifying who's who in open source projects is an essential preprocessing step in many empirical analyses: for example, activity of open source developers could be used externally as a measure of their recognition and experience [46].

To integrate information about individual contributions, we therefore need a unique identity representing the same contributor across different repositories and different projects. To this end, we need to use an identity merging algorithm [28, 49, 106, 231, 245]. However, performance of existing approaches degrades sharply in presence of "noisy" data, *i.e.*, data containing large discrepancies between the aliases used by the same individual: "the more noisy and complex the project data is, the worse the merge algorithms behave" [106].

In this chapter we concentrate on aliases used by developers in version control systems (VCS); here the term "alias" refers to a $\langle name, email \rangle$ tuple, typically available in VCS logs. Even for a single repository type such as VCS, the same contributor may use different aliases at different times, or in different projects within the ecosystem. Our goal is to design an identity merge algorithm with improved robustness with respect to noisy data, common in ecosystems maintained by large developer communities. We start by extracting commit authorship information from all GNOME Git repositories, and discuss differences in the aliases used by GNOME developers in Section 7.2. Next, we evaluate robustness of two state of the art identity merging algorithms with respect to types of differences in aliases in Section 7.3. Based on lessons learned from existing approaches, we propose a new identity merging algorithm using Latent Semantic Analysis (LSA) [171] in Section 7.4, and evaluate it empirically by means of cross-validation in Section 7.5. Our results show equally-good performance as the state of the art in the average case, and a clear improvement over existing approaches on noisy input data in terms of precision and recall.

7.2 Types of differences in GNOME aliases

As case study we select GNOME, a popular free and open source desktop environment for GNU/Linux. GNOME has a long development history (some projects, *e.g.*, gnome-disk-utility, have started in 1997 and are still evolving today), is maintained by a large community of developers (we found 8618 different aliases[2] across 1316 different GNOME projects[3]), and is well-known to researchers [98]. Analysis of the Git logs revealed differences in the aliases used by the same author on both dimensions (*name*, *email*).

Overall, 650 out of 7097 different email addresses (9.16%) are associated with more than one name. For example, the highest number of names for the same email address is 164: these were actually combinations of the author's name and the commit message, filled into the author name field of the Git logs. This contributor also used other email addresses, thus his total number of aliases is even higher, 171. Differences in names corresponding to the same email address can be categorized as follows:

- *ordering* (Rajesh Sola, Sola Rajesh),

[2]We consider data from the *author name/email* fields in the Git logs.

[3]Values computed on October 28, 2011, based on the entire lifetime of the projects available at `http://git.gnome.org/browse/`.

- *misspelling/spacing* (Rene Engelhard, Fene Engelhard),
- *diacritics* (Démurget, Demurget),
- *transliteration* (Γιωργος, Georgios),
- *nicknames* (Jacob "Ulysses" Berkman, Jacob Berkman),
- *punctuation* (J. A. M. Carneiro, J A M Carneiro),
- *middle initials* (Daniel M. Mueth, Daniel Mueth),
- *middle names/patronyms* (Alexander Alexandrov Shopov, Alexander Shopov),
- *additional surnames* (Carlos Garnacho Parro, Carlos Garnacho),
- *incomplete names* (A S Alam, Amanpreet Singh Alam),
- *diminutives/variants* (Mike Gratton, Michael Gratton),
- *irrelevant information incorporated in the name* (*e.g.*, the name of the project: Arturo Tena/libole2, Arturo Tena),
- *username instead of name* (mrhappypants, Aaron Brown),
- *artifacts of the tooling used by developers when committing/storing/migrating data* (*e.g.*, timestamps "(16:06) Alex Roberts", or commit messages in addition to names, "Fixed a wrong translation in ja.po. T.Aihana"),
- *mixed* (combinations of the above).

On the other hand, differences in email addresses (assumed to adhere to the *prefix@domain* format) may be due to:

- organisational policies dictating certain prefixes (*e.g.*, *a.serebrenik* and *aserebre*),
- unavailability of a prefix at free mail services (*e.g.*, *ankit644*),
- personal choice (*e.g.*, *kaffeetisch*), or
- (lack of) sensitivity for punctuation (*e.g.*, *john.smith* and *johnsmith*).

Although the same contributor may use different prefixes for different addresses, she is typically consistent in spelling: discrepancies occurred in only 164 out of 7097 cases (2.31%) due to spam protection, *e.g.*, *gerard DOT b AT domain* and *gerard.b@domain*.

7.3 Existing algorithms

We can classify existing identity merge algorithms into two groups: endogenous and exogenous algorithms. *Endogenous* algorithms [28, 49, 106] try to match full names or email addresses shared by different aliases, or use heuristics to "guess" email prefixes based on combinations of name parts (*e.g.*, *jsmith* and *John Smith*). Endogenous algorithms operate under the "closed world" assumption, *i.e.*, they only use the information available in the repositories the aliases come from. In contrast, *exogenous* algorithms [231, 245] also use external information in addition to heuristics to aid in the matching process, *e.g.*, GPG key servers to determine couplings between email addresses [245]. Many open-source projects do not use GPG servers. In this chapter we focus on endogenous algorithms, and discuss the best-performing *simple* algorithm by Goeminne and Mens [106] and a more advanced one proposed by *Bird et al.* [28]. Other algorithms (see [106]) are similar in spirit and, despite more complex heuristics, are still not robust with respect to noisy data.

As part of different algorithms, a *string* can be normalised (denoted $\overline{string}$) by removing accents, converting uppercase into lowercase, replacing multiple whitespace characters by a single space, and removing leading and trailing whitespace.

7.3.1 Simple algorithm

Aliases $\langle name_1, email_1 \rangle$ and $\langle name_2, email_2 \rangle$ are merged by the simple algorithm [106] if $\left\{\overline{name_1}, \overline{prefix_1}\right\}$ and $\left\{\overline{name_2}, \overline{prefix_2}\right\}$ share at least one element, and at least one shared element has length at least a certain threshold *minLen*. For example, if $minLen = 3$, $\langle$*John Smith*, *jsmith@domainA*$\rangle$ would be merged with $\langle$*Jonathan Smith*, *jsmith@domainB*$\rangle$ because both share *jsmith* of length 6.

The approach is robust against noisy aliases as long as the $\left\{\overline{name}, \overline{prefix}\right\}$ sets are not disjoint. However, it is not uncommon for GNOME developers to use disjoint aliases (*e.g.*, $\langle$*William Lachance, wrlach@domainA*$\rangle$, $\langle$*William Rikard Lachance, wlach@domainB*$\rangle$), resulting in false negatives. Moreover, even though two aliases may have the same email prefix (in which case they would be merged), these may belong to different contributors (*e.g.*, when prefixes consist of common first names, $\langle$*John Harper, john@domainA*$\rangle$, $\langle$*John Lightsey, john@domainB*$\rangle$), resulting in false positives.

7.3.2 Bird *et al.*'s algorithm

A more advanced algorithm was proposed by Bird *et al.* [28], who compute approximate rather than perfect matches using the normalised Levenshtein similarity [106] (denoted *sim*). After a normalisation and cleaning preprocessing step, names are split into two parts (*first* and *last*) using whitespace and commas as separators. Then, given a similarity threshold t, two aliases are merged if:

- $sim(\overline{name_1}, \overline{name_2}) \geq t$;
- or $sim(\overline{first_1}, \overline{first_2}) \geq t$ and $sim(\overline{last_1}, \overline{last_2}) \geq t$;
- or $\overline{prefix_{2(1)}}$ contains $\overline{first_{1(2)}}$ and $\overline{last_{1(2)}}$;
- or $\overline{prefix_{2(1)}}$ contains the initial of $\overline{first_{1(2)}}$ and the entire $\overline{last_{1(2)}}$;
- or $\overline{prefix_{2(1)}}$ contains the entire $\overline{first_{1(2)}}$ and the initial of $\overline{last_{1(2)}}$;
- or $sim(\overline{prefix_1}, \overline{prefix_2}) \geq t$.

This approach is more robust to misspelling or punctuation than the simple algorithm (due to the Levenshtein distance). However, it is still sensitive to ordering of name parts (*e.g.*, *Rajesh Sola* and *Sola Rajesh* would probably not meet the similarity threshold since the first and last names are switched), as well as different alphabets (*e.g.*, Cyrillic, Greek) or names with more than two parts, potentially leading to false negatives. Moreover, similarly to the simple algorithm merging aliases with email prefixes consisting of popular first names may result in false positives.

7.4 Latent semantic analysis

Latent Semantic Analysis (LSA) [171], also known as Latent Semantic Indexing (LSI), is a technique used in natural language processing to analyse relationships between a set of documents and the terms they contain. LSA appeared from the need to find relevant documents from search terms: instead of comparing *terms*, LSA enables comparing *meanings* or *concepts* behind the terms to find relevant documents. In software engineering,

LSA has been used, *e.g.*, to identify traceability links between documentation and source code [193], or to manage evolution thereof [143].

LSA uses a sparse *term-document* matrix A which describes the occurrences of terms in documents, although other weighing schemes may also be applied. The typical question one tries to answer with LSA is: *given a term-document matrix A and a query column matrix q, compute the most similar documents to q.* To this end, A is first transformed using singular value decomposition, *i.e.*, $A = USV^T$, where S is a diagonal matrix of the singular values of A. Next, one can compute a rank-k approximation of A by keeping the first (largest) k singular values in S and the corresponding columns of U and V, *i.e.*, $A_k = U_k S_k V_k^T$. Dimensionality reduction is one of the key reasons for applying LSA to identity merging, since it is believed that by reducing the dimensionality of A, much of the "noise" in the input data is eliminated, and the overall retrieval performance of LSA is improved [193] (recall from Section 7.2 that GNOME aliases are noisy). The lower the k (the higher the reduction), the less A_k reflects the original data, but the more noise is removed. Selection of k is therefore an experimental process, results from previous work being generally not transferable. Finally, the similarity between q and a document d is computed as the cosine of the angle between the two vectors, *i.e.*, $sim(q, d) = \frac{q \cdot d}{|q||d|}$. The higher this value, ranging between -1 and 1, the more similar q and d are.

The only identity-merging-specific step in the LSA methodology is computing the *term-document* matrix A. Computing the *documents* starts by grouping together aliases that share a full email address (the underlying assumption is that email addresses are private, *i.e.*, the same email address is not used by different individuals). Next, for each email address a document is created containing the set of normalised name parts of the names associated with that email address (punctuation is first removed, then names are split on whitespace). Normalisation in this case also includes transliteration, removing diacritics, and removing words consisting of only digits. Only words with length at least *minLen* are kept, where *minLen* is our first parameter. Transliteration improves robustness with respect to different alphabets (*e.g.*, we found GNOME author names in Cyrillic, Greek, or Chinese), ignoring words consisting of only digits improves robustness with respect to artifacts of tooling (*e.g.*, timestamps incorporated in names), and ignoring short words improves robustness with respect to initials, prefixes, or suffixes (*e.g.*, *dr.*, *jr.*). For example, the document corresponding to the email address *george.stefanakis@domain* would contain the terms {*george*, *georgios*, *giorgos*, *stefanakis*} as a result of normalising the names *George Stefanakis*, *Georgios Stefanakis*, *Giorgos Stephanakis*, and $\Gamma\iota\omega\rho\gamma o\varsigma$ $\Sigma\tau\epsilon\varphi\alpha\nu\alpha\kappa\eta\varsigma$. Then, to improve robustness with respect to use of usernames instead of names, we add normalised email prefix parts to the document strings (prefixes are split on dots). Since *george* and *stefanakis* are already part of the document string, nothing is added in our example.

Now A contains m columns and n rows, where m is the number of distinct email addresses (*i.e.*, number of documents), and n is the number of distinct terms from all documents. Next, we record occurrences of terms in documents, and fill in ones in the entries a_{ij} if $term_i$ occurs in $document_j$. Clearly, A is sparse. To improve robustness with respect to misspelling, we then also fill into an empty cell a_{ij} the value of the normalised Levenshtein similarity between $term_i$ and $document_j$ if it is at least a certain threshold *levThr* (our second parameter), where the normalised Levenshtein similarity between a term t and a document d is defined as the maximal normalised Levenshtein similarity between t and each $\tau \in d$. For example, if *levThr* is 0.5, then the cell corresponding to

the term *rene* and the document $d = \{fene, engelhard\}$ contains the value 0.75, since the maximal normalised Levenshtein similarity between *rene* and d is 0.75 (due to *rene* and *fene*), and $0.75 \geq 0.5$. Finally, to improve robustness with respect to common first/last names, we weigh each non-empty value a_{ij} by the inverse document frequency of $term_i$, *i.e.*, $idf(term_i) = \log \frac{m}{\sum_{j=1}^{m} a_{ij}}$.

The aliases corresponding to pairs of documents for which the cosine similarity is at least a certain threshold *cosThr* (our third parameter) are recorded as merged. Our fourth parameter is k (dimensionality reduction).

7.5 Empirical evaluation

To evaluate the performance of the LSA identity merging algorithm (and compare it to the algorithms in Section 7.3) we performed cross-validation on GNOME aliases by means of repeated random sub-sampling. As LSA is computationally more complex than the other algorithms we expect it to be slower but to perform better in terms of correctness. We report performance in terms of the f-measure, a popular information retrieval quality metric [254] that summarises precision ($p = \frac{tp}{tp+fp}$) and recall ($r = \frac{tp}{tp+fn}$), *i.e.*, $f = \frac{2pr}{p+r}$, where *tp* is the number of true positives (correct merges), *fp* is the number of false positives (incorrect merges), and *fn* is the number of false negatives (missed merges). The f-measure ranges between 0 and 1 (the higher the value, the better). Detailed results of the evaluation can be found on `http://www.win.tue.nl/~aserebre/ICSM-ERA-2012.html`.

To evaluate the approaches we first constructed a GNOME "oracle" that decides whether two aliases should be merged, for all pairs of aliases. Construction of such an oracle can be only partly automated (*e.g.*, two aliases with a common email address should be merged), and is essentially a manual, labour-intensive, error-prone process. The oracle was computed by one of the authors and manually inspected by two others, and appears free of evident errors. For the 8618 different $\langle name, email \rangle$ GNOME aliases we found, the oracle contains 4989 unique identities, *i.e.*, on average each GNOME contributor uses approximately 1.73 aliases. The highest number of aliases we found for a GNOME contributor is 171 (Section 7.2), due to commit messages being included in the author name fields.

We treat two cases: an *average-case*, containing random samples of the set of 8618 GNOME aliases, and a *worst-case*, consisting of a subset of 673 "noisy" GNOME aliases, expected to cause false negatives in the simple algorithm. We have obtained this data set by removing contributors with only one alias, as well as contributors with intersecting $\{\overline{name}, \overline{prefix}\}$ sets. It is apriori not clear how the algorithm by Bird et al. will behave on the worst-case data set.

For each algorithm/scenario we performed training/testing steps and repeated the process ten times. Training determines optimal parameter values: for the simple algorithm we varied *minLen* $(1, \ldots, 10)$; for the algorithm by Bird et al. we varied the Levenshtein similarity threshold t $(0.05, \ldots, 1)$; for LSA, to avoid training on all combinations of the 4 parameters, we first performed a sensitivity analysis by fixing 3 and varying the remaining. After the sensitivity analysis we restricted the range of *minLen* to $\{2, 3, 4\}$, *levThr* to $\{0.5, 0.75\}$, *cosThr* to $\{0.65, 0.70, 0.75\}$, and k was fixed to half of the number of terms. In the average case, for each of the ten repetitions, training was performed on one tenth of the GNOME aliases ($\simeq 860$), and testing on ten random subsets with the same size from the remaining aliases. Samples were chosen instead of the entire remaining data

for computational efficiency reasons. In the worst case, because of fewer aliases in the data set (673), for each of the ten repetitions, training was performed on one third of the data and testing on the other two thirds. All algorithms as well as the data, can be made available upon request.

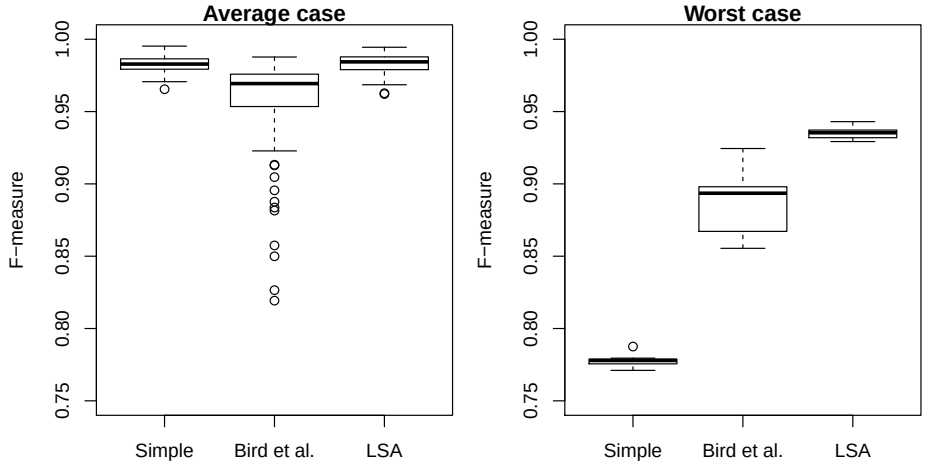


Figure 7.1: The f-measures for the competing approaches. The f-measure ranges between 0 and 1 (the higher the value, the better). LSA performs as well as the simple algorithm in the average case, and significantly better in the worst case. Note that both y-axes start at 0.75.

Figure 7.1 displays the results of the cross-validation. In the average case (left) we observe that LSA performs as well as the simple algorithm (Kruskal-Wallis test followed by pairwise Wilcoxon tests with Bonferroni correction did not reveal enough reasons to assume that the two produce essentially different results at 0.05 significance level), followed by the algorithm of Bird et al. Concurrent results have been obtained in [106]: simple is better than Bird, and is the best of all algorithms tested. LSA and the simple algorithm do, however, behave differently. For example, the simple algorithm does not merge ⟨*Christophe Michael Saout, csaout@domainA*⟩ with ⟨*Christophe Saout, christophe@domainB*⟩ because the two aliases are disjoint, while LSA does. However, the simple algorithm correctly merges ⟨*Gareth Owen, gowen@domainA*⟩ with ⟨*gowen, gowen@domainB*⟩, while LSA does not (the cosine similarity between the documents corresponding to the two is 0.69 and falls just outside the threshold, in this case 0.70). This observation suggests that further improvements of the LSA algorithm, *e.g.*, by using the simple algorithm in a pre-processing step, might be possible, and are considered as future work. On the other hand, the results in the worst case (Figure 7.1 right) show a clear improvement of LSA (median=0.935) over Bird et al's (median=0.893) and the simple algorithms (median=0.778), confirmed by the statistical analysis described above.

7.6 Conclusions

Our main contribution is a generic new identity merging algorithm based on LSA, robust against many types of discrepancies in VCS aliases. Empirical evaluation on GNOME Git repositories has shown equally-good performance of our algorithm as the state of the art in the average case, and better performance in the worst case.

Chapter 8

Diversity among software engineering conferences

In this chapter we study the health of software engineering conferences by means of a suite of metrics created for this purpose. The metrics measure stability of the community, openness to new authors, introversion, representativeness of the PC with respect to the authors' community, availability of PC candidates, and scientific prestige. Using this metrics suite, we assess the health of 11 software engineering conferences over a period of more than 10 years. In general, our findings suggest that software engineering conferences are healthy, but we observe important differences between conferences with a wide scope and those with a more narrow scope. We also find that depending on the chosen health metric, some conferences perform better than others. This knowledge may be used by prospective authors to decide in which conferences to publish, and by conference steering committees or PC chairs to assess their selection process.

8.1 Introduction

In computing science, and especially in software engineering, scientific publications in international conferences (as opposed to journals) are often considered *the* most important way of disseminating research results [48]. The preference for conference publications is motivated by such arguments as: the young age and high dynamism of the field requiring shorter turnaround time between submission and publication than journals typically offer (to avoid results becoming obsolete before their publication) [91]; the increased visibility and publicity associated with presenting a paper and discussing it with peers [91]; the prestige associated with publishing at highly-selective venues with low acceptance rate [201]; the increasing importance given to conference publications by decision makers assessing scientists, both in the USA [225] and in Europe [201].

However, the fundamental role of conferences in computing science is not undisputed [30, 62, 83, 90, 91, 92, 94, 138, 310]. The reported criticism is focused around the limited number of pages, too little time to revise a paper after receiving comments from reviewers, the ultimately higher impact of referenced peer-reviewed journal publications,

and the increased volume of submissions. To keep the review quality high and the reviewer workload low, the latter requires the programme committee (PC) to grow larger. However, "the number of experienced reviewers does not appear to be growing at the same rate" [62], resulting in a "shrinking pool of qualified and willing PC candidates" [30]. This realisation brought a number of conferences to adopt a two-phase review process: first, a broad PC reviews the submissions, then a much smaller Program Board initiates, monitors, and guides the discussions with the PC members. In this way a balance is sought between a reduced review load and high review quality.

Most software engineering conferences follow a single-blind peer reviewing scheme, in which the reviewers know the names of the authors, but not vice versa. This may increase the risk of conferences becoming closed communities, and they may suffer to some extent from introversion. We understand *openness* as the readiness to accept newcomers, either as authors or PC members. Indicative of low openness—closed communities—are, *e.g.*, inviting roughly the same group of people to the PC each year, or preferential acceptance of papers by known authors that have previously published in the same conference. We understand *introversion* as the prevalence of papers (co)authored by PC members among the accepted papers. While theoretically everybody can contribute to any conference, in practice some conferences tend to attract more "new faces" than others. Both problems are well-recognised [30, 62, 290]. Crowcroft et al. [62] argue that "there is a distinct perception that papers authored by researchers with close ties to the PC are preferentially accepted with an implicit or overt tit-for-tat relationship". Similarly, Birman and Schneider [30] question the quality of reviews, but suggest that "work by famous authors is less likely to experience this phenomenon, amplifying a perception of PC unfairness." Therefore, it is useful to study to which extent and for which conferences such symptoms as introversion, closed nature, or shortage of PC candidates occur and, if so, what are the causes and consequences of this occurrence.

In this chapter we assess the health of software engineering conferences with respect to several criteria: community stability (author and PC turnover), openness to new authors, introversion, representativeness of the PC with respect to the authors' community, availability of PC candidates, and scientific prestige. In general, our findings suggest that software engineering conferences are healthy: balanced PC turnover (high enough to avoid introversion, yet low enough to ensure continuity and coherence), high openness to new authors ("new" in terms of both turnover with respect to previous years as well as not having published at that conference ever before), and moderate introversion (in terms of fraction of papers co-authored by PC members). Nonetheless, some conferences perform better than others according to the aforementioned criteria. In addition, we observe important differences between conferences with a wide scope and those with a more narrow scope. This knowledge can be used by conference steering committees and PC chairs, *e.g.*, to assess composition of the PC, paper selection process and adherence to conference charters. Furthermore, prospective authors might consider conference openness as well as prestige when deciding to which conferences to submit their work.

The remainder of this chapter is organised as follows. Section 8.2 describes our research methodology, including the selection of conferences, the metrics proposed to characterise the health factors, and the data extraction process. Section 8.3 details the statistical analysis carried out and its findings. Section 8.4 discusses the results on a per conference basis. Section 8.5 surveys related work. The threats to validity, part of any empirical study, are presented in Section 2.5. Section 8.6 sketches directions for future work and Section 8.7 concludes.

8.2 Methodology

8.2.1 Data extraction

Numerous software engineering conferences are organised every year. Moreover, papers addressing software engineering topics are also solicited by wider-scoped computer science conferences. In our study, we focused on the conferences studied in [290]:

- International Conference on Software Engineering (**ICSE**),
- European Conference on Software Maintenance and Reengineering (**CSMR**),
- International Conference on Program Comprehension (**ICPC**),
- International Conference on Generative Programming and Component Engineering (**GPCE**),
- International Conference on Software Maintenance (**ICSM**), and
- Working Conference on Reverse Engineering (**WCRE**).

Of these 6 conferences, only ICSE has a wide coverage of the software engineering domain, while the others focus on a specific subdomain (maintenance, reverse engineering, program comprehension, and generative programming). To balance our sample in terms of scope, we added three more conferences with wide coverage of software engineering, namely

- International Conference on Automated Software Engineering (**ASE**),
- Symposium on the Foundations of Software Engineering (**FSE**), and
- International Conference on Fundamental Approaches to Software Engineering (**FASE**)

Furthermore, to balance our sample in terms of age, we also included two younger conferences, namely

- Working Conference on Mining Software Repositories (**MSR**), and
- International Working Conference on Source Code Analysis and Manipulation (**SCAM**)

The data we analysed was restricted to the main research track of each conference: number of papers submitted and accepted (without distinguishing between long and short papers, if both were part of the main track), authors of the accepted papers, and composition of the programme committee. In order to facilitate replication of our study, we have published all the data and tooling developed during our work on GitHub at `github.com/tue-mdse/conferenceMetrics`. The data set is described in more detail in [321].

For all considered conferences, most of the data of all accepted papers and their authors was extracted from the DBLP records [1]. The extracted data covers a period of at least ten years, as can be seen in Table 8.1. Data about the composition of the programme committee and number of submitted papers to each conference was retrieved from the websites of each conference and online proceedings volumes. For earlier editions we used the Wayback machine[1] to analyse websites which were no longer available as well as announcements posted by conference organisers in Usenet newsgroups.

Since we are integrating data from different sources, the names of authors and PC members are not necessarily consistent, while it is critical to know the identities of persons

[1] `archive.org/web/web.php`

if we wish to check for signs of introversion. For example, Mark van den Brand is also known as Mark G. J. van den Brand or M. G. J. van den Brand. To match multiple aliases for the same person we performed *identity merging* [28, 106, 163, 231, 245, 319, 320] (Chapter 7), and manually checked the results in a post-processing step.

Conference series	First ed. issued	First ed. included	Last ed. included	Included in [290]	CI	Charter availability
ASE[2]	1986	1994	2013	No	55	Own charter[3], SIGSOFT PC policy[4]
CSMR	1997	1997	2013	Yes	40	Yes, but not public; no guidelines on PC renewal[5]
FASE	1998	1998	2013	No	42	Yes[6]
FSE[7]	1993	1993	2013	No	49	SIGSOFT PC policy
GPCE[8]	2000	2000	2013	Yes	37	Yes; no guidelines on PC renewal
ICPC[9]	1992	1997	2013	Yes	41	Yes[10]
ICSE[11]	1975	1994	2013	Yes	117	SIGSOFT PC policy
ICSM[12]	1983	1994	2013	Yes	53	Yes[13]
MSR	2004	2004	2013	No	32	No[14]
SCAM	2001	2001	2013	No	15	Own charter + part of SIGSOFT PC policy
WCRE	1993	1995	2013	Yes	43	Yes, but not public; no guidelines on PC renewal

Table 8.1: Software engineering conferences considered in the study. Those conference with a wide coverage of the domain are indicated in **boldface**.

8.2.2 Metrics

Table 8.2 shows all metrics we have used. Some of these coincide with those used in [290]. The basic metrics count the number of authors $\#A$ and PC members $\#C$ as well as how many papers have been submitted $\#SP$ to a conference for a particular year. For the $\#C$ metric we also considered PC chairs and General Chairs as PC members.

We quantify the review ***workload*** RL experienced by PC members as the ratio between the number of submitted papers and the size of the PC.

To determine whether a conference community for a particular year is ***stable***, we use two *sliding window* metrics $\#NA$ and $\#NC$, that count the number of New Authors or programme Committee members over several previous years. We also use their relative counterparts RNA and RNC.

[2]Formerly Knowledge-Based Software Engineering Conference (KBSE) and Knowledge-Based Software Assistant (KBSA).

[3]`www.ase-conferences.org/Charter.html`

[4]`www.sigsoft.org/about/policies/pc-policy.htm`

[5]A public charter is in preparation and is expected for 2014.

[6]`www.easst.org/fase/fasech`

[7]We do not distinguish between the years when FSE is colocated with the European Software Engineering Conference (and is known as ESEC/FSE) and the regular editions of FSE.

[8]Formerly Semantics, Applications and Implementation of Program Generation (SAIG).

[9]Formerly Workshop on Program Comprehension (WPC) and International Workshop on Program Comprehension (IWPC).

[10]`www.program-comprehension.org/ICPC-ProgramCommittee-v1.1.pdf`

[11]Formerly National Conference on Software Engineering.

[12]Formerly Conference on Software Maintenance.

[13]`conferences.computer.org/icsm/PC-Guidelines.pdf`

[14]A charter is in preparation and is expected for MSR 2014.

Acronym	Definition
Basic metrics	
$\#A(c, y)$	number of distinct Authors for conference c in year y
$\#C(c, y)$ $[\#PCmem]$	number of PC members for conference c in year y
$\#SP(c, y)$ $[\#subm]$	number of Submitted Papers for conference c in year y
Workload	
$RL(c, y)$ $[revCoeff]$	Review Load for conference c in year y, *i.e.*, $\frac{\#SP(c,y)}{\#C(c,y)}$
Stability	
$\#NA(c, y, n)$	number of New Authors for conference c in year y that were not author in years $y-n$ to $y-1$
$\#NC(c, y, n)$ $[\#(real)newPCmem]$	number of New PC members for conference c in year y that were not PC member in years $y-n$ to $y-1$
$RNA(c, y, n)$	author turnover = Ratio of New Authors for conference c in year y w.r.t. years $y-n$ to $y-1$, *i.e.*, $\frac{\#NA(c,y,n)}{\#A(c,y)}$
$RNC(c, y, n)$ $[(real)newPCprop]$	PC turnover = Ratio of New programme Committee members for conference c in year y w.r.t. years $y-n$ to $y-1$, *i.e.*, $\frac{\#NC(c,y,n)}{\#C(c,y)}$
Openness	
$\#PNA(c, y, n)$	number of Papers of conference c in year y by New Authors for which none of the co-authors has published at this conference in years $y-n$ to $y-1$
$RPNA(c, y, n)$	Ratio of Papers (by New Authors) for conference c in year y for which none of the co-authors has published at this conference in years $y-n$ to $y-1$, *i.e.*, $\frac{\#PNA(c,y,n)}{RA(c,y)*\#SP(c,y)}$
Introversion	
$RAC(c, y, n)$ $[PCaccProp]$	Ratio of accepted papers for conference c in year y co-authored by programme committee members who served at least once during years $y-n$ to y
Representativeness	
$RCnA(c, y, n)$	Ratio of PC members for conference c in year y that have never co-authored a paper at preceding instances of c between $y-n$ and $y-1$
Sustainability	
$SR(c, y, n)$	Sustainability Ratio = ratio between the number of core authors that have not served on the PC in years $y-n$ to y and $\#C(c, y)$.
Prestige	
$RA(c, y)$ $[accRate]$	Ratio of accepted papers for conference c in year y.
$CI(c)$	Conference Impact of conference c = SHINE h-index for c between 2000 and 2012.

Table 8.2: Metrics used to assess conference health. If applicable, the acronym used by Systä et al. [290] is mentioned between square brackets.

To study ***openness*** of a community surrounding a conference, the sliding window metric *#PNA* counts the number of Papers by New Authors, *i.e.*, those papers for which none of the authors published at previous editions of this conference. Similarly, we use its relative counterpart *RPNA*.

By ***introversion*** we understand influence of PC membership on paper co-authorship. Systä, Harsu and Koskimies [290] call this feature "inbreeding" to reflect the negative phenomenon of favouritism of the PC towards papers co-authored by PC members. However, high share of papers co-authored by PC members among the accepted papers might stem from higher quality papers or higher publishing activity of more experienced researchers, who are also more likely to be invited to join the PC. Therefore, we use a more neutral term for this aspect of the conference health. Similarly to [290], we quantify introversion by calculating *RAC*, the ratio of accepted papers co-authored by programme committee members who served at least once in recent years.

We assess the ***representativeness of a PC for the community*** by *RCnA*, the ratio of PC members that have never co-authored a paper in several preceding editions of the same conference. To assess the ***sustainability of the PC-candidates pool*** (its rejuvenation capacity), we measure the sustainability ratio *SR*, the ratio between the number of *core authors* that have not served on the PC at previous editions of this conference (PC candidates) and the size of the PC at that time.

We define a *core author* for a given conference as a person who frequently (co)authored papers published at that conference during the current or previous four editions. Specifically, we consider an author to be core author if either: (i) she has (co)authored papers in at least 3 out of the 5 most recent editions; or (ii) she has (co)authored papers in at least 2 out of the 3 most recent editions. A core author is therefore a very active member of the author community, who probably *deserves* to serve on the PC.

Finally, we use two accepted ***prestige*** measures, conference impact *CI* and acceptance ratio *RA*. We compute *CI* based on the Simple H-INdex Estimator (*SHINE*) [2], a conference-specific variant of Hirsch's *h*-index for quantifying an individual's scientific research output [46, 131]. For a given conference *c*, $CI(c) = SHINE(c, 2000, 2012) = 40$ means that conference *c* has 40 papers published between 2000 and 2012, each with at least 40 citations in the same period. The earliest *SHINE* data available is from 2000. Since computation of the *h*-index can be inaccurate for recent years (due to late propagation of citation information) we use the entire available history of conference citations since 2000 until 2012. In general, *CI* and the conference rankings published by the CORE ERA ranking[15] (Computing Research and Education Association of Australasia) agree: ICSE ($CI = 117$) is the only conference in our list ranked A^*, ASE, FSE and ICSM ($49 \leq CI \leq 59$) got the *A* rank, FASE (42), GPCE (37) and WCRE (43) were ranked *B*, and finally, CSMR (40), ICPC (41), MSR (32) and SCAM (15) were ranked *C*.

8.2.3 Data analysis

Using the R project for statistical computing [238] we visualise and statistically analyse the data to detect patterns and trends, with the aim to detect (counter-)evidence of conference health, *e.g.*, signs of introversion or openness. The visualisation consists of two components, cf. discussion in [140]: (i) a simple graph with all the time series (conferences)

[15] `http://core.edu.au/index.php/categories/conference%20rankings/1`

overlaid, to facilitate comparisons over smaller visual spans, and (ii) small multiples for each of the time series, to facilitate assessing trends visually.

To quantify *monotone* trends we compute Spearman rank correlation ρ between the values of the metrics and the time axis: since the latter is monotonically increasing, strong correlation (either positive or negative) indicates presence of a trend in the metric (either increasing or decreasing, respectively) [60]. Similarly, to verify presence of *linear* trends (between different metrics rather than between a metric and the time axis) we compute Pearson correlation r.

To verify claims such as "conference A tends to have higher values for metric m than conference B", we compare multiple distributions of m (one for each of the 11 conferences). Traditionally, comparison of multiple groups follows a two-step approach: first, a global null hypothesis is tested, and then multiple comparisons are used to test sub-hypotheses pertaining to each pair of groups. The first step is commonly carried out by means of ANOVA or its non-parametric counterpart, the Kruskal-Wallis one-way analysis of variance by ranks [133]. The second step uses the t-test or the rank-based Wilcoxon-Mann-Whitney test [340], with Bonferroni correction [80, 268]. Unfortunately, the global test null hypothesis may be rejected while none of the sub-hypotheses are rejected, or vice versa [95]. Moreover, simulation studies suggest that the Wilcoxon-Mann-Whitney test is not robust to unequal population variances, especially in the unequal sample size case [40, 355]. Therefore, one-step approaches are preferred: these should produce confidence intervals which always lead to the same test decisions as the multiple comparisons. To this end, we employ the recently-proposed multiple contrast test procedure $\widetilde{\mathbf{T}}$ [158] using the traditional 5% family-wise error rate. $\widetilde{\mathbf{T}}$ is robust against unequal population variances. A more comprehensive discussion of $\widetilde{\mathbf{T}}$ goes beyond the scope of this chapter and can be found in [158]. Small examples detailing the application of $\widetilde{\mathbf{T}}$ can be found in [158, 320] (Chapter 2) and additional scientific applications of the technique in [67, 139, 318].

For ease of presentation (given 11 conferences, one has to report the results of $\frac{11\times10}{2} = 55$ comparisons per metric), we use the $\widetilde{\mathbf{T}}$-graphs proposed in [320] (Chapter 2) to summarise the results as a directed acyclic graph. For a particular metric, nodes of the graph correspond to conferences, and edges to results of pairwise comparisons (there is an edge from A to B if A tends to have higher values for that metric than B). Because transitivity is respected by $\widetilde{\mathbf{T}}$ (as opposed to, *e.g.*, the traditional pairwise Wilcoxon-Mann-Whitney tests [39]), we omit direct edges between A and B if there is a path from A to B passing through at least one other node C.

A special case of comparison of multiple distributions is the comparison of two distributions (*e.g.*, all wide-scoped conferences versus all narrow-scoped ones). We need to test whether one of two samples of independent observations tends to have larger values than the other. The Wilcoxon-Mann-Whitney two-sample rank-sum test [13, 151] is not robust against differences in variance [40, 355], and the $\widetilde{\mathbf{T}}$ procedure as described above cannot be used to compare two distributions [158]. We therefore use the two-distributions equivalent of the $\widetilde{\mathbf{T}}$ procedure, *i.e.*, we perform two sample tests for the non-parametric Behrens-Fisher problem [40], and compute confidence intervals for the *relative effect* of the two samples (using the R package `nparcomp` [157]). If the relative effect of samples A and B, which is traditionally denoted $p(A, B)$ [40], exceeds 0.5 then B tends to be larger than A. Therefore, we accompany visualisations of the evolution of different metrics by $\widetilde{\mathbf{T}}$-graphs (for a more rigorous view of the relations between different conferences), and we

report the relative effect p(wide,narrow) (to support claims about the relation between conferences when grouped into wide-scoped and narrow-scoped).

To avoid clutter when reporting p-values, regardless of the statistical procedure applied, and to avoid the possible confusion between the relative effect p and p-values, we superscript the test result using the following convention: no superscript corresponds to p-value ≥ 0.05, * corresponds to $0.01 \leq$ p-value < 0.05, and ** corresponds to p-value < 0.01.

8.3 Results

8.3.1 Workload

We start our analysis of conference health with an overview of their general state: Are software engineering conferences attracting more submissions? Do the programme committees grow? Is there evidence of increased reviewing load?

Figure 8.1 displays the variation of the number of submitted papers per year $\#SP(c, y)$, for each of the conferences. The most popular conference (*i.e.*, the one receiving the most submissions) is ICSE, followed by ASE, FSE and ICSM (Figure 8.3). It is interesting to note that the narrow-scoped ICSM received over the years comparable (in terms of the $\widetilde{\mathbf{T}}$ procedure) numbers of submissions as the wider-scoped FSE, ASE and FASE. Increasing trends are confirmed for MSR ($\rho = 1^{**}$), ICSE ($\rho = 0.89^{**}$) and FASE ($\rho = 0.73^{**}$): these conferences tend to receive more submissions each year, since their first editions. The other conferences exhibit less clear (increasing or decreasing) trends. Overall, wide-scoped conferences (solid lines) tend to receive more submissions than narrow-scoped ones (dashed lines): p(narrow,wide) $= 0.911^{**}$.

With the exception of ICSE, conferences receiving increasingly more submissions resort to increasing the size of their programme committees. Inspection of Figure 8.2 reveals increasing trends for $\#C$ for MSR ($\rho = 0.99^{**}$), FASE ($\rho = 0.87^{**}$), CSMR ($\rho = 0.79^{**}$), ICSM ($\rho = 0.77^{**}$), ICPC ($\rho = 0.76^{**}$) and GPCE ($\rho = 0.75^{**}$): these tend to increase their PC size over the years. We notice that this includes 5 out of 7 of the narrow-scoped conferences. The other conferences exhibit less clear trends. Out of all the conferences in our sample, ICSM consistently has the largest PC (Figure 8.4).

Overall, when comparing wide-scoped conferences to narrow-scoped ones, we observe that the former receive more submissions but have smaller PCs, hence higher review load than the latter. Of course, some PC members engage external reviewers, implying that the actual review load might be lower than the ratio RL of the number of submissions and the number of PC members. Moreover, the actual review load might be higher than RL if some of the PC members, *e.g.*, the PC chairs, do not review submissions at all or review less submissions than other PC members.

Figure 8.6 displays the variation of the review load RL. It is interesting to observe the increasing trend for ICSE ($\rho = 0.80^{**}$), resulting from increasingly more submissions each year and a PC size that does not tend to grow accordingly. ICSE and to a lesser extent FSE are also the conferences with the highest values of RL (Figure 8.5): since 2007, ICSE has stabilised to a ratio of 9 submissions per PC member (if each submission is reviewed on average by 3 PC members, this implies approximately 27 submissions being assigned to each PC member), followed by FSE with a ratio of 7 submissions per PC member

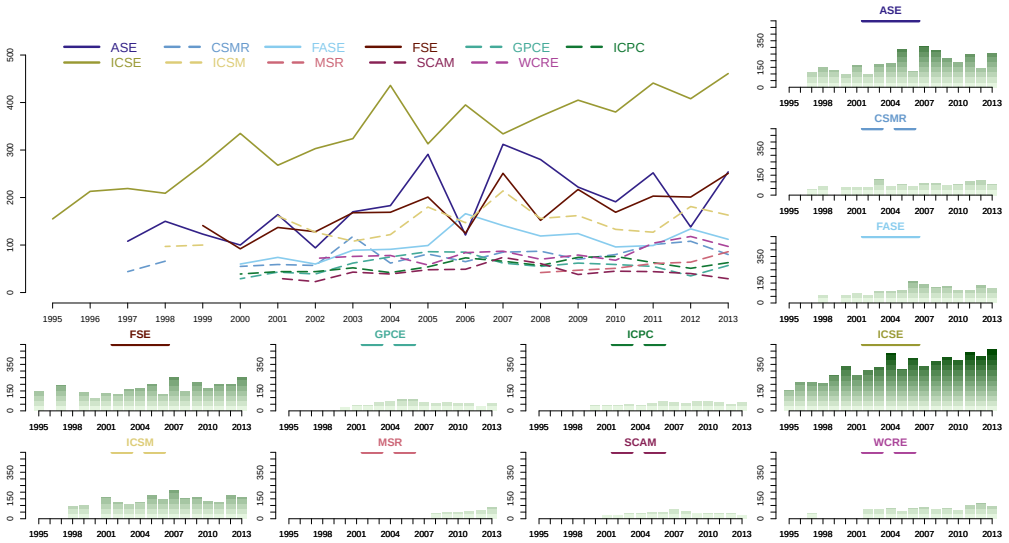


Figure 8.1: Variation of the number of submitted papers per year $\#SP$. "Gaps" correspond to older editions of the conferences, for which we could not retrieve the data. Wide-scoped conferences (solid lines) tend to receive more submissions than narrow-scoped ones (dashed lines): $p(\text{narrow,wide}) = 0.911^{**}$. Relations between individual conferences are visualised in the $\widetilde{\mathbf{T}}$-graph from Figure 8.3.

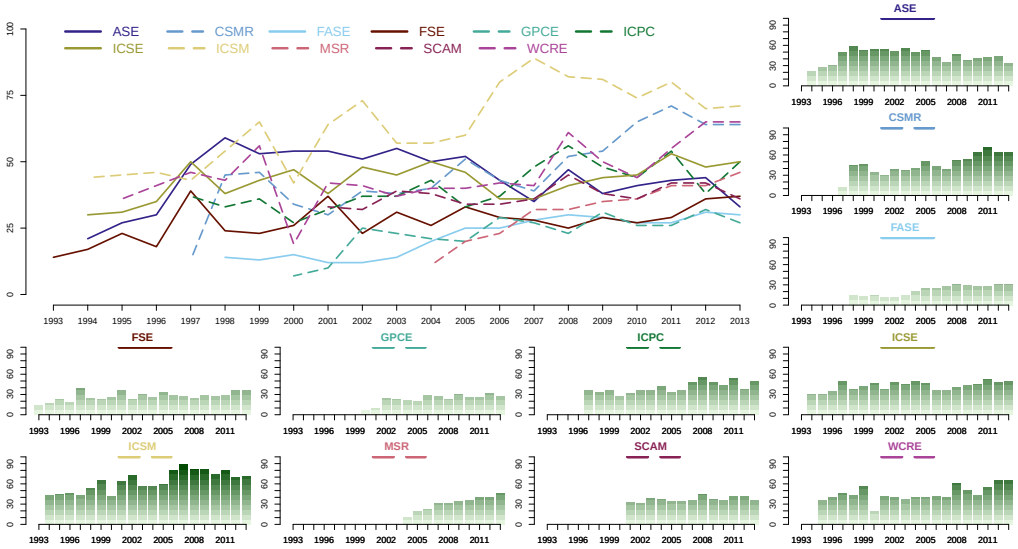


Figure 8.2: Variation of the number of programme committee members $\#C(c, y)$ per year. Wide-scoped conferences (solid lines) have smaller PCs than narrow-scoped ones (dashed lines): $p(\text{narrow,wide}) = 0.352^{**}$. Relations between individual conferences are visualised in the $\widetilde{\mathbf{T}}$-graph from Figure 8.4.

(i.e. 21 submissions assigned to each PC member). Other trends are visible for MSR (increasing, $\rho = 1^{**}$), ASE (increasing, $\rho = 0.78^{**}$), and GPCE (decreasing, $\rho = -0.65^{*}$).

Recognition of an extremely high review load for ICSE has lead the program co-chairs and steering committee of ICSE to adopt the Program Board model for its 2014 edition. This model follows a two-phase review process: first, a broad PC reviews the submissions, then a much smaller Program Board initiates, monitors, and guides the discussions with the PC members. In this way a balance can be found between a reduced review load and high quality of reviews. The Program Board model has recently been used by the International Requirements Engineering Conference (RE) as well as the International Conference on Model-Driven Engineering Languages and System (MoDELS).

Wide-scoped conferences tend to receive more submissions, but typically have smaller PCs than narrow-scoped ones, resulting in higher review load. A potential approach to lowering review loads is the Program Board model used recently by conferences such as RE and MoDELS, and adopted by ICSE for its 2014 edition.

8.3.2 Stability

To study stability of the conference community we analyse the PC turnover and author turnover for the considered conferences. We also assess the influence of the availability of a PC charter on the PC turnover. Indeed, since conferences are frequently subject to charters or guidelines that require PC renewal, they must strike a balance between PC turnover and continuity.

PC turnover

Inviting PC members from previous editions helps to ensure continuity and coherence. When studying $RNC(c, y, 1)$, the PC turnover rate w.r.t. the previous year, we observe a wide variation (Figure 8.7): the lowest observed PC renewal ratio is 8.8% for SCAM 2005, and the highest is 93% for GPCE 2007. Using Spearman rank correlation, no clear monotonic trends could be inferred for $RNC(c, y, 1)$ for any of the conferences. To check whether the chosen width of the sliding window (one year) affects our results, we repeated the analysis by looking at a longer period of 4 previous years. The PC turnover rate

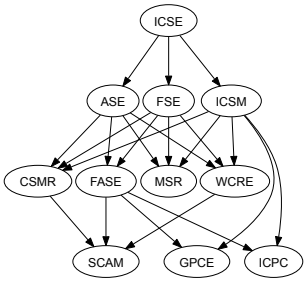


Figure 8.3: $\widetilde{\mathbf{T}}$-graph for the number of submitted papers $\#SP$.

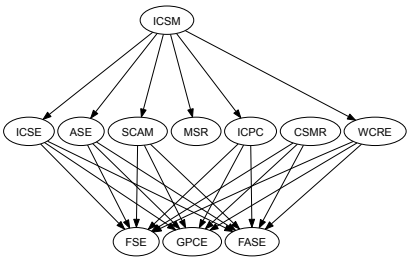


Figure 8.4: $\widetilde{\mathbf{T}}$-graph for the number of PC members $\#C$.

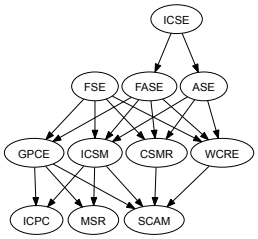


Figure 8.5: $\widetilde{\mathbf{T}}$-graph for the review load $\#RL$.

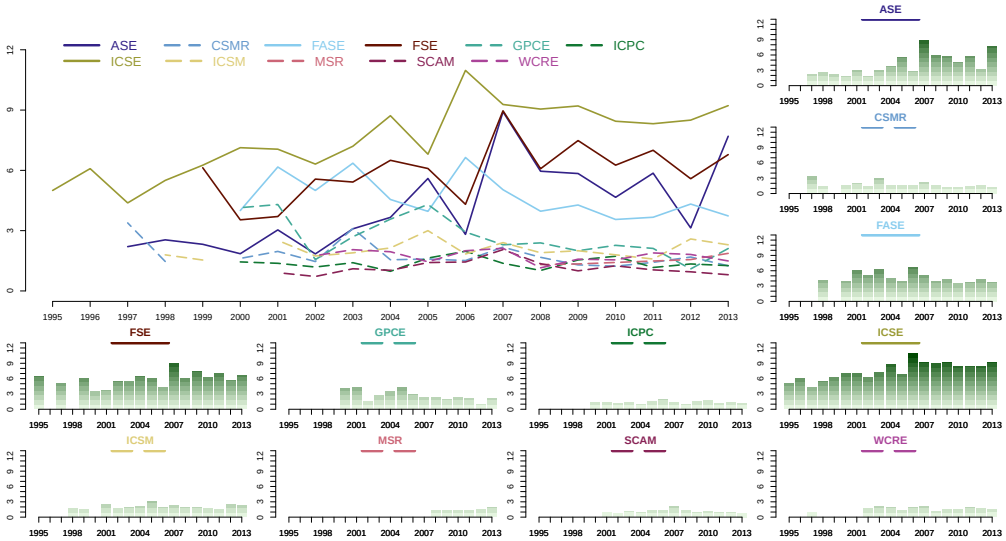


Figure 8.6: Variation of the review load RL per year. Wide-scoped conferences (solid lines) have higher review load than narrow-scoped ones (dashed lines): $p(\text{narrow,wide}) = 0.974^{**}$. Relations between individual conferences are visualised in the $\widetilde{\mathbf{T}}$-graph from Figure 8.5.

$RNC(c, y, 4)$ shows similar behaviour: no clear trends, and a big range of values for the metric (from 8.8% for SCAM 2005 to 85% for GPCE 2007).

Confidence intervals for the relative effect reveal that wide-scoped conferences usually have higher values than narrow-scoped ones: $p(\text{narrow,wide}) = 0.776^{**}$. One can conjecture that PC turnover might be correlated with the number of potential PC members, *i.e.*, senior researchers active in research areas covered by the conference. The first step towards estimating the number of potential PC members, and, hence, sustainability of the conference, is the Sustainability Ratio $SR(c, y, n)$ discussed in Section 8.3.6.

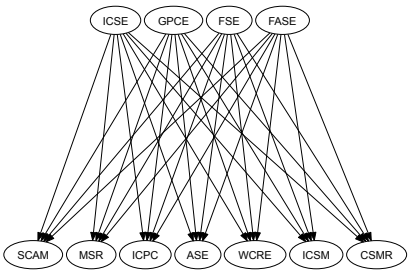


Figure 8.8: $\widetilde{\mathbf{T}}$-graph for the PC turnover w.r.t. previous year $RNC(c, y, 1)$.

We use $\widetilde{\mathbf{T}}$-graphs to compare distributions of RNC-metrics for different conferences. Two groups of conferences become apparent in Figure 8.8: the wide-scoped ICSE, FSE and FASE and the narrow-scoped GPCE, have consistently higher values of $RNC(c, y, 1)$ than the other considered conferences.

We conjecture that presence of GPCE in the high-turnover group of conferences might be due either to relevance of the GPCE topics to a broader scientific community, or failure to establish a core community. One could also argue that the small PC size of GPCE plays a role: it is easier to renew a large fraction of a small rather than a large PC. However, while the PC of GPCE is smaller, *e.g.*, than that of ICSE, it is not significantly

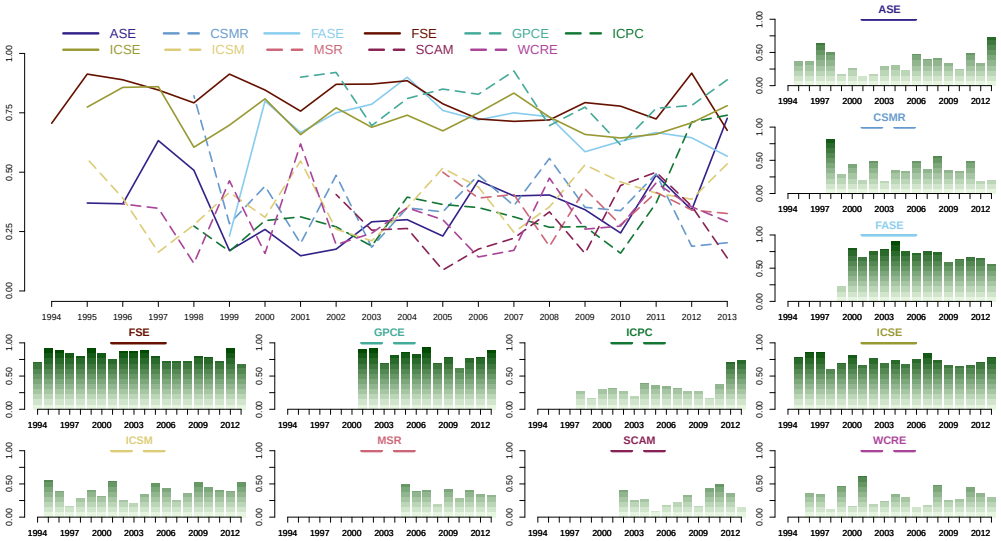


Figure 8.7: Variation of PC turnover w.r.t. previous year $RNC(c, y, 1)$. Wide-scoped conferences (solid lines) usually have higher values than narrow-scoped ones (dashed lines): $p(\text{narrow,wide}) = 0.776^{**}$. The $\widetilde{\mathbf{T}}$-graph in Figure 8.8 confirms this, except for GPCE (which behaves like a wide-scoped conference) and ASE (which behaves like a narrow-scoped conference).

different in size than the PC of FSE or FASE (as resulted from applying $\widetilde{\mathbf{T}}$ to the values of $\#C$, visualised in Figure 8.4). A more detailed investigation of the reasons for high PC turnover in GPCE goes beyond the scope of this chapter.

Wide-scoped conferences ICSE, FSE and FASE, and narrow-scoped GPCE have consistently higher PC turnover than the other conferences. ASE (wide-scoped) appears to be an outlier with respect to the other wide-scoped conferences.

PC charter availability

The PC turnover rate often depends on external factors, such as the presence of some implicit or explicit policy or charter requiring part of the PC to be renewed every year. Conference charters commonly recommend that no PC member should serve four consecutive terms. The ACM SIGSOFT policy, applicable to ICSE, FSE, ASE and SCAM, requires at least one-third of the PC members to change each year. Our results confirm that ICSE and FSE (as well as FASE and GPCE) always conform to this requirement of the ACM SIGSOFT policy. While ASE should also adhere to this policy, this is true for only 10 out of 18 editions considered, with the most recent noncompliance being in 2010. Similarly, SCAM adheres to this policy in only 5 out of 12 editions considered, with the most recent noncompliance being in 2013.

ICSE and FSE always conform to the "at least one third" PC renewal policy, while ASE and SCAM do not.

The official charter of FASE (established at FASE 2004) requires that about 50% of the PC members should be chosen from among PC members of the previous two editions. Let us loosely interpret "about 50%" as the interval between 40% and 60%. Although FASE did not always satisfy this requirement, it has been adhering to this charter regulation since 2009, and the threshold of 50% has always been exceeded since the establishment of the charter.

Finally, the PC charter of ICSM, applied since 2004, requires between 10% and 30% of the members to be new with respect to the preceding year's PC. Surprisingly, in nine years following the application of the charter (2004–2012) only ICSM 2007 adhered to this renewal policy (24,7%). All other ICSM editions in this period *exceed* the required renewal percentage reaching 53% in 2007.

ICSM almost always exceeds the percentage of new PC members prescribed by the charter.

Author turnover

Author turnover is another indicator of conference stability. One can expect that conferences attract local researchers, that might not be ready to participate in the subsequent edition organised at a different location. However, one can also expect a relatively stable group of "core" researchers that are likely to contribute to a number of conference editions. Similar tradeoffs as with PC turnover are in place. On the one hand, a very unstable community might fail to achieve a critical research mass. On the other hand, a very stable community, in which the same authors publish over and over again, can be a sign of introversion.

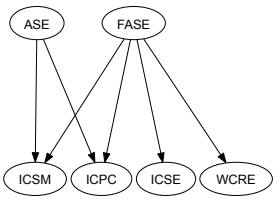


Figure 8.9: $\widetilde{\mathbf{T}}$-graph for the author turnover w.r.t. four previous years $RNA(c, y, 4)$. Statistically indistinguishable conferences are not displayed.

We observed that all considered conferences are very dynamic and have high author turnover $RNA(c, y, 1)$ with respect to the previous edition: from 2000 onwards values exceed 70% and can reach as high as 98% for GPCE 2000 or SCAM 2012, suggesting high openness to new authors. Overall, the lowest author turnover rate of 58% is observed for ICPC in 1999, and the highest one of 100% for FASE 1999. Given these high values, we hypothesise that the "new" authors are not necessarily new but rather returning after a short period of absence.

Figure 8.10 visualizes $RNA(c, y, 4)$ that takes the four preceding years into account. We still observe high turnover (ranging from 49% for WCRE in 2006 to 86% for ASE in 2006), but the results become less extreme. The $\widetilde{\mathbf{T}}$-graph of Figure 8.9 further reveals differences between the wide-scoped ASE and FASE, and the narrow-scoped ICPC, ICSM and WCRE. Considered together, the wide-scoped conferences tend to have higher author turnover (p(narrow,wide) = 0.684**). This is not surprising since wide-scoped conferences have a larger pool of tentative authors that can contribute to them. This finding is concurrent with the observation that such software engineering conferences as ASE, CAV, FASE, FM, FSE, ICSE, ISSTA are quite interdisciplinary [27].

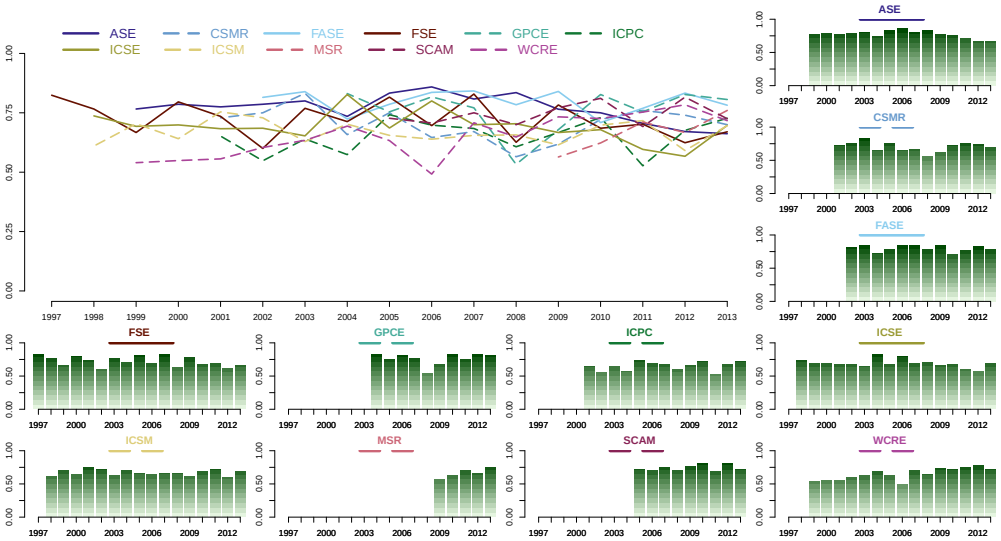


Figure 8.10: Variation of the author turnover w.r.t. four previous years $RNA(c, y, 4)$. Wide-scoped conferences (solid lines) tend to have higher values than narrow-scoped ones (dashed lines): $p(\text{narrow,wide}) = 0.684^{**}$. Relations between individual conferences are visualised in the $\widetilde{\mathbf{T}}$-graph from Figure 8.9.

All conferences have high author turnover: since 2000 there are more than 70% new authors with respect to the previous edition, and more than 50% with respect to the previous four editions. Wide-scoped conferences tend to have higher author turnover than the narrow-scoped ones.

We emphasize that attracting new authors and renewing the PC do not necessarily prevent introversion in wide-scoped conferences (*e.g.*, the new authors could also be PC members). We therefore further investigate openness and introversion in subsections 8.3.3 and 8.3.4, respectively.

8.3.3 Openness

To evaluate openness, *i.e.*, the ability of a conference to attract new authors, we study the evolution of $RPNA(c, y, 4)$—the fraction of papers published at conference c in year y for which none of the co-authors has published at conference c in the 4 preceding years (Figure 8.11). The lower this value, the less "open" the conference is to new authors. By focusing on the percentage of papers rather than the percentage of authors, we avoid the phenomenon of "new faces", *e.g.*, junior co-authors of researchers that have already published at the conference. Moreover, by looking at 4 previous years only, we remove the impact of the amount of historic data on the evaluation of openness.

At the high end of the scale (more open communities), the $\widetilde{\mathbf{T}}$-graph of Figure 8.13 reveals ASE, FASE and SCAM: ASE and SCAM tend to be more open than ICPC and WCRE. Similarly, FASE tends to be more open than ICPC, WCRE, ICSM, MSR and ICSE.

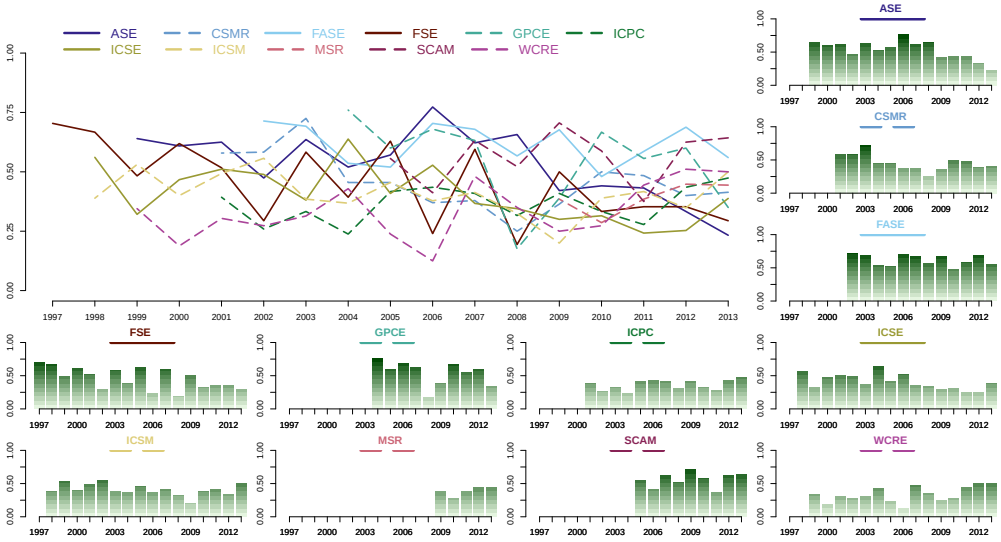


Figure 8.11: Variation of $RPNA(c, y, 4)$—the fraction of papers for which none of the co-authors has previously published at conference c in the 4 preceding years to y. Wide-scoped conferences (solid lines) usually have higher values (are more open) than the narrow-scoped ones (dashed lines): p(narrow,wide) $= 0.627^{**}$. Relations between individual conferences are visualised in Figure 8.13.

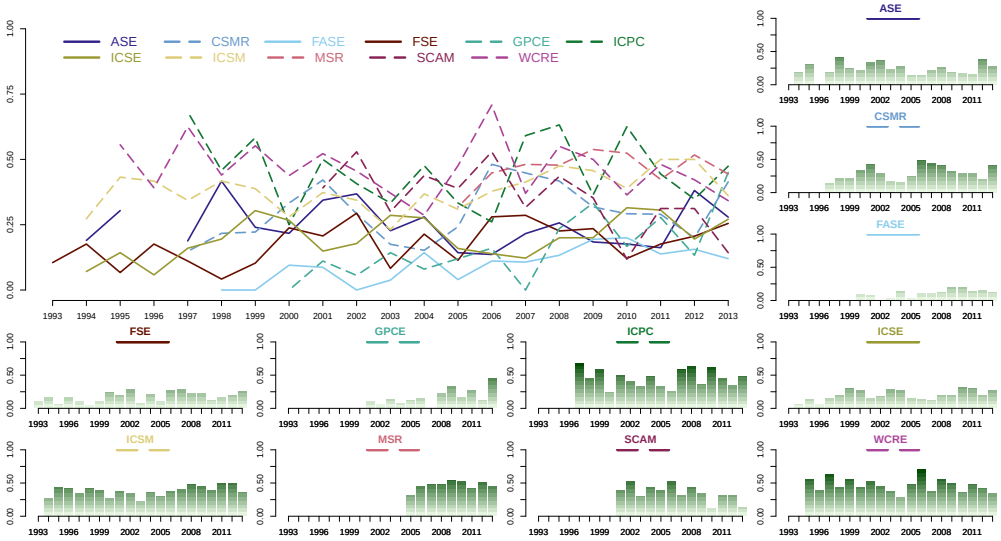


Figure 8.12: Variation of $RAC(c, y, 0)$—the fraction of papers co-authored by PC members in the same year. Wide-scoped conferences (solid lines) tend to have lower values (are less introvert) than the narrow-scoped ones (dashed lines): p(narrow,wide) $= 0.144^{**}$. Relations between individual conferences are visualised in Figure 8.14.

At the low end of the scale (more closed communities), no statistically significant ranking can be inferred between any of the conferences. Overall, the clearest trends are exhibited by ICSE and FSE, and in recent years ASE: over the years, the percentage of papers for which none of the co-authors has ever published at these conferences in the preceding four editions is decreasing (ASE: $\rho = -0.629^{*}$; FSE: $\rho = -0.613^{**}$; ICSE: $\rho = -0.612^{*}$). In other words, ICSE, FSE and ASE are becoming decreasingly open.

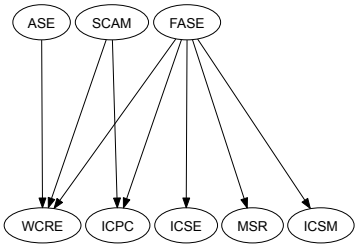


Figure 8.13: $\widetilde{\mathbf{T}}$-graph for the ratio of papers by new authors w.r.t. four previous years $RPNA(c, y, 4)$. Statistically indistinguishable conferences are not displayed.

ASE, FASE and SCAM are among the most open communities (have a low entrance barrier). ICSE, FSE and ASE are becoming increasingly less open over the years.

8.3.4 Introversion

To evaluate introversion we study the evolution of $RAC(c, y, 0)$, the fraction of papers co-authored by PC members in the same year (Figure 8.12). When studying introversion it is essential to consider the PC chairs and the General chairs as PC members because they are still influential enough within the community. Again, the $\widetilde{\mathbf{T}}$-graph reveals differences between the wide-scoped ICSE, FSE, FASE, and ASE, and the narrow-scoped ICSM, CSMR, WCRE, ICPC, MSR, SCAM and GPCE when considered as two groups: the wide-scoped conferences tend to be less introvert (p(narrow,wide) $= 0.144^{**}$).

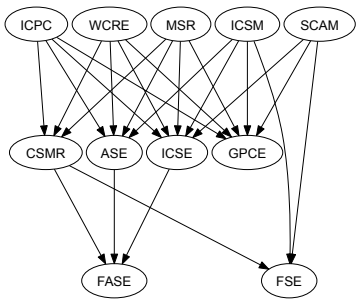


Figure 8.14: $\widetilde{\mathbf{T}}$-graph for the ratio of papers co-authored by PC members in the same year $RAC(c, y, 0)$.

The values range between 0% (no introversion at all) for FASE 2002 or GPCE 2007 and 71% for WCRE 2006 (high introversion). Overall, WCRE, ICPC, MSR, SCAM and ICSM tend to be the most introvert (*i.e.*, tend to have higher values of $RAC(c, y, 0)$ than other conferences), while GPCE, FSE and FASE tend to be the least introvert. However, both FASE ($\rho = 0.79^{**}$) and GPCE ($\rho = 0.73^{**}$) are becoming increasingly introvert. Repeating the analysis with a longer time window using $RAC(c, y, 4)$ concurs with the previous ranking.

WCRE (on average 47% of the papers accepted each year are co-authored by PC members), MSR (46%), ICPC (46%), ICSM (38%) and SCAM (35%) tend to be the most introvert conferences. In contrast, FASE (10%), GPCE (14%) and FSE (18%) tend to be the least introvert. Overall, wide-scoped conferences tend to be less introvert than narrow-scoped ones.

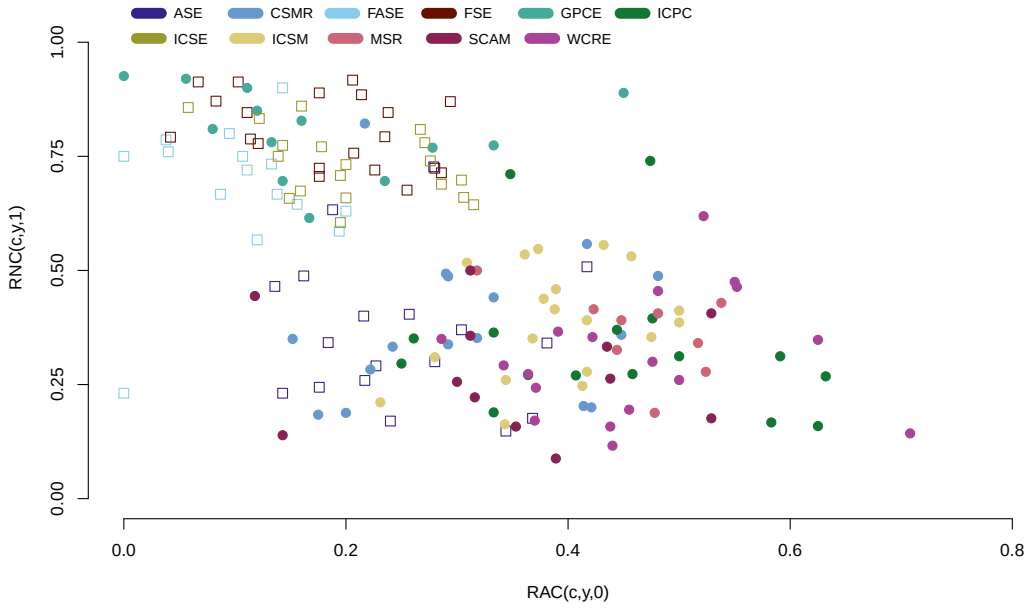


Figure 8.15: For narrow-scoped (filled circles) rather than wide-scoped (empty squares) conferences, higher PC turnover is associated with smaller fractions of PC papers among the accepted papers.

Systä et al. [290] have observed negative linear correlation between $RAC(c, y, 0)$ and $RNC(c, y, 1)$: "the less there is PC turnover, the greater is the proportion of PC papers among the accepted papers". However, they also noticed that the negative correlation does not necessarily hold for individual conference series, *e.g.*, for CSMR the correlation was found to be reversed. We have replicated their study for our larger set of conferences and longer period, and we have observed a similar phenomenon (Figure 8.15): $RAC(c, y, 0)$ and $RNC(c, y, 1)$ show a moderately-strong negative linear correlation ($r = -0.58^{**}$). The moderate correlation is applicable more to the narrow-scoped ($r = -0.54^{**}$) rather than the wide-scoped ($r = -0.45^{**}$) conferences. At the level of individual conference series, none of the conferences confirms this trend. This finding means that inherent features of conferences (rather than differences between individual editions of the same conference) influence the association between a lower PC turnover and a greater proportion of PC papers among the accepted papers.

When observing the whole set of conferences or focusing on narrow-scoped conferences only, higher PC turnover is associated with lower introversion. However, the claim does not typically hold at the level of individual conference series.

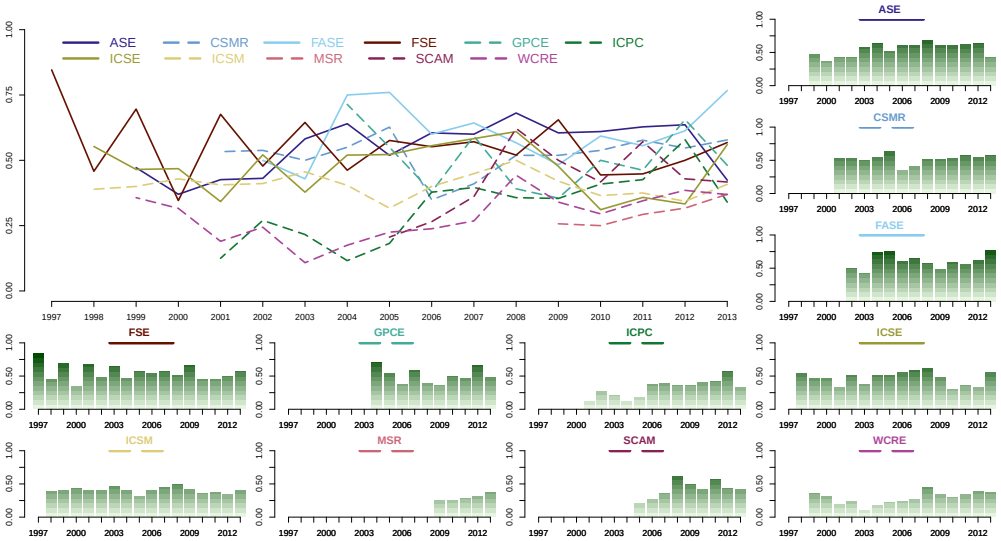


Figure 8.17: Variation of $RCnA(c, y, 4)$—the ratio of PC members for conference c in year y that have never co-authored a paper at any of the four preceding instances of c. The PCs of narrow-scoped conferences (dashed lines) are more representative of their communities than those of wide-scoped conferences (solid lines): $p(\text{narrow,wide}) = 0.801^{**}$. Relations between individual conferences are visualised in Figure 8.16.

8.3.5 Representativeness

Ensuring a right balance between continuity and renewal is not the only sign of a healthy PC. We believe that PC members should be representative of their respective communities, *i.e.*, they should largely be established authors within those communities. *A fortiori* this should also hold for the PC chairs and General chairs.

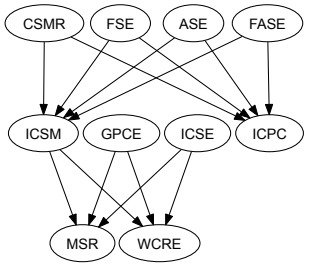


Figure 8.16: $\widetilde{\mathbf{T}}$-graph for $RCnA(c, y, 4)$.

However, not all PC members should be expected to have published at a conference before. For example, PC chairs often invite some PC members with industrial affiliation or background, who typically do not publish often. In the case of FASE, the charter explicitly mentions that "the PC should include at least 10% of members with industrial affiliation or background". Similarly, senior researchers may be invited to serve on the PC, even if they prefer to publish at more prestigious venues or in journals instead of conferences. Nevertheless, we expect high representativeness of the PC, given that all considered conferences are well-established and we analyse at least ten years of history.

To investigate this, we studied $RCnA(c, y, 4)$, the fraction of PC members not having (co)authored papers at conference c during any of the preceding four editions. The higher the values, the less representative we claim the PC to be. Overall, we observe a wide range

of values (Figure 8.17): the lowest is 12% for ICPC 2001 (the earliest ICPC edition for which the metric can be computed), and the highest is 85% for FSE 1997. The $\widetilde{\mathbf{T}}$-graph of Figure 8.16 reveals that values of $RCnA(c, y, 4)$ for WCRE are lower than for all other conferences except ICPC, MSR and SCAM; ICSM and ICPC are both lower than any of CSMR, FSE, ASE and FASE. When viewed together, narrow-scoped conferences have more representative PCs than wide-scoped ones (p(narrow,wide) = 0.801^{**}).

Narrow-scoped conferences have more representative PCs than wide-scoped ones. For example, the WCRE PC is consistently more representative of its community than the PCs of all other conferences except ICPC, MSR and SCAM. ICSM and ICPC also have representative PCs.

MSR ($\rho = 0.9^{*}$) and ICPC ($\rho = 0.731^{**}$) exhibit the clearest increasing trends, suggesting that their PCs are becoming increasingly less representative of their respective communities. However, one should also take into account the fact that the time series for MSR and ICPC started from values that were much lower than for the other conferences, and that the highest values (2013 for MSR and 2012 for ICPC) still fall within the same range as for the other conferences.

ICPC starts off in 2001 by having the most representative PC out of all conferences, but its PC is becoming increasingly less representative over the years.

8.3.6 Sustainability

We next investigate whether the conference communities comprise core authors that have not served on the PC at previous editions of the conference. Presence of such *unsung heroes* among the core authors can be seen as a measure of conference health: such core authors, either senior researchers or PhD students, can and should serve as a pool of candidates for PC membership in the future; moreover, they can contribute to increasing the representativeness of the PCs for their respective communities, should this be desired. Alternatively, the low number of unsung heroes can be seen as a sign of degeneration of the community/field. However, there is high variation in the number of PC members and number of authors between the different conferences (*e.g.*, in 2007 ICSM had 89 PC members, while ICSE had only 35). Therefore, we study variation across conferences of $SR(c, y)$, the average number of unsung heroes per PC member.

ICSE and FSE stand out as the conferences with the most sustainable pools of PC candidates in recent years (Figures 8.19 and 8.18): since 2010 there are on average 1.6 potential replacements (core authors that did not serve on the PC of any of the preceding four editions) for each PC member for ICSE, and 1.2 for FSE. Using the Spearman correlation we also confirm (Figure 8.18) increasing trends for ASE ($\rho = 0.907^{**}$), FSE ($\rho = 0.832^{**}$) and ICSE ($\rho = 0.754^{**}$)—they have increasingly more sustainable pools of PC

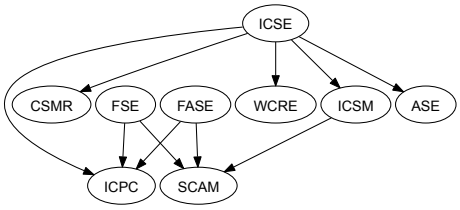


Figure 8.19: $\widetilde{\mathbf{T}}$-graph for $SR(c, y, 4)$.

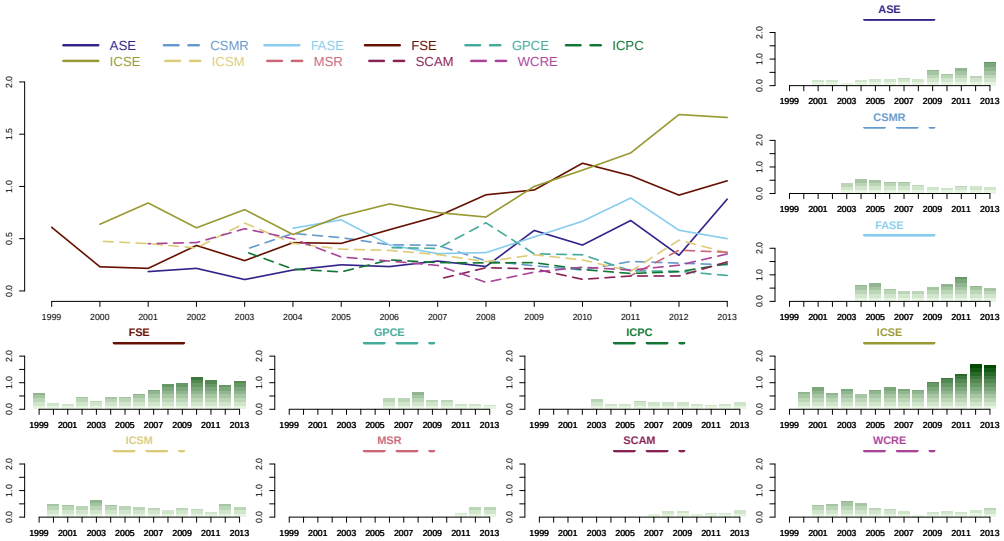


Figure 8.18: Variation of $SR(c, y)$—the average number of *unsung heroes* per PC member. Wide-scoped conferences (solid lines) have more sustainable pools of PC candidates than narrow-scoped ones (dashed lines): $p(\text{narrow,wide}) = 0.804^{**}$. Relations between individual conferences are visualised in Figure 8.19.

candidates; in contrast, GPCE ($\rho = -0.929^{**}$), CSMR ($\rho = -0.773^{*}$) and WCRE ($\rho = -0.721^{**}$) exhibit decreasing trends—finding qualified PC candidates is becoming more challenging. When viewed together, wide-scoped conferences have more sustainable pools of PC candidates than narrow-scoped ones ($p(\text{narrow,wide}) = 0.804^{**}$).

There is high potential to renew the PC members for ICSE, FSE and ASE from within the core authors publishing at these conferences. In contrast, CSMR, WCRE, ICPC and GPCE have increasingly lower potential to renew the PC from within their author communities.

8.3.7 Prestige

It is generally believed that the number of submissions to a conference is directly proportional to its scientific impact [196]. To verify this intuition, we study the relation between the conference impact factor $CI(c)$ and the number of submissions $\#SP(c, y)$. Since $CI(c)$ has a single value per conference series and is computed for the 2000-2012 interval, we contrast it against the mean number of submissions during the same period. Computing the mean is meaningful since for each conference series the distribution of $\#SP(c, y)$ over the years is close to normal (the Shapiro-Wilkinson test fails to reject the normality hypothesis at 95% confidence level). We observe very strong positive and statistically significant linear correlation ($r = 0.95^{**}$), confirming that more prestigious conferences attract more submissions.

The higher the scientific impact of a conference, the more submissions it attracts.

The acceptance ratio has been related to conference prestige, *e.g.*, by Manolopoulos [192], and further debated by Laplante et al. [173]. It is also generally believed that the acceptance rate of a conference is inversely proportional to its scientific impact [196]. To verify this intuition, we study the relation between the conference impact factor $CI(c)$ and the mean acceptance rate $RA(c, y)$ for the 2000-2012 period, following the same reasoning as above. We observe strong negative linear correlation ($r = -0.77$), suggesting that conferences with higher acceptance rates indeed have lower scientific impact. This conclusion is concurrent with the findings of Chen and Konstan [48] based on a study of 600 ACM conferences, and with a study of a database conference ADBIS by Manolopoulos [192].

Higher impact conferences tend to have lower acceptance rates.

Taken together our conclusions further concur with the rules "reverse engineered" by Küngas et al. [167] from the CORE ERA ranking: lower acceptance rate (< 0.363) *in combination with* higher number of citations (< 706) corresponds to *A*-ranked conferences (ICSE, ASE, FSE and ICSM in our list), while higher acceptance rate or lower acceptance rate in combination with lower number of citations correspond to *B*-rated conferences.

8.4 Discussion

In this section we combine and interpret the findings for each considered conference. Furthermore, when some of our indicators reveal that conference health is threatened we propose strategies to improve it.

We stress that our analysis does not suggest "optimal values" for conference health metrics. Based on our experience with software metrics, as well as with different software engineering conferences, we do not believe in "one size fits all" thresholds. Consensus on this kind of "optimal values" is hard to achieve, and once achieved it will miss the specific context of each individual conference. Moreover, the following discussion shows that a relative comparison of different conferences is still possible even in absence of "optimal values".

CSMR has a low author turnover. Together with average openness, this suggests a relatively stable community. Moreover, the PC turnover is low, the introversion and representativeness are neither high nor low. This suggests that the CSMR authors and PC members are relatively disconnected. As a remediation strategy, one can consider inviting some of the "unsung heroes" to join the PC. However, sustainability of the PC pool candidates is decreasing, *i.e.*, finding qualified PC members becomes more difficult.

ICSM, ICPC, and WCRE exhibit similar trends: all these conferences have low author and PC turnover and average openness. The PC members actively contribute to the paper body implying that the PC is representative for its author community but there is also high introversion. Sustainability of the PC for these conferences is low and exhibits a decreasing trend. Improving on the PC sustainability as well as increasing the PC turnover could be achieved by reducing the size of the PC. This would, however, increase the PC members' workload and, therefore, might endanger the quality of the reviews and negatively affect the scientific impact of the conference. Still, we have observed that not only ICSE, but also ASE and FSE with *CI*-values comparable to those of ICSM have much higher ratios of the number of submissions to the number of PC members. Hence, higher reviewer load is not necessarily detrimental for the scientific impact and might be

beneficial for ICSM, ICPC, and WCRE. As a countermeasure against introversion and closedness, these conferences may consider changing the traditional single-blind review by a double-blind review scheme as this scheme is fairer to authors from less prestigious institutions, that are unlikely to be among the PC members [274]. Another alternative might consist in employing a double-open review scheme, although it may increase the reluctance of prospective PC members to join the PC.

GPCE, as opposed to CSMR, ICSM, ICPC and WCRE, has a high PC turnover and low introversion, suggesting that the PC members do not tend to publish at GPCE. However, the author turnover is low. This might be indicative of, on the one hand, relevance of the GPCE topics to the broader scientific community, or alternatively, the failure to establish a relatively stable and sufficiently large core group ready to serve on the PC. Sustainability of the PC for GPCE is low and exhibits a decreasing trend.

ICSE is a highly prestigious conference with high author and PC turnover, suggesting that both authors and PC members can be selected from a large pool of candidates. ICSE, however, appears to become increasingly more difficult to enter, which in the long run might put the sustainability of the candidate-author pool in jeopardy. The ICSE steering committee is aware of this, since they have a mentoring program for new authors and, moreover, decided to adopt a Program Board model as of 2014.

Similarly to ICSE, FSE has high author turnover, high PC turnover and low introversion. Moreover, for FSE the sustainability of the PC is high and shows an increasing trend. Openness of FSE fluctuated greatly in early 2000s and seems to have stabilised at a relatively low level, suggesting that the same warning as for ICSE seems to apply.

FASE, in general, exhibits typical features of wide-scoped conferences, *e.g.*, high author and PC turnover, little introversion and a pool of PC candidates that seems to be highly sustainable in recent years. Unlike ICSE and FSE, FASE is very open, suggesting the future sustainability of the conference.

Unlike FASE, ASE has a low PC turnover but similarly to FASE, it is very open and has a high author turnover. Similarly to CSMR, the PC community of ASE seems to be disconnected from the author community (the representativeness is low). Therefore, similarly to CSMR we suggest inviting some of the "unsung heroes" of ASE to join the PC. Unlike CSMR, the sustainability of the PC candidates pool is higher for ASE.

MSR seems to be a conference at cross-roads: it has a relatively low but increasing workload, caused by the increasing number of submissions and not adequately balanced by the increasing size of the PC. Its PC is representative of the author community but is becoming less so due to a relatively low PC turnover and increasing author turnover. Finally, it does not yet have a charter, but the charter is in preparation and is expected to be presented at MSR 2014. While high author turnover and workload are more typical for wide-scoped conferences, low PC turnover is more common for narrow-scoped conferences. Sustainability of the conference is comparable to the more sustainable narrow-scoped conferences such as ICSM.

SCAM is an introvert conference, highly open to new authors and exhibiting high author turnover. Sustainability of the PC candidates pool is low, *i.e.*, finding qualified PC candidates is challenging. This suggests that while SCAM succeeds in attracting new authors, it does not succeed in retaining those authors for a number of subsequent editions. SCAM has the lowest Conference Impact $CI = 15$ among the conferences considered, which might be the reason why only few authors would explicitly target SCAM on a regular basis.

Towards MCDM

The above discussion of the relative strengths and weaknesses of each conference based on the considered health metrics can be seen as a first step of a multiple-criteria decision making/analysis process (MCDM or MCDA) [300, 356], a well-studied area of operations research. Based on our data, a prospective conference author with no preceding experience in this particular conference might apply MCDM techniques to help decide whether or not to submit a paper using the openness, introversion and prestige metrics as decision criteria. Similarly, a conference steering committee interested in the health assessment of their conference relatively to other software engineering conferences, can consider health assessment as a hierarchical MCDM problem [300, p. 2] with workload, stability, openness, introversion, representativeness, sustainability and prestige as criteria (cf. Table 8.2), and individual metrics as the corresponding subcriteria.

8.5 Related work

In this chapter, we built upon the work by Systä, Harsu and Koskimies [290], replicating and extending their introversion study of 6 software engineering conferences (ICSE, ICSM, ICPC, CSMR, WCRE, and GPCE) observed during the 2004-2009 period[16]. Our study takes into account more conferences, uses a wider range of metrics, considers longer time periods, and uses $\widetilde{\mathbf{T}}$-graphs to perform pairwise comparison of metrics across conferences. We assess the health of software engineering conferences with respect to several criteria (community stability, openness to new authors, introversion, representativeness of the PC with respect to the authors' community, availability of PC candidates, and scientific prestige), and we track how each health factor evolves over time for each of the considered conferences.

Similar in spirit, although not focusing on software engineering conferences[17], Biryukov and Dong [31] investigate how the communities represented by different research subfields within computer science as well as the corresponding conferences are evolving and communicating to each other. They use DBLP data to survey the development of authors' careers, and extract features that can help distinguish between conferences of different rank. For example, *population stability* (akin to our discussion of author turnover from Section 8.3.2) is recognised as "a candidate feature that helps to distinguish between the top and non-top venues". They find that lower-rank conferences are characterised by higher turnover (typically the newcomers constitute about 75-85% and the leavers up to 88% of all authors) and high percentage of pure newcomers among the newcomers (about 75%), the latter suggesting high openness (cf. our discussion in Section 8.3.3). Our results suggest that wide-scoped conferences tend to have higher author turnover than narrow-scoped ones.

Also related, although again not focusing on software engineering conferences, are the works of Elmacioglu and Lee [84], Zhuang et al. [353] and Sakr and Alomari [253], who recognise the impact of PC quality on the conference quality. Elmacioglu and Lee [84] extract information about PC composition for a number of conferences from Calls for Papers published on DBWorld[18], and construct a collaboration graph for authors of these

[16] The GPCE data is collected over the 2002-2009 period.
[17] The only conference in common with our study is FSE.
[18] `http://research.cs.wisc.edu/dbworld/`

conferences from the ACM Guide[19]. By dividing the set of analysed conferences into two groups (reputable and questionable[20]), they show that (i) reputable conferences tend to have smaller PC sizes than less reputable ones (28.8 members on average as opposed to 69.6); (ii) PC members of reputable conferences typically have more publications than those of less reputable ones (complementary to our notion of representativeness, cf. Section 8.3.5); and (iii) most reputable conferences have PC members with high closeness values in the collaboration graph (the more central a node is, the lower its total distance to all other nodes) on average. In our case all considered conferences are well-established and would likely be labeled as "reputable". However, our results using the $\widetilde{\mathbf{T}}$-procedure reveal that ICSM has consistently the largest PC among the considered conferences (63.5 members on average), while FASE, GPCE and FSE have consistently the smallest ones (21.3, 23.1 and 26.7 members on average, respectively).

Similarly to us, Sakr and Alomari [253] also argue in favour of PC renewal: "it is quite unhealthy to have a fixed or slightly different list of members in the program committees for the different venues", since this "may have intended or unintended negative effects in the fairness of evaluating the research contributions or in the quality and variability of the conference programs". They analyse the composition of the PCs for four top-tier and prestigious database conferences (SIGMOD, VLDB, ICDE, EDBT) over a period of 10 years (2001–2010), and report the percentage of overlap in the PC between the different editions of each conference. Although their metric is similar in spirit to our $RNC(c, y, n)$ for different values of n, we cannot directly compare our results since they use a slightly different definition[21]. Nonetheless, both their analysis and ours suggest that from the PC composition viewpoint, the considered conferences are relatively healthy.

Inbreeding, related to introversion, has been studied by Inanc and Tuncer [135], albeit with a different meaning. While we consider the conferences for which PCs favour acceptance of papers submitted by PC members, they refer to a situation wherein PhDs are employed by the very same institution that trained them during their doctoral studies (denoted academic inbreeding). Using a data set of scholars from Turkish technical universities, the authors show that inbreeding has negative consequences, affecting apparent scientific effectiveness as measured by one's h-index.

Our "health assessment" of software engineering conferences can be further put in the context of quality evaluation (*i.e.*, ranking) of scientific venues. We have used *SHINE* (the Simple H-INdex Estimator [2]) to rank the conferences and show, *e.g.*, that higher-impact conferences attract more submissions but tend to have lower acceptance rates. Numerous alternative approaches to ranking scientific venues have been proposed (*e.g.*, [64, 152, 153, 167, 196, 197, 228, 276, 347]), but they fall beyond the scope of this chapter. For example, da Silva et al. [64] propose a ranking scheme for scientific conferences based on ranking the PC members. Their rank measure is based on the h-index of the PC (*i.e.*, the maximum number x of PC members such that each PC member has h-index [131] at least x) as well as the inequality (spread) of the h-indices of the PC members, computed using the Gini index [102].

[19] Currently the ACM Digital Library.

[20] Distinction based on personal experience of Elmacioglu and Lee.

[21] Their metric is defined as the ratio of the number of PC members in common between two editions and the number of distinct PC members of the same two editions.

8.6 Future work

The analysis carried out in this chapter can be extended in many different ways. In this section we present what we believe to be the most interesting future research directions.

8.6.1 More objects of study

The first group of future work directions pertains to enriching the data collected by including more objects in our study.

For instance, throughout our analysis, we restricted ourselves to research papers submitted to the main conference track only. This was a deliberate choice since we did not want to make the analysis overly complex. However, some conferences allow for both long and short papers in the main track. Both types of papers are quite different in nature and should perhaps be accounted for differently. Many conferences also offer different kinds of parallel tracks, in order to facilitate presentation of, *e.g.*, early research achievements, PhD research, industrial results, research tools, and research projects.

A straightforward but labour-intensive extension would consist in replicating our study for other computer science research domains (e.g. databases, artificial intelligence, theoretical computer science). Such replications would also allow one to relate the conference health factors to the considered domain.

While we focused on software engineering *conferences*, it would be useful to apply a similar approach to software engineering *journals*. Many of the metrics proposed would need to be modified to take into account the specificities of journal publication. For example, there is no such thing as a "journal programme committee"[22] since, for every submitted paper, reviewers are assigned "ad hoc" based on the topic of the paper and the expertise and availability of reviewers. It is probably also much harder to obtain historical data about the reviewers for papers that have been published in a particular journal. We are not aware of such data being freely available. Another main difference is that journals do not tend to work with submission deadlines[23]. Papers can be submitted at any time, and the time and process required for reviewing, revising, and resubmitting papers is also much more flexible than for conference papers. We would also have to come up with new metrics that reflect characteristics that are important for journal publications such as, for example, the average time from submission to final acceptance of the paper.

8.6.2 More information about the study objects

Rather than extending the study to include more study objects, one can consider enriching the data set by including additional information about the study objects (conferences) we have already considered. First of all, while we excluded short-paper tracks from consideration, we have included in our data submissions to the main conference track that have been accepted as short papers. Distinction between main track submissions accepted as full papers and accepted as short papers would allow us to investigate impact of the differences between the two kinds of papers on, *e.g.*, openness and introversion.

We can also record additional information about authors and PC members, *e.g.*, their gender (which could be inferred automatically based on their names, *e.g.*, as described

[22] The so-called editorial board is not the same, since it does not contain the full list of reviewers of journal papers.

[23] We exclude here the so-called journal "special issues"

in Chapter 6), geographical location, seniority, and research expertise. Presence of this information would allow us to obtain additional insights in representativeness of the PC with respect to the author community in terms of gender, geographical location, seniority, and research expertise (cf. Section 8.3.5). Moreover, availability of this additional information about the PC members, would allow us to extend the charter adherence study initiated in Section 8.3.2. For instance, the ICSM charter explicitly states that the "Program Chairs should make every effort to achieve diversity on the PC with respect to gender, geographic distribution, experience, and industry versus academic experience" and requires PC to include members "whose areas of expertise sufficiently cover" different sub-areas of ICSM-related research. Furthermore, we can also check whether evidence can be found for PC chairs/General conference chairs favouring committee members of a certain gender, geographical location, seniority, and research expertise.

Information about the PC members' research expertise would make it feasible, with some effort, to come up with a fully objective measure of the conference scope, allowing us to refine the distinction between wide-scope and narrow-scope conferences considered in the current article. To achieve this, in addition to the PC members' research expertise one would need to extract the solicited research topics as found in the call for papers and the websites of each conference, as well as the topics of the actually accepted and published papers. Most software engineering conferences publish their proceedings through a digital library (IEEE, ACM, or Springer) that requires to follow a strict classification scheme of the paper topic and subject area. Based on this, tracking and analysis the different software engineering topics that are covered by each conference, should be possible, which in their turn can form the basis of a new metric that objectively quantifies the narrowness of scope of the conference.

8.6.3 Using the data

The final group of future work directions is related to possible applications and extensions of our work. To start with, we could consider studying the probability of paper acceptance. While the acceptance ratio $RA(c, y)$ can be used as a first rough estimate, better techniques should include information about the authorship (*e.g.*, is the paper more likely to be accepted if the author is also member of the PC or a frequent co-author of some PC members?) and topic(s) of the paper. Based on a more refined acceptance probability estimate, one can also consider modeling the prospective author behaviour by incorporating the effort she needs to put in preparing a submission, and potential benefits such as increased scientific visibility. Similarly to positive emotions affecting work engagement mediated by hope [221], we expect that positive emotions associated with a paper accepted at a prestigious conference in the preceding year give rise to hope, making the author to be more inclined to submit the paper in the year afterwards.

As mentioned in Section 8.4, deciding whether to submit a paper to a conference can be seen as an MCDM problem [356] with openness, introversion and prestige metrics as decision making criteria. Similarly, conference organizers or steering committees can use MCDM to assess conference health, using any of the considered health criteria and associated health metrics of their liking. Using these kinds of analyses, conference organisers can furthermore obtain additional insights in the impact of different policies recommended in the literature, such as increasing the acceptance rate, on future submissions (cf. [173]) and on evolution of the conference communities ("how will the GPCE author community look like in 2020?"). We believe that this kind of customized analysis is far better than

trying to come up with a single index of conference health or a "one size fits all" threshold. Nevertheless, the application of MCDM techniques is, in our case, challenged by the temporal dimension of the data, as well as by the limited number of conferences considered compared to the number of criteria.

Another promising research direction pertains to collaboration between researchers, both at intra-conference level (cohesion) and inter-conference level (coupling). *Conference cohesion* can be evaluated by studying topics of the papers accepted and cooperation between the authors submitting to the same conference: if the author community can be clearly separated into distinct groups not linked by common publications or research topics, the conference is essentially an amalgamation of several disconnected communities. Lack of cohesion might also explain high PC turnover. *Coupling* between different conferences arises if the same persons are PC members or authors of different conferences. The amount of coupling between conferences may play a role in how some of the conference health indicators evolve over time. For example, when renewing the PC, chairs can easily look for new qualified and willing PC candidates in "coupled" conferences. Improved knowledge of conference coupling would allow one to replicate the study on *groups* of related conferences. Indeed, if there is a high overlap in the author and PC communities of two conferences, these conferences could be aggregated into one since an author that has frequently published at one of the venues cannot be reasonably considered as a "new face" for the other. Finally, empirical evidence of high coupling between conferences may help the conference organisers to adapt conference organisation, *e.g.*, by merging or co-locating the events or by trying to differentiate them more. Significant overlaps are likely to be present for some of the narrow-scoped conferences we have studied (in particular, ICSM, WCRE and CSMR that target more or less the same subdomains of software engineering). This is one of the main reasons why CSMR and WCRE have decided to merge their conferences into a single event, called CSMR-WCRE in 2014.

Finally, one can study "migration" of researchers from one conference to another, from one research topic to another, akin to migration of software developers between different projects within the GNOME ecosystem [198].

8.7 Conclusions

The goal of this chapter was to assess how the health of software engineering conferences evolves over time, in terms of a variety of criteria: stability and representativeness of the PC with respect to the authors' community, openness to new authors, introversion, availability of PC candidates and scientific prestige. To this extent, we proposed a suite of metrics aiming to measure these health indicators. The used metrics and methodology are applicable to other scientific conferences as well.

Our analysis, covering 11 software engineering conferences over a period of more than 10 years, indicate that these conferences are relatively healthy: balanced PC turnover (high enough to avoid introversion, yet low enough to ensure continuity and coherence), high openness to new authors ("new" in terms of both turnover with respect to previous years as well as not having published at that conference ever before), and moderate introversion (in terms of fraction of papers co-authored by PC members). Nonetheless, we observed important differences across conferences according to the aforementioned criteria, suggesting different strengths, weaknesses, opportunities and

threats. We also observed important differences between wide-scoped and narrow-scoped software engineering conferences.

We do not believe in identifying a "holy grail" of conference health assessment, *i.e.*, a small set of metrics or optimal values that are universally indicative of conference health. Based on our previous experience with software metrics, as well as our experience as author, PC member or PC chair of different software engineering conferences, we are convinced that consensus on optimal metrics values is hard to achieve and, once achieved, will miss conference-specific context. Instead, we presented a range of comparative analyses and countermeasures that can be useful for steering committees, programme committees, prospective authors and researchers in different ways. A range of strategies could be used to increase conference health, such as: inviting "unsung heroes" to join the PC, reducing the size of the PC, replacing the traditional single-blind review process by either a double-blind or double-open review process, reducing reviewer load by resorting to a Program Board review model or a "rolling deadline" model (such as the one followed by the VLDB conference series), merging conferences together or differentiating them better.

Chapter 9

Conclusions

This chapter concludes this dissertation by revisiting each research question from Chapter 1 and discussing our main contributions, then sketching some directions for future research. Additional details are available in the chapters that cover the research questions.

9.1 Contributions

The work included in this dissertation emerged from the observation that contributors to online software communities are diverse individuals: they have different age, gender, personalities, educational and cultural backgrounds, geographical locations, motivations to contribute, and expertise. Yet, despite these differences, they succeed to work together effectively and productively, creating and maintaining software, or sharing knowledge.

This dissertation explored, through a series of empirical studies, if and how different facets of diversity among participants in online software communities or between the communities themselves relate to individual-level and community-level outcomes. We focused on few representative communities, such as the community around the GNOME desktop environment—one of the largest and most popular OSS ecosystems, the community around the STACK OVERFLOW question and answer site—the largest and most popular knowledge sharing platform for programmers, or the community around GITHUB—currently the largest code hosting platform in the OSS world. Through these studies we explored a number of research questions, as follows.

RQ1. How diverse are OSS communities in terms of developer expertise?

To address this question, we considered two dimensions of developer expertise. First, in Chapter 2, we distinguished between different activities carried out by contributors to the GNOME ecosystem (*i.e.*, expertise in different domains of OSS development), among which coding, localization, and development documentation were the most prominent. Given this distinction, we explored using a portfolio of statistical techniques how the

workload and specialisation of labour of GNOME contributors vary in relation to these activities. We found the GNOME ecosystem to follow a heavy-tailed distribution of activity, typical of OSS: a large fraction of contributors are largely inactive, while a small number of contributors carry most of the development weight. Overall, contributors prefer to restrict themselves to a small number of activity types. This fact is especially evident for the many occasional contributors, who typically contribute to a single project and a single activity type (mostly localization). In contrast, frequent contributors exhibit different contribution patterns. Although in aggregate they also specialise, tending to prefer coding, we observed a positive correlation between their workload, their participation in many projects within the ecosystem, and their involvement in many activity types, indicative of a high level of commitment to GNOME. This empirical case study allowed us to confirm that there is no such thing as a uniform ecosystem of projects and contributors. When taking into account the activity types and the workload, there is a lot of variation across projects and across contributors, but with a clear preference towards the domains of coding, localization, development documentation and build systems. This empirical case study also included important methodological contributions. Our proposed set of metrics, of which particularly important are the specialisation metrics defined based on the Gini inequality index, can be reused easily for studying other software ecosystems. An additional contribution consists in introducing $\widetilde{\mathbf{T}}$-graphs, a novel approach for reporting the results of comparing multiple distributions.

Second, in Chapter 3, we studied expertise in different programming languages from a community-level point of view (as opposed to individual-level in Chapter 2). Our main contribution was a measure of expertise diversity with respect to a certain technical skill (we chose knowledge of programming languages), inspired by measures of linguistic diversity from the social sciences. To implement this measure, we tackled the controversial computer science domain of similarities between programming languages. To this end, we also introduced a novel measure of similarity between programming languages (which represents an interesting contribution by itself), based on shared knowledge of the languages by developers participating in STACK OVERFLOW. Besides being a methodological contribution, our programming language expertise diversity measure can be applied, *e.g.*, in community monitoring scenarios. Indeed, if tracked in a dashboard-like application, our measure can signal when an OSS community is at risk of not having enough maintainers for code implemented in a certain programming language. Therefore, risky languages can be discovered on time, and preventive action can be taken to ensure the maintainability of affected components.

RQ2. How does participating in social Q&A impact the working rhythms of OSS developers?

To address this question, we continued our exploration of diversity of activities in online software communities from a broader perspective. Rather than distinguishing between different activities carried out *within* a given community (Chapter 2), in Chapter 4 we studied effects of participating simultaneously *across* communities. Specifically, we studied the interplay between coding and knowledge sharing activities performed by OSS developers contributing to both GITHUB and STACK OVERFLOW. One of our main contributions was the assembly of a joint data set from the two sources, in which contributors active on both platforms are identifiable, which permitted linking their activities across the two resources and also over time. The techniques used to create

this data set were later refined in Chapter 5, where they have been applied to create a longitudinal data set of contributors to both `r-help` mailing lists and Stack Exchange question and answer sites. To the best of our knowledge, we are the first to create and analyse such overlapping data sets between Stack Exchange and OSS communities.

The joint GitHub and Stack Overflow data set allowed us to better understand participation in the two communities. First, we focussed on differences in Stack Overflow involvement of the GitHub developers. We found a direct relationship between GitHub commit activity and Stack Overflow question answering activity: the more active a committer, the more answers she gives. In other words, highly productive committers tend to take the role of a "teacher", being more actively involved in providing answers than in asking questions. Similarly, the more active an answerer, the more commits she authors. In other words, top users on Stack Overflow are "superstars" rather than "slackers": they do not just compete for reputation points and badges, but are actually active (OSS) software developers. In contrast, we found an inverse relationship between GitHub commit activity and Stack Overflow question asking activity: active GitHub committers ask fewer questions than others; less active question askers produce more commits. Overall, these findings suggest that an activity-based ranking of Stack Overflow contributors reflects one extracted from their OSS contributions to GitHub, increasing the confidence in the reliability of social signals based on Stack Overflow (*e.g.*, answering questions on Stack Overflow can be seen as a proxy for one's commit activity on GitHub).

Next, we studied whether the working rhythms of GitHub contributors are related to their Stack Overflow activities. We observed that asking questions on Stack Overflow influences how developers distribute their time over commits on GitHub, while answering questions on Stack Overflow does not seem to have the same effect. Specifically, developers who ask many questions on Stack Overflow commit changes to GitHub in bursts of intense activity followed by longer periods of inactivity, *i.e.*, they focus their attention at any given time. We conjecture that this effect is due to differences between more novice and more experienced developers: we have seen that the developers asking questions are predominantly the less active (presumably more novice) ones, while those providing answers are the more active (presumably more experienced) ones; this observation seems to suggest that the more experienced developers have no trouble also contributing their knowledge on Stack Overflow, since doing so does not seem to affect their working rhythms; in contrast, less experienced developers incur a higher cost of participating in Stack Overflow, since their activities there leave a mark on their working rhythms on GitHub.

Finally, we associated GitHub commits and Stack Overflow questions and answers over time, in an attempt to understand whether activities in the two platforms show signs of coordination. We widely found that the rate of asking or answering questions on Stack Overflow is related to the rate of commit activities in GitHub. In other words, despite the interruptions incurred, Stack Overflow activities accelerate GitHub committing. Although this finding holds for many different groups of developers, those frequently contributing to GitHub as well as those that have been involved in GitHub for sufficiently long time seem to benefit the most from participating in Stack Overflow in terms of the speed with which they commit changes.

RQ3. How are social Q&A sites changing knowledge sharing in OSS communities?

The previous research question established Stack Overflow as an effective professional expertise development platform for software developers, with positive individual-level effects on working rhythms and productivity associated with participating in it. Focusing on gamification, perhaps the most distinctive feature of social Q&A platforms such as Stack Overflow, here we sought to understand some of the community-level effects of gamification and participating in social Q&A on knowledge-sharing in online software communities in general.

To address this question, we performed two studies. The first study, comprising Chapter 5, charted the changes in behaviour of contributors to mailing lists as they migrate into the gamified Stack Exchange knowledge-sharing environment. We started by assembling a joint data set for R (a widely-used tool for data analysis), which integrates data from the `r-help` mailing list and the most prominent two Stack Exchange sites for R support, Stack Overflow and Cross Validated, connecting the same people over time in the different venues. Using this data set, we found that user support activities show a strong shift away from mailing lists and towards Stack Exchange, with interesting phenomena being associated with the transition. In particular, we observed that the movement to social, gamified Q&A is correlated with an increase in the engagement of knowledge providers, and an increase in their rapidity of response by a factor of 4. These findings support the incorporation of gamification elements into community and tool design, to increase user engagement and, indirectly, increase the popularity of the community/tool. For example, gamification may be of particular interest in knowledge communities from closed environments such as commercial corporations, where knowledge is proprietary and the set of potential knowledge providers is limited. Moreover, specifically for software engineering, any collaborative tools used by developers, such as issue trackers, code review tools, internationalisation support systems, or Q&A platforms, could all benefit from incorporating gamification elements into their design. While there is a significant body of literature on *why* gamification features should positively influence participation in knowledge communities, facilitate user contributions, and foster productivity, our study is among few that added concrete evidence on *how* gamification features and community design affect productivity of the contributors. This study also offered important methodological contributions, such as linking user identities across different repositories, or appropriately reconstructing discussion threads from email archives, such that comparing activity in communities organised in different ways (*e.g.*, mailing lists and Stack Overflow) is sound.

The second study, comprising Chapter 6, also focused on global effects associated with incorporating gamification into community design, but from a different perspective, that of gender representation and participation patterns. This study quantitatively investigated two different types of communities, the gamified one around the Stack Overflow Q&A site, and the non-gamified ones around the mailing lists of two popular web Content Management Systems, Drupal and WordPress. The main driver behind this study was the anecdotal evidence suggesting that Stack Overflow actively discourages the participation of women through its heavy use of (male-oriented) gamification elements. Since gender is not explicitly recorded in any of the studied communities, a special tool that infers gender based on name and nationality was developed, and can be considered a contribution by itself. Then, the communities were compared in terms of number

of male and female participants, and type and length of participation by each gender. Unsurprisingly, we found that women represent a minority in all three communities, in line with the overall gender representation in OSS. More surprisingly, two classes of communities emerged in terms of participation. In one class, represented by Drupal and WordPress communities, women and men show broadly similar contribution patterns in terms of questions, answers, and length of engagement. However, in the other class, represented by Stack Overflow, women disengage sooner than men, although relative to the duration of their stay in the community they are at least as active. Together with an observed tendency to remain anonymous on Stack Overflow more so than in the Drupal and WordPress communities, this finding draws attention to the potential drawbacks of incorporating gamification elements into community design. Despite producing excellent technical content, Stack Overflow seems to be a relatively "unhealthy" community, since it sets up higher barriers to entry for its participants, particularly for women. This further exacerbates the division in gender participation and engagement to online software communities, which produce quality content using incentives that discourage a more generalised participation.

RQ4. How can we improve existing MSR identity merging techniques?

To address this question, we selected in Chapter 7 two representative existing identity merging algorithms and studied their performance on data extracted from Gnome repositories. Gnome proved to be a very interesting data set, since having a long history, many contributors, and different activity types (see also Chapter 2) ensured there would be significant variance (and noise) in the identity data (names and email addresses) of its contributors. Indeed, after manually building an oracle that links the different aliases used by contributors to Gnome's source code repositories, we found, *e.g.*, developers that used 171 different aliases (name–email-address pairs), or 164 different names. Using this data set, we confirmed that the two existing algorithms selected are not robust enough with respect to the types of differences in contributor aliases found in Gnome. We then introduced a new identity merging algorithm inspired by Latent Semantic Analysis (LSA), a popular information retrieval technique. Rather than building many heuristics into our algorithm, we delegated dealing with domain-specific (*i.e.*, community-specific) differences in aliases to the noise reduction capabilities of LSA, making our technique more generic. Finally, using cross-validation, we showed that our algorithm outperforms the state-of-the-art in terms of precision and recall on noisy input data.

RQ5. How can we transfer MSR techniques to studying diversity among software engineering conferences?

We addressed this question in Chapter 8 through a bibliometric study trying to assess the health of 11 software engineering conferences over a period of more than 10 years. This study had two main contributions. Our first main contribution was showing how MSR techniques can be applied successfully beyond software repositories (in this case to studying diversity among software engineering conferences): how to integrate data from multiple ill-structured conference repositories, how to link personal identities across the different resources, how to define appropriate metrics, or how to visualise and statistically analyse these data. These efforts resulted in a publicly available data set, containing for each conference series historical data about accepted papers and composition of programme committees. This data set is a contribution by itself and can be used, *e.g.*, by

conference steering committees or programme committee chairs to assess their selection process, compare against other conferences in the field, or select who to invite to serve on program committees. Our second main contribution was reflecting back on the "health" of the software engineering scientific community to which we belong, using this data set and a novel suite of metrics.

9.2 Discussion

Reflecting on the results presented in this dissertation, we note that despite exploring different facets of diversity and different online software communities, two recurring themes emerged that can help explain our results and position them also beyond the software engineering domain.

The first theme is that of *specialisation*. In a collaborative effort, specialisation of participants is a key component to the group's success, as evidenced by ample scholarly interest over the years, especially in the organisational management literature (see, *e.g.*, [156] and [235] for starting points). Even though it has long been believed that "economies [tend to] flourish as knowledge is more and more separated and divided among agents" [235], suggesting that mutual ignorance across specialties (*i.e.*, high specialisation) is desirable, it seems essential in flourishing knowledge economies for "islands of shared knowledge" to exist [235] (*i.e.*, not too high specialisation).

Many of our results can be related to specialisation. For example, Chapter 2 tried to quantify how specialisation manifests itself in a well-established OSS knowledge economy—the GNOME ecosystem. Our findings suggest that islands of shared knowledge are indeed observable within the GNOME community (and are perhaps partly responsible for its success): contributors with a high level of commitment to GNOME (as manifested through their workload) tend to create islands of shared knowledge by participating in many projects within the ecosystem, or becoming involved in many activity types. Chapter 5 offered a similar finding. In the knowledge sharing community around R, most members chose to specialise in a single information sharing platform (either the mailing list or STACK EXCHANGE sites), while few contributed to both. However, as observed, members of this latter group creating the islands of shared knowledge (*i.e.*, bridging the two information sharing platforms) are the main actors in the R community, *i.e.*, they are responsible for more of the activity than those who specialise in any of the two venues. In contrast, Chapter 3 dealt with over-specialisation and its potentially devastating effects on a community's survival. Our model of expertise diversity shows how mutual ignorance across expertise domains (in our example programming languages), *i.e.*, too much specialisation, can have negative effects on the community as a whole.

The second theme is that of *signaling*. Signaling theory originated from both economics and biology. In economics, signals are personal attributes within the control of an individual, such as education, used by employers to assess potential employees [93, 279]. In biology, signals can be behaviors or ornaments in animals—such as a male peacock's heavy tail feathers—which, by being costly to maintain and making the animal more vulnerable to predators, increase its perceived quality. Having survived despite this handicap, the male peacock is perceived by potential female mates as being more attractive and more fit [349, 350]. Signaling theory has also been used as a framework for suggesting which pieces of information from online communities (signals) are more reliable when making sense of a person's expertise [266]. In an unpublished work referenced by Shami *et al.* [266],

Judith Donath distinguishes between three types of signals in digital artifacts: handicap signals, index signals, and conventional signals. Handicap signals are typically considered the most reliable of the three, since they are costly to produce (similarly to the male peacock's tale being a hindrance) and "the quality they signal is 'wasted' in the production of the signal, and the signal tends to be more expensive to produce for an individual with less of the quality" [266]. Referring to online communities, Shami *et al.* give the example of participation in online forums. "An employee with over 10,000 forum posts proves that she has enough time to be active in the forum, while still maintaining her job responsibilities. She is signaling that she is competent enough to balance her job responsibilities and help others." Many of our results can also be related to signaling. For example, in Chapters 2, 4, and 5 we have consistently found evidence for the presence of handicap signals in the online software communities we studied. In Chapter 2 we observed a positive correlation between a contributor's workload, their participation in many projects within the ecosystem, and their involvement in many activity types. In Chapter 4 we found a direct relationship between GitHub commit activity and Stack Overflow question answering activity. Finally, in Chapter 5 we observed that contributors who engage in both the R mailing list and Stack Exchange sites are more active than those who focus on just one platform.

9.3 Future work

We have identified numerous directions for future research at the end of each main body chapter. In this section we recapitulate some of the main directions.

Replication. The results included in this dissertation have been obtained on a small—although we believe representative—sample of online software communities: Gnome in **RQ1** and **RQ4**, GitHub in **RQ2**, Stack Overflow in **RQ1**, **RQ2**, and **RQ3**, and Cross Validated, R, Drupal, and WordPress in **RQ3**. Future research could verify not only to which extent our results are replicable, but also to which extent they are transferrable to other communities. To facilitate replication, we have made most of our data sets and tools publicly available. For example, our `genderComputer` tool that tries to infer a person's gender based on their name and nationality (used to address **RQ3**) is available at `http://github.com/tue-mdse/genderComputer`; our LSA-inspired identity merging algorithm (used to address **RQ4**) is available at `https://github.com/tue-mdse/aliasMerger`; and our bibliometric data set of software engineering conferences (used to address **RQ5**) is available at `https://github.com/tue-mdse/conferenceMetrics`.

More refined analyses. Our results can be further refined. For example, when studying, in **RQ2**, how participating in social Q&A impacts the productivity of OSS developers, we have only considered metadata about activities on GitHub and Stack Overflow. Future research could consider refining the classification of activities, to distinguish between questions/answers pertaining to different subjects (as expressed by tags) and commits pertaining to different projects. Then, using information retrieval techniques, questions/answers could be classified as being related, or not, to a given commit. For instance, we expect to observe a closer relation between commits and the topics of questions asked than the topics of answers given, as answers are more likely to pertain to the general knowledge of the individual.

Alternatively, when studying, in **RQ3**, how social Q&A sites are changing knowledge sharing in OSS communities, we have also only considered metadata about activities of contributors to `r-help` and Stack Exchange. To better understand the effects associated with a transition from mailing lists to social Q&A and, *e.g.*, whether mailing lists will eventually die off, future research could also consider analysing the *content* of the discussions from the two venues using natural language processing techniques. In this way, Q&A site designers could better understand the needs of the different groups of community members (*e.g.*, developers and users), and how to better cater for them (*e.g.*, which type of questions are better suited for social Q&A and which for mailing lists). Additionally, future research could consider analysing the *atmosphere* in the two information sharing venues using sentiment analysis techniques, to understand, *e.g.*, how "friendly" each community is, or how the personalities and communication styles of certain contributors influence the retention of community members.

Social diversity. To address diversity of OSS communities in **RQ1**, we have only considered technical aspects, such as knowledge of programming languages, or specialisation towards certain activity types. However, there are indicators that a team's social diversity is reflected in their technical performance. Greater social team diversity is often viewed as positive in offline settings, since it brings to the table a mixture of cultural and educational backgrounds, and access to different networks and broader ranges of information, which can enhance a team's creativity, adaptability, and problem solving skills. It is not yet clear if the mechanisms through which social diversity is said to influence team outcomes are transferrable to online settings, such as those from distributed OSS teams. Future research could investigate if and how diversity along social dimensions (*e.g.*, gender, experience, or nationality) in online settings is related to team outcomes.

Triangulation. Some of the results presented in this dissertation were purely quantitative. While we have done our best to select appropriate data collection and data analysis techniques in order to reduce the threats to internal validity, we do acknowledge the value of qualitative research. Using data collection and data analysis techniques associated with *both* quantitative and qualitative data, *i.e.*, a mixed methods approach, can help compensate the weaknesses of one method by the strengths of others [61, 82]. The study of effects associated with gamified knowledge sharing environments, comprising Chapter 5, best illustrates this approach. There we have followed a sequential explanatory strategy [82], by using qualitative user survey results to assist in interpreting and explaining the findings of the quantitative study, derived after statistical analysis. Similarly, we plan to strengthen the triangulation of our other quantitative findings through qualitative analysis, such as interviews and questionnaires.

Tool design. Our results for **RQ3** have shown that gamification mechanisms such as those found on Stack Overflow are very successful at encouraging members to contribute more, but less successful at retaining women. Based on these insights, future research could explore how gender and other human aspects, such as cultural background, could play a role when incorporating gamification and social media elements into the design of software engineering tools and the improvement of software engineering practices. Although different gender-sensitive design practices exist, even the most promising in terms of empowering female end users—such as the user-participatory design techniques—are hardly ever used [249]. Similarly, research on gamification and culture is in its infancy [150].

Bibliography

[1] The DBLP computer science bibliography. `http://dblp.uni-trier.de`. accessed October 2012.

[2] Simple h-index estimation. `http://shine.icomp.ufam.edu.br/index.php`. accessed November 2013.

[3] *Encyclopedia of gender and information technology*. IGI Global, 2006.

[4] Alison E. Adam. Hacking into hacking: Gender and the hacker phenomenon. *ACM SIGCAS Computers and Society*, 33(4):3, 2003.

[5] Paul J. Adams, Andrea Capiluppi, and Cornelia Boldyreff. Coordination and productivity issues in free software: The role of Brooks' law. In *International Conference on Software Maintenance (ICSM)*, pages 319–328. IEEE, 2009.

[6] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Very Large Data Bases*, pages 487–499. Morgan Kaufmann, 1994.

[7] Navid Ahmadi, Mehdi Jazayeri, Francesco Lelli, and Sasa Nesic. A survey of social software engineering. In *IEEE/ACM International Conference on Automated Software Engineering (ASE) Workshops*, pages 1–12. IEEE, 2008.

[8] Alfred V. Aho, Michael R Garey, and Jeffrey D. Ullman. The transitive reduction of a directed graph. *SIAM Journal on Computing*, 1(2):131–137, 1972.

[9] Michael G. Akritas, Steven F. Arnold, and Edgar Brunner. Nonparametric hypotheses and rank statistics for unbalanced factorial designs. *Journal of the American Statistical Association*, 92(437):258–265, 1997.

[10] Paul D. Allison. Measures of inequality. *American Sociological Review*, pages 865–880, 1978.

[11] Ashton Anderson, Daniel P. Huttenlocher, Jon M. Kleinberg, and Jure Leskovec. Discovering value from community activity on focused question answering sites: a case study of Stack Overflow. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 850–858. ACM, 2012.

[12] Judd Antin and Elizabeth F Churchill. Badges in Social Media: A Social Psychological Perspective. In *ACM CHI Conference on Human Factors in Computing Systems*. ACM, 2011.

[13] Giuliano Antoniol, Massimiliano Di Penta, and Mark Harman. Search-based techniques applied to optimization of project planning for a massive maintenance project. In *International Conference on Software Maintenance (ICSM)*, pages 240–249. IEEE, 2005.

[14] Shlomo Argamon, Moshe Koppel, Jonathan Fine, and Anat Shimoni. Gender, genre, and writing style in formal written texts. *Text*, pages 321–346, 8 2003.

[15] Janet Armentor-Cota. Gender and chat rooms. In *Encyclopedia of gender and information technology*, pages 361–364. IGI Global, 2006.

[16] Janet Armentor-Cota. Multiple perspectives on the influence of gender in online interactions. *Sociology Compass*, 5(1):23–36, 2011.

[17] B***. email about Stack Overflow. `http://pastebin.com/14TeVv79`, November 2012.

[18] Alberto Bacchelli, Luca Ponzanelli, and Michele Lanza. Harnessing Stack Overflow for the IDE. In *Workshop on Recommendation Systems for Software Engineering (RSSE)*, pages 26–30. IEEE, 2012.

[19] Richard P. Bagozzi and Utpal M. Dholakia. Open source software user communities: A study of participation in Linux user groups. *Management Science*, 52(7):1099–1115, 2006.

[20] Albert-László Barabási. The origin of bursts and heavy tails in human dynamics. *Nature*, 435:207–211, 2005.

[21] Victor R. Basili and David M. Weiss. A methodology for collecting valid software engineering data. *IEEE Transactions on Software Engineering*, SE-10(6):728–738, 1984.

[22] Gareth Baxter, Marcus Frean, James Noble, Mark Rickerby, Hayden Smith, Matt Visser, Hayden Melton, and Ewan Tempero. Understanding the shape of Java software. *ACM SIGPLAN Notices*, 41(10):397–412, 2006.

[23] Andrew Begel, Jan Bosch, and Margaret-Anne Storey. Bridging software communities through social networking. *IEEE Software*, 30(1), 2013.

[24] Andrew Begel, Jan Bosch, and Margaret-Anne Storey. Social networking meets software development: Perspectives from GitHub, MSDN, Stack Exchange, and TopCoder. *IEEE Software*, 30(1):52–66, 2013.

[25] Nicolas Bettenburg and Ahmed E. Hassan. Studying the impact of social structures on software quality. In *International Conference on Program Comprehension (ICPC)*, pages 124–133. IEEE, 2010.

[26] Bruce Bimber. Measuring the gender gap on the internet. *Social Science Quarterly*, 81(3):868–876, 2000.

[27] Christian Bird, Earl T. Barr, Andre Nash, Premkumar T. Devanbu, Vladimir Filkov, and Zhendong Su. Structure and dynamics of research collaboration in computer science. In *SIAM International Conference on Data Mining (SDM)*, pages 826–837, 2009.

[28] Christian Bird, Alex Gourley, Premkumar T. Devanbu, Michael Gertz, and Anand Swaminathan. Mining email social networks. In *International Working Conference on Mining Software Repositories (MSR)*, pages 137–143. ACM, 2006.

[29] Christian Bird, Alex Gourley, Premkumar T. Devanbu, Anand Swaminathan, and Greta Hsu. Open borders? Immigration in open source projects. In *International Working Conference on Mining Software Repositories (MSR)*, page 6. IEEE, 2007.

[30] Ken Birman and Fred B. Schneider. Viewpoint - program committee overload in systems. *Communications of the ACM*, 52(5):34–37, 2009.

[31] Maria Biryukov and Cailing Dong. Analysis of computer science communities based on DBLP. In *European Conf. Research and Advanced Technology for Digital Libraries (ECDL)*, pages 228–235, 2010.

[32] Inger Boivie. Women, men and programming: Knowledge, metaphors and masculinity. In *Gender Issues in Learning and Working with Information Technology: Social Constructs and Cultural Contexts*, pages 1–24. IGI Global, 2010.

[33] Andrea Bonaccorsi, Silvia Giannangeli, and Cristina Rossi. Entry strategies under competing standards: Hybrid business models in the open source software industry. *Management Science*, 52(7):1085–1098, 2006.

[34] Henny P. A. Boshuizen, Rainer Bromme, and Hans Gruber. *On the long way from novice to expert and how travelling changes the traveller.* Springer, 2004.

[35] Danah M. Boyd and Nicole B. Ellison. Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication*, 13(1):210–230, 2007.

[36] Joel Brandt, Philip J. Guo, Joel Lewenstein, Mira Dontcheva, and Scott R. Klemmer. Two studies of opportunistic programming: interleaving web foraging, learning, and writing code. In *ACM CHI Conference on Human Factors in Computing Systems*, pages 1589–1598. ACM, 2009.

[37] Tom Brijs, Koen Vanhoof, and Geert Wets. Defining interestingness for association rules. *Information Theories & Applications*, 10(4):370–375, 2003.

[38] Frederick P Brooks. *The Mythical Man-Month: Essays on Software Engineering.* Addison-Wesley, 1975.

[39] Bruce M Brown and Thomas P Hettmansperger. Kruskal-Wallis, Multiple Comparisons and Efron Dice. *Australian & New Zealand Journal of Statistics*, 44(4):427–438, 2002.

[40] Edgar Brunner and Ullrich Munzel. The Nonparametric Behrens-Fisher Problem: Asymptotic Theory and a Small-Sample Approximation. *Biometrical Journal*, 42(1):17–25, 2000.

[41] Edgar Brunner and Ullrich Munzel. *Nichtparametrische Datenanalysen: Unverbundene Stichproben*. Statistik und ihre Anwendungen. Springer, 2002.

[42] Margaret M. Burnett, Laura Beckwith, Susan Wiedenbeck, Scott D. Fleming, Jill Cao, Thomas H. Park, Valentina Grigoreanu, and Kyle Rector. Gender pluralism in problem-solving software. *Interacting with Computers*, 23(5):450–460, 2011. Feminism and HCI: New Perspectives.

[43] Judith Butler. *Gender Trouble: Feminism and the Subversion of Identity*. Routledge Classics. Routledge, 1999.

[44] Andrea Capiluppi, Patricia Lago, and Maurizio Morisio. Characteristics of open source projects. In *European Conference on Software Maintenance and Reengineering (CSMR)*, pages 317–327. IEEE, 2003.

[45] Andrea Capiluppi, Alexander Serebrenik, and Leif Singer. Assessing technical candidates on the social web. *IEEE Software*, 30(1):45–51, 2013.

[46] Andrea Capiluppi, Alexander Serebrenik, and Ahmmad Youssef. Developing an h-index for OSS developers. In *International Working Conference on Mining Software Repositories (MSR)*, pages 251–254. IEEE, 2012.

[47] Jason R. Casebolt, Jonathan L. Krein, Alexander C. MacLean, Charles D. Knutson, and Daniel P. Delorey. Author entropy vs. file size in the GNOME suite of applications. In *International Working Conference on Mining Software Repositories (MSR)*, pages 91–94. IEEE, 2009.

[48] Jilin Chen and Joseph A. Konstan. Conference paper selectivity and impact. *Communications of the ACM*, 53(6):79–83, 2010.

[49] Peter Christen. A comparison of personal name matching: Techniques and practical issues. In *International Conference on Data Mining (ICDM)*, pages 290–294. IEEE, 2006.

[50] Peter Christen, Tim Churches, and Markus Hegland. Febrl–a parallel open source data linkage system. In *Advances in Knowledge Discovery and Data Mining*, volume 3056 of *Lecture Notes in Computer Science*, pages 638–647. Springer, 2004.

[51] Pauline Rose Clance. *The impostor phenomenon: overcoming the fear that haunts your success*. Peachtree Publishers, 1985.

[52] Jacob Clark Blickenstaff. Women and science careers: leaky pipeline or gender filter? *Gender and Education*, 17(4):369–386, 2005.

[53] Aaron Clauset, Cosma Rohilla Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *SIAM Review*, 51:661–703, 2009.

[54] William S. Cleveland. Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, 74(368):829–836, 1979.

[55] Bentley Coffey and Patrick A. McLaughlin. Do masculine names help female lawyers become judges? Evidence from South Carolina. *American Law and Economics Review*, 11(1):112–133, 2009.

[56] Stack Overflow community. Top users on Stack Overflow: slackers or superstars? `http://meta.stackoverflow.com/q/12468`, 2009.

[57] Joel Cordeiro, Bruno Antunes, and Paulo Gomes. Context-based search to overcome learning barriers in software development. In *International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE)*, pages 47–51, 2012.

[58] Frank A. Cowell. Measurement of inequality. In *Handbook of Income Distribution*, volume 1 of *Handbooks in Economics*, pages 87–166. Elsevier, 2000.

[59] Frank A. Cowell and Stephen P. Jenkins. How much inequality can we explain? A methodology and an application to the United States. *Economic Journal*, 105(429):421–430, 1995.

[60] Paul S. P. Cowpertwait and Andrew V. Metcalfe. *Introductory time series with R*. Springer, 2009.

[61] John W. Creswell. *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. SAGE Publications, 2003.

[62] Jon Crowcroft, Srinivasan Keshav, and Nick McKeown. Viewpoint: Scaling the academic publication process to internet scale. *Communications of the ACM*, 52(1):27–30, 2009.

[63] Kevin Crowston, Kangning Wei, James Howison, and Andrea Wiggins. Free/Libre open-source software development: What we know and what we do not know. *ACM Computing Surveys (CSUR)*, 44(2):7, 2012.

[64] Roberto da Silva, José Palazzo Moreira de Oliveira, José Valdeni de Lima, and Viviane Moreira. Statistics for ranking program committees and editorial boards. *CoRR*, abs/1002.1060, 2010.

[65] Laura A. Dabbish, H. Colleen Stuart, Jason Tsay, and James D. Herbsleb. Social coding in GitHub: transparency and collaboration in an open software repository. In *ACM Conference on Computer Supported Cooperative Work (CSCW)*, pages 1277–1286. ACM, 2012.

[66] Yanja Dajsuren, Christine M. Gerpheide, Alexander Serebrenik, Anton Wijs, Bogdan Vasilescu, and Mark G. J. van den Brand. Formalizing correspondence rules for automotive architectural views. In *International ACM Sigsoft Conference on Quality of Software Architectures (QoSA)*, pages 129–138. ACM, 2014.

[67] Yanja Dajsuren, Mark G. J. van den Brand, Alexander Serebrenik, and Serguei Roubtsov. Simulink models are also software: modularity assessment. In *International ACM Sigsoft Conference on Quality of Software Architectures (QoSA)*, pages 99–106. ACM, 2013.

[68] Marco D'Ambros and Michele Lanza. Visual software evolution reconstruction. *Journal of Software Maintenance and Evolution*, 21:217–232, 2009.

[69] Whitney Darrow. *I'm Glad I'm a Boy! I'm Glad I'm a Girl!* Windmill Books, 1970.

[70] Paul A. David and Joseph S. Shapiro. Community-based production of open-source software: What do we know about the developers who participate? *Information Economics and Policy*, 20(4):364–398, 2008.

[71] Julius Davies, Daniel M. German, Michael W. Godfrey, and Abram Hindle. Software bertillonage: Finding the provenance of an entity. In *International Working Conference on Mining Software Repositories (MSR)*, pages 183–192. ACM, 2011.

[72] Felipe de Mendiburu. *Agricolae. Practical manual.* Faculty of Economics and Planning, La Molina National Agrarian University, La Molina, Lima, Peru, 2010.

[73] Daniel P. Delorey, Charles D. Knutson, and Christophe Giraud-Carrier. Programming language trends in open source development: An evaluation using data from all production phase Sourceforge projects. In *Workshop on Public Data about Software Development (WoPDaSD)*, 2007.

[74] Sebastian Deterding. Gamification: designing for motivation. *Interactions*, 19(4):14–17, 2012.

[75] Sebastian Deterding, Miguel Sicart, Lennart Nacke, Kenton O'Hara, and Dan Dixon. Gamification: Using game-design elements in non-gaming contexts. In *ACM CHI Conference on Human Factors in Computing Systems*, pages 2425–2428. ACM, 2011.

[76] Trung T. Dinh-Trong and James M. Bieman. The FreeBSD project: a replication case study of open source development. *IEEE Transactions on Software Engineering*, 31(6):481–494, 2005.

[77] J. R. Doyle and D. D. Stretch. The classification of programming languages by usage. *Man-Machine Studies*, 26(3):343–360, 1987.

[78] Drupal. Drupal mailing lists. `http://drupal.org/mailing-lists/`, 2012.

[79] Nicolas Ducheneaut. Socialization in an open source software community: A socio-technical analysis. *Computer Supported Cooperative Work*, 14(4):323–368, 2005.

[80] Olive Jean Dunn. Multiple comparisons among means. *Journal of the American Statistical Association*, 56(293):52–64, 1961.

[81] Charles W. Dunnett. A multiple comparison procedure for comparing several treatments with a control. *Journal of the American Statistical Association*, 50(272):1096–1121, 1955.

[82] Steve Easterbrook, Janice Singer, Margaret-Anne Storey, and Daniela Damian. Selecting empirical methods for software engineering research. In *Guide to advanced empirical software engineering*, pages 285–311. Springer, 2008.

[83] Michael Eckmann, Anderson Rocha, and Jacques Wainer. Relationship between high-quality journals and conferences in computer vision. *Scientometrics*, 90(2):617–630, 2012.

[84] Ergin Elmacioglu and Dongwon Lee. Oracle, where shall I submit my papers? *Communications of the ACM*, 52(2):115–118, 2009.

[85] Neil Ernst and John Mylopoulos. On the perception of software quality requirements during the project lifecycle. In *Requirements Engineering: Foundation for Software Quality*, volume 6182 of *Lecture Notes in Computer Science*, pages 143–157. Springer, 2010.

[86] Brynn Evans and Stuart Card. Augmented information assimilation: social and algorithmic web aids for the information long tail. In *ACM CHI Conference on Human Factors in Computing Systems*, pages 989–998. ACM, 2008.

[87] Jon Eyolfson, Lin Tan, and Patrick Lam. Correlations between bugginess and time-based commit characteristics. *Empirical Software Engineering*, pages 1–31, 2013.

[88] James D. Fearon. Ethnic and cultural diversity by country. *Journal of Economic Growth*, 8(2):195–222, 2003.

[89] Allan Fisher, Jane Margolis, and F. Miller. Undergraduate women in computer science: experience, motivation and culture. *ACM SIGCSE Bulletin*, 29(1):106–110, 1997.

[90] Lance Fortnow. Viewpoint - time for computer science to grow up. *Communications of the ACM*, 52(8):33–35, 2009.

[91] Massimo Franceschet. The role of conference publications in CS. *Communications of the ACM*, 53(12):129–132, 2010.

[92] Massimo Franceschet. The skewness of computer science. *Information Processing and Management*, 47(1):117–124, 2011.

[93] Robert Frank. *Microeconomics and behavior*. Granite Hill Publishers, 2008.

[94] Jill Freyne, Lorcan Coyle, Barry Smyth, and Padraig Cunningham. Relative status of journal and conference publications in computer science. *Communications of the ACM*, 53(11):124–132, 2010.

[95] K Ruben Gabriel. Simultaneous test procedures—some theory of multiple comparisons. *The Annals of Mathematical Statistics*, 40(1):224–250, 1969.

[96] David Gelernter and Suresh Jagannathan. *Programming linguistics*. MIT Press, 1990.

[97] Anne Gentle. *Conversation and Community: The Social Web for Documentation*. XML Press, 2009.

[98] Daniel M. German. The GNOME project: a case study of open source, global software development. *Software Process*, 8(4):201–215, 2003.

[99] Daniel M. German. Using software trails to reconstruct the evolution of software. *Journal of Software Maintenance and Evolution*, 16:367–384, 2004.

[100] Daniel M. German, Bram Adams, and Ahmed E. Hassan. The evolution of the R software ecosystem. In *European Conference on Software Maintenance and Reengineering (CSMR)*, 2013.

[101] Mohammad Gharehyazie, Daryl Posnett, Bogdan Vasilescu, and Vladimir Filkov. Developer initiation and social interactions in OSS: A case study of the Apache software foundation. *Empirical Software Engineering*, 2014.

[102] Corrado Gini. Measurement of inequality of incomes. *Economic Journal*, 31:124–126, 1921.

[103] Tamar Ginossar. Online participation: a content analysis of differences in utilization of two online cancer communities by men and women, patients and family members. *Health Communication*, 23(1):1–12, 2008.

[104] Paola Giuri, Matteo Ploner, Francesco Rullani, and Salvatore Torrisi. Skills, division of labor and performance in collective inventions: Evidence from open source software. *International Journal of Industrial Organization*, 28(1):54–68, 2010.

[105] Uri Gneezy, Muriel Niederle, and Aldo Rustichini. Performance in competitive environments: Gender differences. *The Quarterly Journal of Economics*, 118(3):1049–1074, 2003.

[106] Mathieu Goeminne and Tom Mens. A comparison of identity merge algorithms for software repositories. *Science of Computer Programming*, 78(8):971–986, 2011.

[107] Mathieu Goeminne and Tom Mens. Evidence for the Pareto principle in Open Source Software Activity. In *International Workshop on Software Quality and Maintainability (SQM)*. CEUR-WS workshop proceedings, 2011.

[108] Mathieu Goeminne and Tom Mens. Analysing ecosystems for open source software developer communities. In *Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry*. Palgrave-MacMillan, 2013.

[109] Sean P. Goggins, Christopher Mascaro, and Giuseppe Valetto. Group informatics: A methodological approach and ontology for sociotechnical group research. *Journal of the American Society for Information Science and Technology*, 64(3):516–539, 2013.

[110] Georgios Gousios. The GHTorrent dataset and tool suite. In *International Working Conference on Mining Software Repositories (MSR)*, pages 233–236. IEEE, 2013.

[111] Georgios Gousios, Eirini Kalliamvakou, and Diomidis Spinellis. Measuring developer contribution from software repository data. In *International Working Conference on Mining Software Repositories (MSR)*, pages 129–132. ACM, 2008.

[112] Georgios Gousios and Diomidis Spinellis. GHTorrent: Github's data from a firehose. In *International Working Conference on Mining Software Repositories (MSR)*, pages 12–21. IEEE, 2012.

[113] Georgios Gousios, Bogdan Vasilescu, Alexander Serebrenik, and Andy Zaidman. Lean GHTorrent: GitHub data on demand. In *International Working Conference on Mining Software Repositories (MSR), Data Track*, pages 384–387. ACM, 2014.

[114] Agueda Gras-Velazquez, Alexa Joyce, and Maïté Debry. Women and ICT. *Why are girls still not attracted to ICT studies and careers?*, 2009.

[115] Sherri Grasmuck, Jason Martin, and Shanyang Zhao. Ethno-racial identity displays on Facebook. *Journal of Computer-Mediated Communication*, 15(1):158–188, 2009.

[116] Joseph H. Greenberg. The measurement of linguistic diversity. *Language*, 32(1):109–115, 1956.

[117] Keith Grint and Rosalind Gill. *The Gender-Technology Relation: Contemporary Theory and Research: An Introduction.* Gender & society: feminist perspectives on the past & present series. Taylor & Francis Group, 1995.

[118] Anja Guzzi, Alberto Bacchelli, Michele Lanza, Martin Pinzger, and Arie van Deursen. Communication in open source software development mailing lists. In *International Working Conference on Mining Software Repositories (MSR)*, pages 277–286. IEEE, 2013.

[119] Zev Handel. What is Sino-Tibetan? Snapshot of a field and a language family in flux. *Language and Linguistics Compass*, 2(3):422–441, 2008.

[120] Sandra G. Harding. *The Science Question in Feminism.* Cornell University Press, 1986.

[121] Ahmed E. Hassan. The road ahead for mining software repositories. In *Frontiers of Software Maintenance (FoSM)*, pages 48–57. IEEE, 2008.

[122] Paul Heggarty. Beyond lexicostatistics: How to get more out of 'word list' comparisons. *Diachronica*, 27(2):301–324, 2010.

[123] Andrea Hemetsberger and Christian Reinhardt. Learning and knowledge-building in open-source communities a social-experiential approach. *Management Learning*, 37(2):187–214, 2006.

[124] James D. Herbsleb and Rebecca E. Grinter. Splitting the organization and integrating the code: Conway's law revisited. In *International Conference on Software Engineering (ICSE)*, pages 85–95. ACM, 1999.

[125] Amaç Herdağdelen and Marco Baroni. Stereotypical gender actions can be extracted from web text. *Journal of the American Society for Information Science and Technology*, 62(9):1741–1749, 2011.

[126] Susan C. Herring. Gender and democracy in computer-mediated communication. In *Computerization and Controversy: Value Conflicts and Social Choices*, pages 476–489. Academic Press, 2nd edition, 1996.

[127] Susan C. Herring and John C. Paolillo. Gender and genre variation in weblogs. *Journal of SocioLinguistics*, 10(4):439–459, 2006.

[128] Catherine Hill, Christianne Corbett, and Andresse St Rose. *Why So Few? Women in Science, Technology, Engineering, and Mathematics.* ERIC, 2010.

[129] Abram Hindle, Michael W. Godfrey, and Richard C. Holt. Release pattern discovery: A case study of database systems. In *International Conference on Software Maintenance (ICSM)*, pages 285–294. IEEE, 2007.

[130] Abram Hindle, Israel Herraiz, Emad Shihab, and Zhen Ming Jiang. Mining challenge 2010: FreeBSD, GNOME Desktop and Debian/Ubuntu. In *International Working Conference on Mining Software Repositories (MSR)*, pages 82–85. IEEE, 2010.

[131] Jorge E. Hirsch. An index to quantify an individual's scientific research output. *Proceedings of the National Academy of Sciences of the United States of America*, 102(46):16569–16572, 2005.

[132] L. Hodgkinson. Is technology masculine? Theorising the absence of women. In *International Symposium on Technology and Society.*, pages 121–126. IEEE, 2000.

[133] Myles Hollander, Douglas A. Wolfe, and Eric Chicken. *Nonparametric statistical methods.* Wiley, 1973.

[134] James Howison, Kevin Crowston, and Andrea Wiggins. Validity issues in the use of social network analysis with digital trace data. *Journal of the Association for Information Systems*, 12, 2011.

[135] Ozlem Inanc and Onur Tuncer. The effect of academic inbreeding on scientific effectiveness. *Scientometrics*, 88(3):885–898, 2011.

[136] European Union Initiative. Science: It's a girl thing! http://science-girl-thing.eu/, 2012. Accessed: 30/07/2012.

[137] ISO/IEC/IEEE. Standard 9945:2009 information technology – portable operating system interface (posix) base specifications, issue 7, 2009.

[138] Hosagrahar Visvesvaraya Jagadish. The conference reviewing crisis and a proposed solution. *SIGMOD Record*, 37(3):40–45, 2008.

[139] Thomas Jaki and Ludwig A. Hothorn. Statistical evaluation of toxicological assays: Dunnett or Williams test—take both. *Archives of Toxicology*, pages 1–10, 2013.

[140] Waqas Javed, Bryan McDonnel, and Niklas Elmqvist. Graphical perception of multiple time series. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):927–934, 2010.

[141] Thomas C. Jepsen. Just what is an ontology, anyway? *IT Professional*, 11(5):22–27, 2009.

[142] Corey Jergensen, Anita Sarma, and Patrick Wagstrom. The onion patch: migration in open source ecosystems. In *ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE)*, pages 70–80. ACM, 2011.

[143] Hsin-Yi Jiang, Tien N. Nguyen, Xiang Chen, Hojun Jaygarl, and Carl K. Chang. Incremental latent semantic indexing for automatic traceability link evolution management. In *IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 59–68. IEEE, 2008.

[144] Capers T. Jones. *Estimating software costs*, volume 3. McGraw-Hill, 1998.

[145] Capers T. Jones. *Applied Software Measurement: Global Analysis of Productivity and Quality*. McGraw-Hill, 2008.

[146] Siim Karus and Harald Gall. A study of language usage evolution in open source software. In *International Working Conference on Mining Software Repositories (MSR)*, pages 13–22. ACM, 2011.

[147] Alison Kelly. The construction of masculine science. In *Science for Girls*, pages 66–77. Open University Press, 1987.

[148] Alison Kelly. Why girls don't do science. In *Science for Girls*, pages 12–17. Open University Press, 1987.

[149] Maurice G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

[150] Rilla Khaled. It's not just whether you win or lose: Thoughts on gamification and culture. In *Gamification Workshop at CHI*. ACM, 2011.

[151] Foutse Khomh, Massimiliano Di Penta, and Yann-Gaël Guéhéneuc. An exploratory study of the impact of code smells on software change-proneness. In *Working Conference on Reverse Engineering (WCRE)*, pages 75–84. IEEE, 2009.

[152] Ralf Klamma, Manh Cuong Pham, and Yiwei Cao. You never walk alone: Recommending academic events based on social network analysis. In *International Conference on Complex Sciences*, pages 657–670, 2009.

[153] Jack P. C. Kleijnen and Willem J. H. Van Groenendaal. Measuring the quality of publications: new methodology and case study. *Inf. Process. Manage.*, 36(4):551–570, 2000.

[154] Donald E. Knuth. *The art of computer programming. Vol. 3: Sorting and searching*. Addison Wesley, 1973.

[155] Stefan Koch and Georg Schneider. Effort, co-operation and co-ordination in an open source software project: GNOME. *Information Systems Journal*, 12(1):27–42, 2002.

[156] Bruce Kogut and Udo Zander. What firms do? Coordination, identity, and learning. *Organization Science*, 7(5):502–518, 1996.

[157] Frank Konietschke. *nparcomp. Reference manual*, 2012.

[158] Frank Konietschke, Ludwig A. Hothorn, and Edgar Brunner. Rank-based multiple test procedures and simultaneous confidence intervals. *Electronic Journal of Statistics*, 6:738–759, 2012.

[159] Moshe Koppel, Shlomo Argamon, and Anat Shimoni. Automatically categorizing written texts by author gender. *Literary and Linguistic Computing*, 17(4):401–412, 2002.

[160] Nalini P. Kotamraju. Art versus code: The gendered evolution of web design skills. In *Society Online: The Internet in Context*, pages 189–200. Sage Publications, 2003.

[161] Erik Kouters. Identity matching and geographical movement of open-source software mailing list participants. Master's thesis, Eindhoven University of Technology, 2014.

[162] Erik Kouters, Bogdan Vasilescu, and Alexander Serebrenik. Who's who on GNOME mailing lists: Identity merging on a large data set. In *12th Belgian-Netherlands Software Evolution Seminar (BeNeVol)*, pages 33–34, 2013.

[163] Erik Kouters, Bogdan Vasilescu, Alexander Serebrenik, and Mark G. J. van den Brand. Who's who in GNOME: using LSA to merge software repository identities. In *International Conference on Software Maintenance (ICSM)*, pages 592–595. IEEE, 2012.

[164] Jens Krinke, Nicolas Gold, Yue Jia, and David Binkley. Cloning and copying between GNOME projects. In *International Working Conference on Mining Software Repositories (MSR)*, pages 98–101. IEEE, 2010.

[165] Sandeep Krishnamurthy. On the intrinsic and extrinsic motivation of free/libre/open source (FLOSS) developers. *Knowledge, Technology & Policy*, 18(4):17–39, 2006.

[166] Victor Kuechler, Claire Gilbertson, and Carlos Jensen. Gender differences in early free and open source software joining process. In *Open Source Systems: Long-Term Sustainability*, volume 378 of *IFIP Advances in Information and Communication Technology*, pages 78–93. Springer, 2012.

[167] Peep Küngas, Siim Karus, Svitlana Vakulenko, Marlon Dumas, Cristhian Parra, and Fabio Casati. Reverse-engineering conference rankings: what does it take to make a reputable conference? *Scientometrics*, pages 1–15, 2013.

[168] T. E. Kurtz, R. F. Link, J. W. Tukey, and D. L. Wallace. Short-cut multiple comparisons for balanced single and double classifications: Part 2. Derivations and approximations. *Biometrika*, 52(3/4):485–498, 1965.

[169] Vivian A. Lagesen. A cyberfeminist utopia?: Perceptions of gender and computer science among Malaysian women computer science students and faculty. *Science Technology Human Values*, 33(1):5–27, January 2008.

[170] Karim R. Lakhani and Eric von Hippel. How open source software works: "free" user-to-user assistance. *Research Policy*, 32(6):923–943, 2003.

[171] Thomas K. Landauer and Susan T. Dumais. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211, 1997.

[172] Catharina Landström. Queering feminist technology studies. *Feminist Theory*, 8(1):7–26, 2007.

[173] Phil Laplante, J. Rockne, P. Montuschi, T. Baldwin, M. Hinchey, L. Shafer, J. Voas, and Wenping Wang. Quality in conference publishing. *IEEE Transactions on Professional Communication*, 52(2):183–196, 2009.

[174] Samuel Lee, Nina Moisa, and Marco Weiss. Open source as a signalling device - an economic analysis. Working Paper Series: Finance and Accounting 102, Dept Finance, Goethe Univ Frankfurt am Main, 2003.

[175] Alkeline van Lenning. The body as crowbar: Transcending or stretching sex? *Feminist Theory*, 5(1):25–47, 2004.

[176] Nina Lerman, Ruth Oldenziel, and Arwen P. Mohun. *Gender and Technology: A Reader*. Gender and Technology. Johns Hopkins University Press, 2003.

[177] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

[178] Ann Light. HCI as heterodoxy: Technologies of identity and the queering of interaction with computers. *Interacting with Computers*, 23(5):430–438, 2011.

[179] Yuwei Lin. A techno-feminist view on the open source software development. In *Encyclopedia of gender and information technology*, pages 1148–1153. IGI Global, 2006.

[180] Erik Linstead and Pierre Baldi. Mining the coherence of GNOME bug reports with statistical topic models. In *International Working Conference on Mining Software Repositories (MSR)*, pages 99–102. IEEE, 2009.

[181] Todd Little. Schedule estimation and uncertainty surrounding the cone of uncertainty. *IEEE Software*, 23(3):48–54, 2006.

[182] Maria Lohan and Wendy Faulkner. Masculinities and technologies: Some introductory remarks. *Men and Masculinities*, 6(4):319–329, 2004.

[183] Luis Lopez-Fernandez, Gregorio Robles, Jesús M. González-Barahona, and Israel Herraiz. Applying social network analysis techniques to community-driven libre software projects. *International Journal of Information Technology and Web Engineering*, 1(3):27–48, 2006.

[184] Max O. Lorenz. Methods of measuring the concentration of wealth. *Journal of the American Statistical Association*, 9(70):209–219, 1905.

[185] Panagiotis Louridas, Diomidis Spinellis, and Vasileios Vlachos. Power laws in software. *ACM Transactions on Software Engineering and Methodology*, 18:2:1–2:26, 2008.

[186] Xiaoguang Lu, Hong Chen, and Anil Jain. Multimodal facial gender and ethnicity identification. In *Advances in Biometrics*, volume 3832 of *Lecture Notes in Computer Science*, pages 554–561. Springer, 2005.

[187] Bart Luijten, Joost Visser, and Andy Zaidman. Assessment of issue handling efficiency. In *International Working Conference on Mining Software Repositories (MSR)*, pages 94–97. IEEE, 2010.

[188] Mircea Lungu, Michele Lanza, Tudor Gîrba, and Romain Robbes. The small project observatory: Visualizing software ecosystems. *Science of Computer Programming*, 75:264–275, 2010.

[189] Mircea Lungu, Jacopo Malnati, and Michele Lanza. Visualizing Gnome with the small project observatory. In *International Working Conference on Mining Software Repositories (MSR)*, pages 103–106. IEEE, 2009.

[190] Jay Lyman. Getting in touch with the feminine side of open source. *Linux Today*, August 2005.

[191] Lena Mamykina, Bella Manoim, Manas Mittal, George Hripcsak, and Björn Hartmann. Design lessons from the fastest Q&A site in the west. In *ACM CHI Conference on Human Factors in Computing Systems*, pages 2857–2866. ACM, 2011.

[192] Yannis Manolopoulos. A statistic study for the adbis period 1994-2006. In *ADBIS Research Communications*, volume 215 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2006.

[193] Andrian Marcus and Jonathan I. Maletic. Recovering documentation to source code traceability links using latent semantic indexing. In *International Conference on Software Engineering (ICSE)*, pages 125–137. IEEE, 2003.

[194] Jane Margolis and Allan Fisher. *Unlocking the clubhouse: Women in computing*. The MIT Press, 2003.

[195] Jonathan Marshall. Online life and gender vagueness and impersonation. In *Encyclopedia of gender and information technology*, pages 932–937. IGI Global, 2006.

[196] Waister Silva Martins, Marcos André Gonçalves, Alberto H. F. Laender, and Gisele L. Pappa. Learning to assess the quality of scientific conferences: a case study in computer science. In *Joint Conference on Digital Library (JCDL)*, pages 193–202, 2009.

[197] Waister Silva Martins, Marcos André Gonçalves, Alberto H. F. Laender, and Nivio Ziviani. Assessing the quality of scientific conferences based on bibliographic citations. *Scientometrics*, 83(1):133–155, 2010.

[198] Tom Mens, Maëlick Claes, Philippe Grosjean, and Alexander Serebrenik. Studying evolving software ecosystems based on ecological models. In Tom Mens, Alexander Serebrenik, and Anthony Cleve, editors, *Evolving Software Systems*, pages 297–326. Springer, 2013.

[199] Tom Mens and Mathieu Goeminne. Analysing the evolution of social aspects of open source software ecosystems. In *International Workshop on Software Ecosystems*, pages 1–14. CEUR-WS, 2011.

[200] Carolyn Merchant. *The Death of Nature: Women, Ecology, and the Scientific Revolution*. ISSR library. HarperCollins, 1990.

[201] Bertrand Meyer, Christine Choppy, Jørgen Staunstrup, and Jan van Leeuwen. Viewpoint - Research evaluation for computer science. *Communications of the ACM*, 52(4):31–34, 2009.

[202] Iwona Miliszewska, Gayle Barker, Fiona Henderson, and Ewa Sztendur. The issue of gender equity in computer science—what students say. *Journal of Information Technology Education*, 5(1):107–120, 2006.

[203] Ivan Mistrík, John Grundy, Andre Van der Hoek, and Jim Whitehead. *Collaborative software engineering: challenges and prospects*. Springer, 2010.

[204] Jens Moberg, Charlotte Gooskens, John Nerbonne, and Nathan Vaillette. Conditional entropy measures intelligibility among related languages. In *Proceedings of Computational Linguistics in the Netherlands*, pages 51–66, 2007.

[205] Audris Mockus, Roy T. Fielding, and James D. Herbsleb. Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology*, 11(3):309–346, 2002.

[206] Jae Yun Moon and Lee Sproull. Essence of distributed work: The case of Linux kernel. *First Monday*, 5(11), 2000.

[207] Karine Mordal, Nicolas Anquetil, Jannik Laval, Alexander Serebrenik, Bogdan Vasilescu, and Stéphane Ducasse. Software quality metrics aggregation in industry. *Journal of Software: Evolution and Process*, 25(10):1117–1135, 2013.

[208] D. Nafus, J. Leach, and B. Krieger. FLOSSPOLS Deliverable D 16 Gender: Integrated Report of Findings. `http://www.flosspols.org/deliverables/D16HTML/FLOSSPOLS-D16-Gender_Integrated_Report_of_Findings.htm`, 2006.

[209] Dawn Nafus. 'Patches don't have gender': What is not open in open source software. *New Media & Society*, 14(4):669–683, 2012.

[210] Nachiappan Nagappan, Brendan Murphy, and Victor Basili. The influence of organizational structure on software quality: an empirical case study. In *International Conference on Software Engineering (ICSE)*, pages 521–530. ACM, 2008.

[211] Kumiyo Nakakoji, Yasuhiro Yamamoto, Yoshiyuki Nishinaka, Kouichi Kishida, and Yunwen Ye. Evolution patterns of open-source software systems and communities. In *International Workshop on Principles of Software Evolution (IWPSE)*, pages 76–85. ACM, 2002.

[212] National Science Foundation. Women, Minorities, and Persons with Disabilities in Science and Engineering. `http://www.nsf.gov/statistics/wmpd/pdf/nsf04317.pdf`, 2004. NSF 04-317.

[213] Margaret A. Neale, Gregory B. Northcraft, and Karen A. Jehn. Exploring Pandora's box; the impact of diversity and conflict on work group performance. *Performance Improvement Quarterly*, 12(1):113–126, 1999.

[214] Dave Neary and Vanessa David. The GNOME Census: Who writes GNOME? In *GNOME Users And Developers European Conference*, 2010.

[215] Sylvie Neu, Michele Lanza, Lile Hattori, and Marco D'Ambros. Telling stories about GNOME with Complicity. In *International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT)*, pages 1–8. IEEE, 2011.

[216] Donald E. Neumann. An enhanced neural network technique for software risk analysis. *IEEE Transactions on Software Engineering*, 28(9):904–912, 2002.

[217] Robert G. Newcombe. Two-sided confidence intervals for the single proportion: comparison of seven methods. *Statistics in Medicine*, 17(8):857–872, 1998.

[218] Muriel Niederle and Lise Vesterlund. Do women shy away from competition? do men compete too much? *The Quarterly Journal of Economics*, 122(3):1067–1101, 2007.

[219] Gottfried Emanuel Noether. Why Kendall tau? *Teaching Statistics*, 3(2):41–43, 1981.

[220] Andy Oram. Why do people write free documentation? Results of a survey. `http://www.onlamp.com/lpt/a/7062`, 2007.

[221] Else Ouweneel, Pascale M. Le Blanc, Wilmar B. Schaufeli, and Corine I. van Wijhe. Good morning, good day: A diary study on positive emotions, hope, and work engagement. *Human Relations*, 65(9):1129–1154, 2012.

[222] Malcolm R. Parks. Making friends in cyberspace. *Journal of Computer-Mediated Communication*, 1(4), 1996.

[223] Chris Parnin, Christoph Treude, Lars Grammel, and Margaret-Anne Storey. Crowd documentation: Exploring the coverage and the dynamics of API discussions on Stack Overflow. Technical report, Georgia Institute of Technology, 2012.

[224] G. P. Patil and C. Taillie. Diversity as a concept and its measurement. *Journal of the American Statistical Association*, 77(379):548–561, 1982.

[225] David Patterson, Lawrence Snyder, and Jeffrey Ullman. Best practices memo: evaluating computer scientists and engineers for promotion and tenure. *Computing Research News*, 11(4), 1999.

[226] Karl Pearson. Note on Regression and Inheritance in the Case of Two Parents. *Royal Society Proceedings*, 58:240–242, 1895.

[227] Mike Perrow and David Barber. Tagging of name records for genealogical data browsing. In *Joint Conference on Digital Library (JCDL)*, pages 316–325. IEEE, 2006.

[228] Manh Cuong Pham, Ralf Klamma, and Matthias Jarke. Development of computer science disciplines: a social network analysis approach. *Social Network Analysis and Mining*, 1(4):321–340, 2011.

[229] Martin Pinzger, Nachiappan Nagappan, and Brendan Murphy. Can developer-module networks predict failures? In *ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE)*, pages 2–12. ACM, 2008.

[230] Daniel Pletea, Bogdan Vasilescu, and Alexander Serebrenik. Security and emotion: Sentiment analysis of security discussions on GitHub. In *International Working Conference on Mining Software Repositories (MSR), Challenge Track*, pages 348–351. ACM, 2014.

[231] Wouter Poncin, Alexander Serebrenik, and Mark G. J. van den Brand. Process mining software repositories. In *European Conference on Software Maintenance and Reengineering (CSMR)*, pages 5–14. IEEE, 2011.

[232] Daryl Posnett, Raissa D'Souza, Premkumar Devanbu, and Vladimir Filkov. Dual ecological measures of focus in software development. In *International Conference on Software Engineering (ICSE)*, pages 452–461. IEEE, 2013.

[233] Daryl Posnett, Vladimir Filkov, and Premkumar T. Devanbu. Ecological inference in empirical software engineering. In *IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 362–371, 2011.

[234] Daryl Posnett, Eric Warburg, Premkumar T. Devanbu, and Vladimir Filkov. Mining Stack Exchange: Expertise is evident from earliest interactions. In *ASE International Conference on Social Informatics (SocialInformatics)*. ASE, 2012.

[235] Steven Postrel. Islands of shared knowledge: Specialization and mutual understanding in problem-solving teams. *Organization Science*, 13(3):303–320, 2002.

[236] Colin Potts and Lara Catledge. Collaborative conceptual design: A large software project case study. *Computer Supported Cooperative Work*, 5(4):415–445, 1996.

[237] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C/C++: The Art of Scientific Computing Code*. Cambridge University Press, 2002.

[238] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2010.

[239] Eric S. Raymond. The cathedral and the bazaar: Musings on linux and open source by accidental revolutionary revised edition. *Cambridge, UK, O'Reily*, 1999.

[240] Peter Rechenberg. Programming languages as thought models. *Structured Programming*, 11(3):105–116, 1990.

[241] Eric S. Roberts, Marina Kassianidou, and Lilly Irani. Encouraging women in computer science. *ACM SIGCSE Bulletin*, 34(2):84–88, 2002.

[242] Jeffrey A. Roberts, Il-Horn Hann, and Sandra A. Slaughter. Understanding the motivations, participation, and performance of open source software developers: A longitudinal study of the Apache projects. *Management Science*, 52(7):984–999, 2006.

[243] Lynne D. Roberts and Malcolm R. Parks. The social geography of gender-switching in virtual environments on the internet. *Information, Communication and Society*, 2(4):521–540, 1999.

[244] Gregorio Robles, Laura Arjona-Reina, Bogdan Vasilescu, Alexander Serebrenik, and Jesús M. González-Barahona. FLOSS 2013: A survey dataset about free software contributors: Challenges for curating, sharing, and combining. In *International Working Conference on Mining Software Repositories (MSR), Data Track*, pages 396–399. ACM, 2014.

[245] Gregorio Robles and Jesús M. González-Barahona. Developer identification methods for integrated data from various sources. In *International Working Conference on Mining Software Repositories (MSR)*, pages 106–110. ACM, 2005.

[246] Gregorio Robles and Jesús M. González-Barahona. Contributor turnover in libre software projects. In *Open Source Systems*, volume 203, pages 273–286. Springer, 2006.

[247] Gregorio Robles, Jesús M. González-Barahona, Daniel Izquierdo-Cortazar, and Israel Herraiz. Tools for the study of the usual data sources found in libre software projects. *International Journal of Open Source Software and Processes*, 1(1):24–45, 2009.

[248] Gregorio Robles, Jesús M. González-Barahona, and Juan Julian Merelo. Beyond source code: the importance of other artifacts in software development (a case study). *Journal of Systems and Software*, 79(9):1233–1248, 2006.

[249] Els Rommes. Gender sensitive design practices. In *Encyclopedia of gender and information technology*, pages 675–681. IGI Global, 2006.

[250] Christian Rose. Re: Handling Translations. `https://mail.gnome.org/archives/gnome-web-list/2001-August/msg00073.html`, September 2001.

[251] Christian Rose. Re: Git vs SVN (was: Can we improve things?). `https://mail.gnome.org/archives/foundation-list/2007-September/msg00050.html`, September 2007.

[252] Nigel Ross. Writing in the information age. *English Today*, 22:39–45, 6 2006.

[253] Sherif Sakr and Mohammad Alomari. A decade of database conferences: a look inside the program committees. *Scientometrics*, 91(1):173–184, 2012.

[254] G. Salton and M.J. McGill. *Introduction to modern information retrieval*. McGraw-Hill, 1986.

[255] Holger Schackmann and Horst Lichter. Evaluating process quality in GNOME based on change request data. In *International Working Conference on Mining Software Repositories (MSR)*, pages 95–98. IEEE, 2009.

[256] Herbert Schildt. *C/C++ Programmer's Reference*. McGraw-Hill, 2nd edition, 2000.

[257] Diana Schimke, Heidrun Stoeger, and Albert Ziegler. The relationship between social presence and group identification within online communities and its impact on the success of online communities. In *International Conference on Online Communities and Social Computing*, pages 160–168. Springer, 2007.

[258] Bram Schoenmakers, Niels van den Broek, Istvan Nagy, Bogdan Vasilescu, and Alexander Serebrenik. Assessing the complexity of upgrading software modules. In *Working Conference on Reverse Engineering (WCRE)*, pages 433–440. IEEE, 2013.

[259] Shalom H. Schwartz. A theory of cultural values and some implications for work. *Applied psychology*, 48(1):23–47, 1999.

[260] Carolyn B. Seaman and Victor R. Basili. Communication and organization in software development: an empirical study. *IBM Systems Journal*, 36(4):550–563, 1997.

[261] Jasjeet S. Sekhon. Multivariate and propensity score matching software with automated balance optimization: The matching package for R. *Journal of Statistical Software*, 42(7):1–52, 2011.

[262] Alexander Serebrenik and Mark G. J. van den Brand. Theil index for aggregation of software metrics values. In *International Conference on Software Maintenance (ICSM)*, pages 1–9. IEEE, 2010.

[263] Alexander Serebrenik, Bogdan Vasilescu, and Mark G. J. van den Brand. Similar tasks, different effort: Why the same amount of functionality requires different development effort? In *10th BElgian-NEtherlands software eVOLution seminar*, pages 4–5, 2011.

[264] Chirag Shah, Sanghee Oh, and Jung Sun Oh. Research agenda for social Q&A. *Library & Information Science Research*, 31(4):205–209, 2009.

[265] Sonali K. Shah. Motivation, governance, and the viability of hybrid forms in open source software development. *Management Science*, 52(7):1000–1014, 2006.

[266] N Sadat Shami, Kate Ehrlich, Geri Gay, and Jeffrey T. Hancock. Making sense of strangers' expertise from signals in digital artifacts. In *ACM CHI Conference on Human Factors in Computing Systems*, pages 69–78. ACM, 2009.

[267] Lindsay H. Shaw and Larry M. Gant. Users divided? Exploring the gender gap in internet use. *CyberPsychology & Behavior*, 5(6):517–527, 2002.

[268] David J. Sheskin. *Handbook of Parametric and Nonparametric Statistical Procedures*. Chapman & Hall, 4 edition, 2007.

[269] Bianca Shibuya and Tetsuo Tamai. Understanding the process of participating in open source communities. In *Emerging Trends in FLOSS*, pages 1–6. IEEE, 2009.

[270] Emad Shihab, Zhen Ming Jiang, and Ahmed E. Hassan. On the use of internet relay chat (IRC) meetings by developers of the GNOME GTK+ project. In *International Working Conference on Mining Software Repositories (MSR)*, pages 107–110. IEEE, 2009.

[271] Herbert A. Simon. Designing organizations for an information rich world. In *Computers, communications, and the public interest*, pages 37–72. Johns Hopkins Press, 1971.

[272] Leif Singer, Fernando Figueira Filho, Brendan Cleary, Christoph Treude, Margaret-Anne Storey, and Kurt Schneider. Mutual assessment in the social programmer ecosystem: An empirical investigation of developer profile aggregators. In *ACM Conference on Computer Supported Cooperative Work and Social Computing (CSCW)*. ACM, 2013.

[273] Vandana Singh, Michael B. Twidale, and David M. Nichols. Users of open source software - How do they get help? In *Hawaii International Conference on System Sciences (HICSS)*, pages 1–10. IEEE, 2009.

[274] Richard Snodgrass. Single- versus double-blind reviewing: an analysis of the literature. *Sigmod Record*, 35(3):8–21, 2006.

[275] Anousak Souphavanh and Theppitak Karoonboonyanan. *Free/Open Source Software: Localization*. United Nations Asia Pacific Development Information Programme, 2005.

[276] Maria Aparecida M. Souto, Mariusa Warpechowski, and José Palazzo M. Oliveira. An ontological approach for the quality assessment of computer science conferences. In *Advances in Conceptual Modeling – Foundations and Applications*, volume 4802 of *Lecture Notes in Computer Science*, pages 202–212. Springer, 2007.

[277] Sulayman K. Sowe, Ioannis Stamelos, and Lefteris Angelis. Understanding knowledge sharing activities in free/open source software projects: An empirical study. *Journal of Systems and Software*, 81(3):431–446, 2008.

[278] Charles Spearman. The proof and measurement of association between two things. *American Journal of Psychology*, 15(1):72–101, 1904.

[279] Michael Spence. Job market signaling. *The Quarterly Journal of Economics*, pages 355–374, 1973.

[280] Megan Squire. How the FLOSS research community uses email archives. *International Journal of Open Source Software and Processes*, 4(1):37–59, 2012.

[281] Richard M. Stallman. EMACS the extensible, customizable self-documenting display editor. *Sigplan Notices*, 16(6):147–156, 1981.

[282] Linda Stepulevage. Gender/Technology relations: complicating the gender binary. *Gender and Education*, 13(3):325–338, 2001.

[283] Daniel Stone. Re: [fdo] Re: On translation regressions due to freedesktop.org dependencies. `https://mail.gnome.org/archives/gnome-i18n/2004-July/msg00146.html`, July 2004.

[284] Margaret-Anne Storey. Msr 2012 keynote: The evolution of the social programmer. In *International Working Conference on Mining Software Repositories (MSR)*, pages 140–140. IEEE, 2012.

[285] Margaret-Anne Storey, Christoph Treude, Arie van Deursen, and Li-Te Cheng. The impact of social media on software engineering practices and tools. In *FSE/SDP Workshop on the Future of Software Engineering Research (FoSER)*, pages 359–364. ACM, 2010.

[286] John Suler. Do boys (and girls) just wanna have fun? Gender switching in cyberspace. In *Gender communication*, pages 149–152. Kendall/Hunt, 2004.

[287] Nicole Sullivan. Don't feed the trolls. `http://www.youtube.com/watch?v=ulNSlES1Fds`, 2012.

[288] Morris Swadesh, Joel Sherzer, and Dell Hymes. *The Origin and Diversification of Language*. Adeline Transaction, 1971.

[289] Janet Swisher. Open source user assistance: Ensuring that everybody wins. *Open Source Business Resource*, 2010.

[290] Tarja Systä, Maarit Harsu, and Kai Koskimies. Inbreeding in software engineering conferences. `http://www.cs.tut.fi/~tsysta/`. accessed November 2013.

[291] Craig Taube-Schock, Robert J. Walker, and Ian H. Witten. Can we avoid high coupling? In *European Conference on Object-Oriented Programming (ECOOP)*, pages 204–228. Springer, 2011.

[292] Paul A. Taylor. *Hackers: crime in the digital sublime*. Psychology Press, 1999.

[293] Antonio Terceiro, Luiz Romario Rios, and Christina Chavez. An empirical study on the structural complexity introduced by core and peripheral developers in free software projects. In *Brazilian Symposyum on Software Engineering*, pages 21–29. IEEE, 2010.

[294] Henri Theil. *Economics and Information Theory*. North-Holland, 1967.

[295] Henri Theil. *Principles of Econometrics*. John Wiley, 1971.

[296] Ferdian Thung, Tegawendé F. Bissyandé, David Lo, and Lingxiao Jiang. Network structure of social coding in GitHub. In *European Conference on Software Maintenance and Reengineering (CSMR)*, pages 323–326. IEEE, 2013.

[297] Eileen M. Trauth, Jeria L. Quesenberry, and Allison J. Morgan. Understanding the under representation of women in IT: Toward a theory of individual differences. In *SIGMIS Conference on Computer Personnel Research: Careers, culture, and ethics in a networked environment*, pages 114–119. ACM, 2004.

[298] Christoph Treude, Ohad Barzilay, and Margaret-Anne Storey. How do programmers ask and answer questions on the web? In *International Conference on Software Engineering (ICSE)*, pages 804–807. ACM, 2011.

[299] Christoph Treude, Fernando Figueira Filho, Brendan Cleary, and Margaret-Anne Storey. Programming in a socially networked world: the evolution of the social programmer. In *CSCW 2012 Workshop: The Future of Collaborative Software Development*, pages 1–3, 2012.

[300] Evangelos Triantaphyllou. *Multi-Criteria Decision Making Methods: A Comparative Study*. Applied Optimization. Springer, 2000.

[301] Jason T. Tsay, Laura Dabbish, and James D. Herbsleb. Social media and success in open source projects. In *ACM Conference on Computer Supported Cooperative Work (CSCW)*, pages 223–226. ACM, 2012.

[302] Jason T. Tsay, Laura Dabbish, and James D. Herbsleb. Influence of social and technical factors for evaluating contribution in GitHub. In *International Conference on Software Engineering (ICSE)*, pages 356–366. ACM, 2014.

[303] John W. Tukey. Quick and dirty methods in statistics, part II, Simple analysis for standard designs. In *American Society for Quality Control*, pages 189–197, 1951.

[304] Sherry Turkle. *The Second Self: Computers and the Human Spirit.* The MIT Press, 2005.

[305] Sherry Turkle. *Life on the Screen.* Simon & Schuster, 2011.

[306] VA. Never see any women answering questions? `http://meta.stackoverflow.com/q/34070/185480`, 2011.

[307] VA. What can Stack Overflow do to persuade female programmers to participate more? `http://meta.stackoverflow.com/q/30411/185480`, 2011.

[308] Sergi Valverde. Crossover from endogenous to exogenous activity in open-source software development. *Europhysics Letters*, 77(2):20002, 2007.

[309] Nancy A. Van House. Feminist HCI meets Facebook: Performativity and social networking sites. *Interacting with Computers*, 23(5):422–429, 2011.

[310] Moshe Y. Vardi. Conferences vs. journals in computing research. *Communications of the ACM*, 52(5):5, 2009.

[311] Rajesh Vasa, Markus Lumpe, Philip Branch, and Oscar M. Nierstrasz. Comparative analysis of evolving software systems using the Gini coefficient. In *International Conference on Software Maintenance (ICSM)*, pages 179–188. IEEE, 2009.

[312] Bogdan Vasilescu. Academic papers using Stack Overflow data. `http://meta.stackoverflow.com/q/134495`, 2012.

[313] Bogdan Vasilescu. Human aspects, gamification, and social media in collaborative software engineering. In *Companion Proceedings of the 36th International Conference on Software Engineering (ICSE)*, pages 646–649. ACM, 2014.

[314] Bogdan Vasilescu. Software developers are humans, too! In *Companion of the ACM Conference on Computer Supported Cooperative Work and Social Computing (CSCW)*, pages 97–100. ACM, 2014.

[315] Bogdan Vasilescu, Andrea Capiluppi, and Alexander Serebrenik. Gender, representation and online participation: A quantitative study of Stack Overflow. In *ASE International Conference on Social Informatics (SocialInformatics)*, pages 332–338. IEEE, 2012.

[316] Bogdan Vasilescu, Andrea Capiluppi, and Alexander Serebrenik. Gender, representation and online participation: A quantitative study. *Interacting with Computers*, pages 1–24, 2013.

[317] Bogdan Vasilescu, Vladimir Filkov, and Alexander Serebrenik. Crowdsourced knowledge catalyzes software development. In *12th Belgian-Netherlands Software Evolution Seminar (BeNeVol)*, pages 60–62, 2013.

[318] Bogdan Vasilescu, Vladimir Filkov, and Alexander Serebrenik. Stack Overflow and GitHub: Associations between software development and crowdsourced knowledge. In *International Conference on Social Computing (SocialCom)*, pages 188–195. IEEE, 2013.

[319] Bogdan Vasilescu, Alexander Serebrenik, Premkumar T. Devanbu, and Vladimir Filkov. How social Q&A sites are changing knowledge sharing in open source software communities. In *ACM Conference on Computer Supported Cooperative Work and Social Computing (CSCW)*, pages 342–354. ACM, 2014.

[320] Bogdan Vasilescu, Alexander Serebrenik, Mathieu Goeminne, and Tom Mens. On the variation and specialisation of workload – A case study of the GNOME ecosystem community. *Empirical Software Engineering*, 19(4):955–1008, 2013.

[321] Bogdan Vasilescu, Alexander Serebrenik, and Tom Mens. A historical dataset of software engineering conferences. In *International Working Conference on Mining Software Repositories (MSR), Data Track*, pages 373–376. IEEE, 2013.

[322] Bogdan Vasilescu, Alexander Serebrenik, Tom Mens, Mark G. J. van den Brand, and Ekaterina Pek. How healthy are software engineering conferences? *Science of Computer Programming*, 89, Part C:251–272, 2014.

[323] Bogdan Vasilescu, Alexander Serebrenik, and Mark G. J. van den Brand. By no means: a study on aggregating software metrics. In *International Workshop on Emerging Trends in Software Metrics (WETSoM)*, pages 23–26. ACM, 2011.

[324] Bogdan Vasilescu, Alexander Serebrenik, and Mark G. J. van den Brand. You can't control the unfamiliar: A study on the relations between aggregation techniques for software metrics. In *International Conference on Software Maintenance (ICSM)*, pages 313–322. IEEE, 2011.

[325] Bogdan Vasilescu, Alexander Serebrenik, and Mark G. J. van den Brand. The Babel of software development: Linguistic diversity in open source. In *International Conference on Social Informatics (SocInfo)*, volume 8238 of *Lecture Notes in Computer Science*, pages 391–404. Springer, 2013.

[326] Bogdan Vasilescu, Stef van Schuylenburg, Jules Wulms, Alexander Serebrenik, and Mark G. J. van den Brand. Continuous integration in a social-coding world: Empirical evidence from GitHub. In *International Conference on Software Maintenance and Evolution (ICSME), ERA Track*. IEEE, 2014.

[327] Julita Vassileva. Motivating participation in social computing applications: a user modeling perspective. *User Modeling and User-Adapted Interaction*, 22(1-2):177–201, 2012.

[328] Luis Villa. Re: GNOME Project Organogram. `https://mail.gnome.org/archives/marketing-list/2007-February/msg00027.html`, February 2007.

[329] Quang H. Vuong. Likelihood ratio tests for model selection and non-nested hypotheses. *Econometrica*, 57(2):307–333, 1989.

[330] Judy Wajcman. Feminist theories of technology. *Cambridge Journal of Economics*, 34(1):143–152, January 2010.

[331] Gang Wang, Hsinchun Chen, and Homa Atabakhsh. Automatically detecting deceptive criminal identities. *Communications of the ACM*, 47(3):70–76, 2004.

[332] Shaowei Wang, David Lo, Bogdan Vasilescu, and Alexander Serebrenik. EnTagRec: An enhanced tag recommendation system for software information sites. In *International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2014.

[333] Youcheng Wang and Daniel R. Fesenmaier. Modeling participation in an online travel community. *Journal of Travel Research*, 42(3):261–270, 2004.

[334] David A. Watt and William Findlay. *Programming language design concepts*. Wiley, 2004.

[335] Jeff Waugh. GNOME community celebrates 10 years of software freedom, innovation and industry adoption. `https://mail.gnome.org/archives/gnome-announce-list/2007-August/msg00048.html`, August 2007.

[336] Steven Weber. *The success of open source*. Harvard University Press, 2004.

[337] Gerald M. Weinberg. *The psychology of computer programming*. Van Nostrand Reinhold, 1971.

[338] Jan Martijn E. M. van der Werf, Boudewijn F. van Dongen, Cor A. J. Hurkens, and Alexander Serebrenik. Process discovery using integer linear programming. *Fundamenta Informaticae*, 94(3-4):387–412, 2009.

[339] Michele A. Whitecraft and Wendy M. Williams. Why aren't more women in computer science? In *Making Software: What Really Works, and Why We Believe It*, pages 221–238. O'Reilly Media, Inc., 2010.

[340] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

[341] Edwin B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927.

[342] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in software engineering: an introduction*. Kluwer, 2000.

[343] Richard T. A. Wood and Mark D. Griffiths. Why Swedish people play online poker and factors that can increase or decrease trust in poker web sites: A qualitative investigation. *Journal of Gambling Issues*, 21:80–97, 2008.

[344] WordPress. WordPress mailing lists. `http://codex.wordpress.org/Mailing_Lists`, 2012.

[345] Ye Wu, Changsong Zhou, Jinghua Xiao, Jürgen Kurths, and Hans Joachim Schellnhuber. Evidence for a bimodal distribution in human communication. *Proceedings of the National Academy of Sciences*, 107(44):18803–18808, 2010.

[346] Qi Xuan, Mohammad Gharehyazie, Premkumar T Devanbu, and Vladimir Filkov. Measuring the effect of social communications on individual working rhythms: A case study of open source software. In *ASE International Conference on Social Informatics (SocialInformatics)*, pages 78–85. IEEE, 2012.

[347] Su Yan and Dongwon Lee. Toward alternative measures for ranking venues: a case of database research community. In *Joint Conference on Digital Library (JCDL)*, pages 235–244, 2007.

[348] Liguo Yu and Srini Ramaswamy. Mining CVS repositories to understand open-source project developer roles. In *International Working Conference on Mining Software Repositories (MSR)*, page 8. IEEE, 2007.

[349] Amotz Zahavi. Mate selection—a selection for a handicap. *Journal of Theoretical Biology*, 53(1):205–214, 1975.

[350] Amotz Zahavi, Avishag Zahavi, Naama Zahavi-Ely, Melvin Patrick Ely, and Amir Balaban. *The handicap principle: a missing piece of Darwin's puzzle*. Oxford University Press, 1997.

[351] Andy Zaidman, Bart Van Rompaey, Arie van Deursen, and Serge Demeyer. Studying the co-evolution of production and test code in open source and industrial developer test processes through repository mining. *Empirical Software Engineering*, 16(3):325–364, 2011.

[352] Achim Zeileis. *ineq: Measuring Inequality, Concentration, and Poverty*. R Foundation for Statistical Computing, 2009.

[353] Ziming Zhuang, Ergin Elmacioglu, Dongwon Lee, and C. Lee Giles. Measuring conference quality by mining program committee characteristics. In *Joint Conference on Digital Library (JCDL)*, pages 225–234, 2007.

[354] Li Zhuolun, Ke-Wei Huang, and Huseyin Cavusoglu. Can we gamify voluntary contributions to online Q&A communities? Quantifying the impact of badges on user engagement. In *International Conference on Web Information System Engineering (WISE)*, 2012.

[355] Donald W. Zimmerman and Bruno D. Zumbo. Parametric alternatives to the Student t test under violation of normality and homogeneity of variance. *Perceptual and motor skills*, 74(3(1)):835–844, 1992.

[356] Stanley Zionts. MCDM—if not a roman numeral, then what? *Interfaces*, 9(4):94–101, 1979.

[357] Justin Zobel and Philip Dart. Phonetic string matching: Lessons from information retrieval. In *International ACM SIGIR Conference on Research and Development on Information Retrieval (SIGIR)*, pages 166–172. ACM, 1996.

[358] Liesbet van Zoonen. Feminist theory and information technology. *Media, Culture and Society*, 14(1):12–35, 1992.

Appendix A

Activity type rules

Rules used to assign each file to an activity type. The rule for each activity type is defined by a regular expression. If the expression matches the file's path, the activity type is associated to the file. The rules are assessed in sequence. Among the rules matching the file, the last one is used to determine the activity type. We do not allow for multiple classification, as this poses problems with the definition of some metrics and the statistical analysis of some results.

Activity type acronym	Regular expressions			
Unknown unknown	.*			
Documentation doc	.*\.((s|x|g|p|(gt))?)htm(l?)		.*/translators	.*/contributors
	.*/doc(-?)book(s?)/.*	.*\.page	.*/info	.*\.1
	.*\.zabw	.*/potfiles	.*/install	.*\.wml
	.*\.chm	.*\.ods	.*/copyright	.*/version
	.*\.css	.*\.vcard(~?)	.*/plan	.*/feature(s?)
	.*\.txt((\.bak)?)	.*/credits	.*/notes	.*/licence
	.*\.txt((\.old)?)	.*\.man	.*/howto	.*/license
	.*\.rtf	.*\.ics	.*/faq	.*/maintainers
	.*\.tex	.*/documenters	.*copying	.*/copying
	.*\.sgml	.*\.gnumeric	.*/copying.*	.*/committers
	.*\.pdf	.*\.vcf	.*/doc(s?)/.*	.*/thanks
	.*\.xsd	.*\.schemas	.*/help(s?)/.*	.*/authors
	.*\.texi	.*\.doc	.*/bugs	.*\.docx
Image img	.*\.png	.*\.ppm	.*\.icns	.*\.eps
	.*\.pgm	.*\.jpg	.*\.chm	.*\.xbm
	.*\.jpeg	.*\.bmp	.*\.chm	.*\.vdx
	.*\.gif	.*\.sgv(z?)	.*\.nsh	.*\.ico
	.*\.xcf			
Localization l10n	.*/.potfiles\.in(~?)	.*\.i18ns(~?)	.*\.pot(~?)	/po/.*
	/strings.properties	.*\.mo(~?)	.*\.linguas	.*\.wxl
	.*\.gmo(~?)	.*\.resx(~?)	.*/locale(s?)/.*	.*\.po(~?)
	.*\.charset(~?)			
User interface ui	.*\.glade(\d?)((\.bak)?)(~?)		.*\.desktop	.*\.xul(~?)
	.*\.gladed(\d?)((\.bak)?)(~?)		.*\.ui	.*\.xpm
	.*\.gladep(\d?)((\.bak)?)(~?)		.*\.theme	

Activity type acronym	Regular expressions			
Multimedia media	.*\.mp3	.*\.mp4	.*/media(s?)/.*	.*\.pfm
	.*\.mpv	.*\.mml	.*/font(s?)/.*	.*\.gnect
	.*\.ogg	.*\.ogv	.*/icon(s?)/.*	.*\.shape
	.*\.wav	.*\.au	.*\.otf(~?)	.*\.gnl
	.*\.mov	.*\.avi	.*\.sfd(~?)	.*\.pgn
	.*\.mid	.*\.xspf	.*\.ttf(~?)	.*\.cdf
	.*\.m4f	.*\.ps	.*\.afm	.*\.bse
	.*\.pls	.*\.omf	.*\.pfb	.*\.cur
Coding code	.*\.dmg(~?)	.*\.swg(~?)	.*\.so(~?)	.*\.i(~?)
	.*\.o(~?)	.*\.exe(~?)	.*\.oafinfo(~?)	.*\.pyd(~?)
	.*\.awk(~?)	.*\.scm(~?)	.*\.glsl(~?)	.*\.patch(~?)
	.*\.c((\.swp)?)(~?)	.*/script(s?)/.*	.*\.jar(~?)	.*/src/.*
	.*\.m((\.swp)?)(~?)	.*\.cs(~?)	.*\.idl(~?)	.*\.s(~?)
	.*\.r((\.swp)?)(~?)	.*\.cxx(~?)	.*\.pyc(~?)	.*\.asm(x?)(~?)
	.*\.py((\.swp)?)(~?)	.*\.y((\.swp)?)(~?)	.*\.gi((\.swp)?)(~?)	
	.*\.t((\.swp)?)(~?)	.*\.dll(~?)	.*\.h\.template((\.swp)?)(~?)	
	.*\.js((\.swp)?)(~?)	.*\.rb((\.swp)?)(~?)	.*\.c\.template((\.swp)?)(~?)	
	.*\.hg((\.swp)?)(~?)	.*\.pm((\.swp)?)(~?)	.*\.php((\.swp)?)(\d?)(~?)	
	.*\.cc((\.swp)?)(~?)	.*\.sh((\.swp)?)(~?)	.*\.php((\.swp)?)(\d?)(~?)	
	.*\.el((\.swp)?)(~?)	.*\.hh((\.swp)?)(~?)	.*\.h((pp)?)((\.swp)?)(~?)	
	.*\.xs((\.swp)?)(~?)	.*\.pl((\.swp)?)(~?)	.*\.h\.tmpl((\.swp)?)(~?)	
	.*\.mm((\.swp)?)(~?)	.*\.idl((\.swp)?)(~?)	.*\.h.win32((\.swp)?)(~?)	
	.*\.xpt((\.swp)?)(~?)	.*\.ccg((\.swp)?)(~?)	.*\.c\.tmpl((\.swp)?)(~?)	
	.*\.snk((\.swp)?)(~?)	.*\.inc((\.swp)?)(~?)	.*\.asp(x?)((\.swp)?)(~?)	
	.*\.cpp((\.swp)?)(~?)	.*\.gob((\.swp)?)(~?)	.*\.vapi((\.swp)?)(~?)	
	.*\.giv((\.swp)?)(~?)	.*\.dtd((\.swp)?)(~?)	.*\.gidl((\.swp)?)(~?)	
	.*\.giv((\.swp)?)(~?)	.*\.ada((\.swp)?)(~?)	.*\.defs((\.swp)?)(~?)	
	.*\.tcl((\.swp)?)(~?)	.*\.vbs((\.swp)?)(~?)	.*\.java((\.swp)?)(~?)	
	.*\.nib((\.swp)?)(~?)	.*\.sed((\.swp)?)(~?)	.*\.vala((\.swp)?)(~?)	
Meta meta	.*\.svn(.*)	.*\.git(.*)	.*\.doap	.*\.mdp
	.*\.cvs(.*)	.*\.bzr(.*)	.*\.mds	.*\.vbg
	.*\.sln			
Configuration config	.*\.conf	.*\.cfg	.*\.anjuta	.*\.dsw
	.*\.gnorba	.*\.project	.*\.pgp(~?)	.*\.ini
	.*\.prefs	.*\.vsprops	.*\.gpg(~?)	.*\.config
	.*\.vmrc	.*\.csproj	.*\.gpg\.pub(~?)	.*\.xml
	.*\.cproj	.*\.cbproj	.*\.pgp\.pub(~?)	.*\.dsp
	.*\.emacs	.*\.groupproj	.*\.xcconfig	.*\.plist
	.*\.pbxproj	.*anjuta\.session	.*/.*setting(s?).*/.*\.jp	
	.*/.*config(s?).*/.*\.jp			
Building build	.*\.cmake	.*/install-sh	.*/build/.*	.*\.ezt
	.*\.cbp	.*\.pch	.*/pkg-info	.*\.wxilib
	.*\.m4(~?)	.*makefile.*	.*\.prj	.*\.plo
	.*\.mk	.*\.make	.*\.deps	.*\.wxiproj
	.*\.am(~?)	.*\.mp4	.*\.builder	.*\.lo
	.*\.target	.*\.iss	.*\.nsi	.*\.wxi
	.*/configure((\..+)?)	.*\.wxs	.*/mkbundle\..+	.*\.in
	.*/autogen\.((.+\.)?)sh		.*\.wpj	
	.*\.vc(x?)proj(i?)n((\.filters((in)?))?)		.*\.vcproj((\.filters((in)?))?)	
Development documentation devdoc	.*readme.*	.*/changelog.*	.*\.dia(~?)	.*\.ical
	.*/changes	.*/status	.*/fixme	.*\.doxi
	.*/todo.*	.*/hacking.*	.*/news.*	.*/roadmap
	.*\.rst	.*/devel(-?)doc(s?)/.*		
Database db	.*\.sql	.*\.sqlite	.*\.mdb	.*\.yaml
	.*\.sdb	.*\.dat	.*\.yaml	.*\.json
	.*\.db	.*/berkeleydb.*/.*		
Testing test	.*\.test(s?)/.*	.*/.*test\..*	.*/test.*\..*	
Library lib	.*/library/.*	.*/libraries/.*		

Summary

Social Aspects of Collaboration in Online Software Communities

Software engineering is inherently a collaborative venture, involving many stakeholders that coordinate their efforts to produce large software systems. In distributed (online) settings, endemic to open-source software (OSS) development, such collaborations span geographies and cultures. Contributors with different skill sets, personalities, cultural backgrounds, or geographic locations *self-organise* in online software communities and *voluntarily contribute* to a collaborative effort, such as developing and evolving software systems, offering user support, or sharing knowledge. Myriads of online software communities exist. Among the most visible are those around the Linux operating system and its various distributions, the Gnome desktop environment, the Apache or Mozilla software foundations or, more recently, the GitHub repository hosting platform, or the Stack Exchange network of question and answer sites (*e.g.*, Stack Overflow for programming-related questions).

Different communities operate by different rules and offer different incentives to contribute. More traditional communities such as Gnome or Apache rely mostly on the intrinsic motivations of developers. In contrast, younger communities offer social media and gamification features (*e.g.*, contributors to Stack Overflow are rewarded for their activity with reputation points and badges) as well as increased visibility to peers (*e.g.*, the activity and achievements of GitHub and Stack Overflow contributors are aggregated and displayed on public profile pages). Furthermore, online software communities are often interdependent. For example, a Gnome contributor may engage simultaneously in different communities part of the Gnome ecosystem, where she may take on different roles or instead choose to specialise in similar tasks. Similarly, a GitHub contributor may participate simultaneously in Stack Overflow, where she may seek help from peers or instead share her knowledge and expertise to help educate others.

Starting from the recent realisation of the empirical software engineering research community that *social* aspects are at least as important for the success of distributed collaborations as *technical* ones, this dissertation is an attempt to understand collaboration in representative online software communities from a social perspective. To this end, we mines and analyses a wealth of publicly available trace data using state-of-the-art statistical techniques, from various standpoints outlined below. The work in this dissertation sits at the intersection of two research communities, computer-supported cooperative work (CSCW) and software engineering (SE). Consequently, it offers contributions to both SE and CSCW, ranging from methods and tools to mine and analyse large amounts of

(social) trace data, to practical implications for software maintainability, software team management, knowledge management, or community design.

First, we propose individual-level measures of *workload*, *involvement*, and *specialisation of labour* in online software development communities, and report on a case study of the GNOME ecosystem community. Additionally, we propose a community-level measure of *skill diversity* with respect to a certain technical skill, such as knowledge of a given programming language. Social interactions between contributors to software development communities and their degree of participation have been reported repeatedly to influence software quality and complexity. Similarly, skill heterogeneity in a software community is important for the community's survival and performance.

Second, we analyse the effects of simultaneously contributing to multiple communities on individual *working rhythms*. We focus on developers contributing source code to GITHUB repositories while seeking and sharing knowledge on STACK OVERFLOW. Despite the popularity of STACK OVERFLOW, its role in the work cycle of OSS developers is yet to be understood. On the one hand, participation in it has the potential to increase the knowledge of individual developers thus improving and speeding up the development process. On the other hand, participation in STACK OVERFLOW may interrupt the regular working rhythm of a developer, hence also possibly slow down the development process. We show that active GITHUB contributors typically engage in STACK OVERFLOW as experts, asking few questions and providing many answers for others. Moreover, despite the interruptions incurred, the STACK OVERFLOW activity rate correlates with the code changing rate on GITHUB.

Third, we chart the changes in behaviour of contributors to online knowledge sharing communities as they *migrate into gamified environments*. To this end, we assemble a joint data set for R (a widely-used tool for data analysis) by integrating data from mailing lists and STACK EXCHANGE sites, having activities of individual contributors linked across the two resources and also over time. We find that user support activities show a strong shift away from mailing lists (historically the hub for development and user support activities in online software communities) and towards STACK EXCHANGE. Moreover, knowledge providers contributing to both communities provide faster answers on STACK EXCHANGE than on the mailing list, and their total output increases after the transition.

Fourth, we revisit the gamified STACK OVERFLOW environment from the perspectives of *gender representation and participation patterns*, in comparison to traditional information sharing venues such as mailing lists. In addition to encouraging competitiveness through its gamification features, anecdotal evidence around STACK OVERFLOW suggests that it is an unfriendly community that promotes one-upmanship, or fosters flame-wars and the down-voting of individuals. We find that while women are under-represented in all studied communities, they show different participation patterns on STACK OVERFLOW, where they disengage sooner than men. However, relative to the duration of their engagement in the community, women on STACK OVERFLOW are at least as active as men.

Fifth, we address the recurring mining software repositories challenge of inconsistent identity data. To this end, we analyse the robustness of two representative existing identity merging algorithms with respect to different types of noise typical of software repositories, and we propose a new identity merging algorithm inspired by Latent Semantic Analysis, a popular information retrieval technique expected to perform well in presence of noise. Using data extracted from GNOME Git repositories, we evaluate the performance of our algorithm empirically by means of cross-validation, and show an improvement over existing approaches in terms of precision and recall on worst-case input data.

Last, we provide an example of the applicability of methods developed or refined as part of this work beyond online software communities. Specifically, we propose a metrics suite to study the health of software engineering conferences and conference communities, measuring such attributes as stability, openness to new authors, or introversion. Using this metrics suite, we assess the health of 11 established software engineering conferences over a period of more than 10 years. In general, we find that software engineering conferences are healthy, with some notable differences depending on the chosen health metric, or a conference's wide or narrow scope. Beyond demonstrating the generalisability of our techniques, this latter study has implications for conference steering committees or program committee chairs wishing to assess their selection process, or prospective authors trying to decide in which conferences to publish.

Curriculum Vitae

Personal Information

Name: Bogdan Nicolae Vasilescu
Date of birth: August 5, 1985
Place of birth: Ploiesti, Romania

Education

MSc. Computer Science and Engineering (cum laude) *2009–2011*
Eindhoven University of Technology
Eindhoven, The Netherlands

Dipl. Eng. Systems and Computer Science (cum laude) *2004–2009*
Petroleum-Gas University
Ploiesti, Romania

Professional Experience

PhD candidate *2011–2014*
Eindhoven University of Technology
Eindhoven, The Netherlands

IPA Dissertation Series

Titles in the IPA Dissertation Series since 2008

W. Pieters. *La Volonté Machinale: Understanding the Electronic Voting Controversy.* Faculty of Science, Mathematics and Computer Science, RU. 2008-01

A.L. de Groot. *Practical Automaton Proofs in PVS.* Faculty of Science, Mathematics and Computer Science, RU. 2008-02

M. Bruntink. *Renovation of Idiomatic Crosscutting Concerns in Embedded Systems.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2008-03

A.M. Marin. *An Integrated System to Manage Crosscutting Concerns in Source Code.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2008-04

N.C.W.M. Braspenning. *Model-based Integration and Testing of High-tech Multi-disciplinary Systems.* Faculty of Mechanical Engineering, TU/e. 2008-05

M. Bravenboer. *Exercises in Free Syntax: Syntax Definition, Parsing, and Assimilation of Language Conglomerates.* Faculty of Science, UU. 2008-06

M. Torabi Dashti. *Keeping Fairness Alive: Design and Formal Verification of Optimistic Fair Exchange Protocols.* Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2008-07

I.S.M. de Jong. *Integration and Test Strategies for Complex Manufacturing Machines.* Faculty of Mechanical Engineering, TU/e. 2008-08

I. Hasuo. *Tracing Anonymity with Coalgebras.* Faculty of Science, Mathematics and Computer Science, RU. 2008-09

L.G.W.A. Cleophas. *Tree Algorithms: Two Taxonomies and a Toolkit.* Faculty of Mathematics and Computer Science, TU/e. 2008-10

I.S. Zapreev. *Model Checking Markov Chains: Techniques and Tools.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-11

M. Farshi. *A Theoretical and Experimental Study of Geometric Networks.* Faculty of Mathematics and Computer Science, TU/e. 2008-12

G. Gulesir. *Evolvable Behavior Specifications Using Context-Sensitive Wildcards.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-13

F.D. Garcia. *Formal and Computational Cryptography: Protocols, Hashes and Commitments.* Faculty of Science, Mathematics and Computer Science, RU. 2008-14

P. E. A. Dürr. *Resource-based Verification for Robust Composition of Aspects.*

Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-15

E.M. Bortnik. *Formal Methods in Support of SMC Design.* Faculty of Mechanical Engineering, TU/e. 2008-16

R.H. Mak. *Design and Performance Analysis of Data-Independent Stream Processing Systems.* Faculty of Mathematics and Computer Science, TU/e. 2008-17

M. van der Horst. *Scalable Block Processing Algorithms.* Faculty of Mathematics and Computer Science, TU/e. 2008-18

C.M. Gray. *Algorithms for Fat Objects: Decompositions and Applications.* Faculty of Mathematics and Computer Science, TU/e. 2008-19

J.R. Calamé. *Testing Reactive Systems with Data - Enumerative Methods and Constraint Solving.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-20

E. Mumford. *Drawing Graphs for Cartographic Applications.* Faculty of Mathematics and Computer Science, TU/e. 2008-21

E.H. de Graaf. *Mining Semi-structured Data, Theoretical and Experimental Aspects of Pattern Evaluation.* Faculty of Mathematics and Natural Sciences, UL. 2008-22

R. Brijder. *Models of Natural Computation: Gene Assembly and Membrane Systems.* Faculty of Mathematics and Natural Sciences, UL. 2008-23

A. Koprowski. *Termination of Rewriting and Its Certification.* Faculty of Mathematics and Computer Science, TU/e. 2008-24

U. Khadim. *Process Algebras for Hybrid Systems: Comparison and Development.* Faculty of Mathematics and Computer Science, TU/e. 2008-25

J. Markovski. *Real and Stochastic Time in Process Algebras for Performance Evaluation.* Faculty of Mathematics and Computer Science, TU/e. 2008-26

H. Kastenberg. *Graph-Based Software Specification and Verification.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-27

I.R. Buhan. *Cryptographic Keys from Noisy Data Theory and Applications.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-28

R.S. Marin-Perianu. *Wireless Sensor Networks in Motion: Clustering Algorithms for Service Discovery and Provisioning.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-29

M.H.G. Verhoef. *Modeling and Validating Distributed Embedded Real-Time Control Systems.* Faculty of Science, Mathematics and Computer Science, RU. 2009-01

M. de Mol. *Reasoning about Functional Programs: Sparkle, a proof assistant for Clean.* Faculty of Science, Mathematics and Computer Science, RU. 2009-02

M. Lormans. *Managing Requirements Evolution.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2009-03

M.P.W.J. van Osch. *Automated Model-based Testing of Hybrid Systems.* Faculty of Mathematics and Computer Science, TU/e. 2009-04

H. Sozer. *Architecting Fault-Tolerant Software Systems.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-05

M.J. van Weerdenburg. *Efficient Rewriting Techniques.* Faculty of Mathematics and Computer Science, TU/e. 2009-06

H.H. Hansen. *Coalgebraic Modelling: Applications in Automata Theory and Modal Logic.* Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2009-07

A. Mesbah. *Analysis and Testing of Ajax-based Single-page Web Applications.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2009-08

A.L. Rodriguez Yakushev. *Towards Getting Generic Programming Ready for Prime Time.* Faculty of Science, UU. 2009-9

K.R. Olmos Joffré. *Strategies for Context Sensitive Program Transformation.* Faculty of Science, UU. 2009-10

J.A.G.M. van den Berg. *Reasoning about Java programs in PVS using JML.* Faculty of Science, Mathematics and Computer Science, RU. 2009-11

M.G. Khatib. *MEMS-Based Storage Devices. Integration in Energy-Constrained Mobile Systems.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-12

S.G.M. Cornelissen. *Evaluating Dynamic Analysis Techniques for Program Comprehension.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2009-13

D. Bolzoni. *Revisiting Anomaly-based Network Intrusion Detection Systems.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-14

H.L. Jonker. *Security Matters: Privacy in Voting and Fairness in Digital Exchange.* Faculty of Mathematics and Computer Science, TU/e. 2009-15

M.R. Czenko. *TuLiP - Reshaping Trust Management.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-16

T. Chen. *Clocks, Dice and Processes.* Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2009-17

C. Kaliszyk. *Correctness and Availability: Building Computer Algebra on top of Proof Assistants and making Proof Assistants available over the Web.* Faculty of Science, Mathematics and Computer Science, RU. 2009-18

R.S.S. O'Connor. *Incompleteness & Completeness: Formalizing Logic and Analysis in Type Theory.* Faculty of Science, Mathematics and Computer Science, RU. 2009-19

B. Ploeger. *Improved Verification Methods for Concurrent Systems.* Faculty of Mathematics and Computer Science, TU/e. 2009-20

T. Han. *Diagnosis, Synthesis and Analysis of Probabilistic Models.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-21

R. Li. *Mixed-Integer Evolution Strategies for Parameter Optimization and Their Applications to Medical Image Analysis.* Faculty of Mathematics and Natural Sciences, UL. 2009-22

J.H.P. Kwisthout. *The Computational Complexity of Probabilistic Networks.* Faculty of Science, UU. 2009-23

T.K. Cocx. *Algorithmic Tools for Data-Oriented Law Enforcement.* Faculty of Mathematics and Natural Sciences, UL. 2009-24

A.I. Baars. *Embedded Compilers.* Faculty of Science, UU. 2009-25

M.A.C. Dekker. *Flexible Access Control for Dynamic Collaborative Environments.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-26

J.F.J. Laros. *Metrics and Visualisation for Crime Analysis and Genomics.* Faculty of Mathematics and Natural Sciences, UL. 2009-27

C.J. Boogerd. *Focusing Automatic Code Inspections.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2010-01

M.R. Neuhäußer. *Model Checking Nondeterministic and Randomly Timed Systems.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2010-02

J. Endrullis. *Termination and Productivity.* Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2010-03

T. Staijen. *Graph-Based Specification and Verification for Aspect-Oriented Languages.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2010-04

Y. Wang. *Epistemic Modelling and Protocol Dynamics.* Faculty of Science, UvA. 2010-05

J.K. Berendsen. *Abstraction, Prices and Probability in Model Checking Timed Automata.* Faculty of Science, Mathematics and Computer Science, RU. 2010-06

A. Nugroho. *The Effects of UML Modeling on the Quality of Software.* Faculty of Mathematics and Natural Sciences, UL. 2010-07

A. Silva. *Kleene Coalgebra.* Faculty of Science, Mathematics and Computer Science, RU. 2010-08

J.S. de Bruin. *Service-Oriented Discovery of Knowledge - Foundations, Implementations and Applications.* Faculty of Mathematics and Natural Sciences, UL. 2010-09

D. Costa. *Formal Models for Component Connectors.* Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2010-10

M.M. Jaghoori. *Time at Your Service: Schedulability Analysis of Real-Time and Distributed Services.* Faculty of Mathematics and Natural Sciences, UL. 2010-11

R. Bakhshi. *Gossiping Models: Formal Analysis of Epidemic Protocols.* Faculty of Sciences, Department of Computer Science, VUA. 2011-01

B.J. Arnoldus. *An Illumination of the Template Enigma: Software Code Generation with Templates.* Faculty of Mathematics and Computer Science, TU/e. 2011-02

E. Zambon. *Towards Optimal IT Availability Planning: Methods and Tools.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2011-03

L. Astefanoaei. *An Executable Theory of Multi-Agent Systems Refinement.* Faculty of Mathematics and Natural Sciences, UL. 2011-04

J. Proença. *Synchronous coordination of distributed components.* Faculty of Mathematics and Natural Sciences, UL. 2011-05

A. Moralı. *IT Architecture-Based Confidentiality Risk Assessment in Networks of Organizations.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2011-06

M. van der Bijl. *On changing models in Model-Based Testing.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2011-07

C. Krause. *Reconfigurable Component Connectors.* Faculty of Mathematics and Natural Sciences, UL. 2011-08

M.E. Andrés. *Quantitative Analysis of Information Leakage in Probabilistic and Nondeterministic Systems.* Faculty of Science, Mathematics and Computer Science, RU. 2011-09

M. Atif. *Formal Modeling and Verification of Distributed Failure Detectors.* Faculty of Mathematics and Computer Science, TU/e. 2011-10

P.J.A. van Tilburg. *From Computability to Executability – A process-theoretic view on automata theory.* Faculty of Mathematics and Computer Science, TU/e. 2011-11

Z. Protic. *Configuration management for models: Generic methods for model comparison and model co-evolution.* Faculty of Mathematics and Computer Science, TU/e. 2011-12

S. Georgievska. *Probability and Hiding in Concurrent Processes.* Faculty of Mathematics and Computer Science, TU/e. 2011-13

S. Malakuti. *Event Composition Model: Achieving Naturalness in Runtime Enforcement.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2011-14

M. Raffelsieper. *Cell Libraries and Verification.* Faculty of Mathematics and Computer Science, TU/e. 2011-15

C.P. Tsirogiannis. *Analysis of Flow and Visibility on Triangulated Terrains.* Faculty of Mathematics and Computer Science, TU/e. 2011-16

Y.-J. Moon. *Stochastic Models for Quality of Service of Component Connectors.* Faculty of Mathematics and Natural Sciences, UL. 2011-17

R. Middelkoop. *Capturing and Exploiting Abstract Views of States in OO Verification.* Faculty of Mathematics and Computer Science, TU/e. 2011-18

M.F. van Amstel. *Assessing and Improving the Quality of Model Transformations.* Faculty of Mathematics and Computer Science, TU/e. 2011-19

A.N. Tamalet. *Towards Correct Programs in Practice.* Faculty of Science, Mathematics and Computer Science, RU. 2011-20

H.J.S. Basten. *Ambiguity Detection for Programming Language Grammars.* Faculty of Science, UvA. 2011-21

M. Izadi. *Model Checking of Component Connectors.* Faculty of Mathematics and Natural Sciences, UL. 2011-22

L.C.L. Kats. *Building Blocks for Language Workbenches.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2011-23

S. Kemper. *Modelling and Analysis of Real-Time Coordination Patterns.* Faculty of Mathematics and Natural Sciences, UL. 2011-24

J. Wang. *Spiking Neural P Systems.* Faculty of Mathematics and Natural Sciences, UL. 2011-25

A. Khosravi. *Optimal Geometric Data Structures.* Faculty of Mathematics and Computer Science, TU/e. 2012-01

A. Middelkoop. *Inference of Program Properties with Attribute Grammars, Revisited.* Faculty of Science, UU. 2012-02

Z. Hemel. *Methods and Techniques for the Design and Implementation of Domain-Specific Languages.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2012-03

T. Dimkov. *Alignment of Organizational Security Policies: Theory and Practice.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2012-04

S. Sedghi. *Towards Provably Secure Efficiently Searchable Encryption.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2012-05

F. Heidarian Dehkordi. *Studies on Verification of Wireless Sensor Networks and Abstraction Learning for System Inference.* Faculty of Science, Mathematics and Computer Science, RU. 2012-06

K. Verbeek. *Algorithms for Cartographic Visualization.* Faculty of Mathematics and Computer Science, TU/e. 2012-07

D.E. Nadales Agut. *A Compositional Interchange Format for Hybrid Systems: Design and Implementation.* Faculty of Mechanical Engineering, TU/e. 2012-08

H. Rahmani. *Analysis of Protein-Protein Interaction Networks by Means of*

Annotated Graph Mining Algorithms. Faculty of Mathematics and Natural Sciences, UL. 2012-09

S.D. Vermolen. *Software Language Evolution*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2012-10

L.J.P. Engelen. *From Napkin Sketches to Reliable Software*. Faculty of Mathematics and Computer Science, TU/e. 2012-11

F.P.M. Stappers. *Bridging Formal Models – An Engineering Perspective*. Faculty of Mathematics and Computer Science, TU/e. 2012-12

W. Heijstek. *Software Architecture Design in Global and Model-Centric Software Development*. Faculty of Mathematics and Natural Sciences, UL. 2012-13

C. Kop. *Higher Order Termination*. Faculty of Sciences, Department of Computer Science, VUA. 2012-14

A. Osaiweran. *Formal Development of Control Software in the Medical Systems Domain*. Faculty of Mathematics and Computer Science, TU/e. 2012-15

W. Kuijper. *Compositional Synthesis of Safety Controllers*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2012-16

H. Beohar. *Refinement of Communication and States in Models of Embedded Systems*. Faculty of Mathematics and Computer Science, TU/e. 2013-01

G. Igna. *Performance Analysis of Real-Time Task Systems using Timed Automata*. Faculty of Science, Mathematics and Computer Science, RU. 2013-02

E. Zambon. *Abstract Graph Transformation – Theory and Practice*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2013-03

B. Lijnse. *TOP to the Rescue – Task-Oriented Programming for Incident Response Applications*. Faculty of Science, Mathematics and Computer Science, RU. 2013-04

G.T. de Koning Gans. *Outsmarting Smart Cards*. Faculty of Science, Mathematics and Computer Science, RU. 2013-05

M.S. Greiler. *Test Suite Comprehension for Modular and Dynamic Systems*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2013-06

L.E. Mamane. *Interactive mathematical documents: creation and presentation*. Faculty of Science, Mathematics and Computer Science, RU. 2013-07

M.M.H.P. van den Heuvel. *Composition and synchronization of real-time components upon one processor*. Faculty of Mathematics and Computer Science, TU/e. 2013-08

J. Businge. *Co-evolution of the Eclipse Framework and its Third-party Plug-ins*. Faculty of Mathematics and Computer Science, TU/e. 2013-09

S. van der Burg. *A Reference Architecture for Distributed Software Deployment*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2013-10

J.J.A. Keiren. *Advanced Reduction Techniques for Model Checking*. Faculty of Mathematics and Computer Science, TU/e. 2013-11

D.H.P. Gerrits. *Pushing and Pulling: Computing push plans for disk-shaped robots, and dynamic labelings for moving points*. Faculty of Mathematics and Computer Science, TU/e. 2013-12

M. Timmer. *Efficient Modelling, Generation and Analysis of Markov Automata*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2013-13

M.J.M. Roeloffzen. *Kinetic Data Structures in the Black-Box Model.* Faculty of Mathematics and Computer Science, TU/e. 2013-14

L. Lensink. *Applying Formal Methods in Software Development.* Faculty of Science, Mathematics and Computer Science, RU. 2013-15

C. Tankink. *Documentation and Formal Mathematics — Web Technology meets Proof Assistants.* Faculty of Science, Mathematics and Computer Science, RU. 2013-16

C. de Gouw. *Combining Monitoring with Run-time Assertion Checking.* Faculty of Mathematics and Natural Sciences, UL. 2013-17

J. van den Bos. *Gathering Evidence: Model-Driven Software Engineering in Automated Digital Forensics.* Faculty of Science, UvA. 2014-01

D. Hadziosmanovic. *The Process Matters: Cyber Security in Industrial Control Systems.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2014-02

A.J.P. Jeckmans. *Cryptographically-Enhanced Privacy for Recommender Systems.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2014-03

C.-P. Bezemer. *Performance Optimization of Multi-Tenant Software Systems.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2014-04

T.M. Ngo. *Qualitative and Quantitative Information Flow Analysis for Multi-threaded Programs.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2014-05

A.W. Laarman. *Scalable Multi-Core Model Checking.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2014-06

J. Winter. *Coalgebraic Characterizations of Automata-Theoretic Classes.* Faculty of Science, Mathematics and Computer Science, RU. 2014-07

W. Meulemans. *Similarity Measures and Algorithms for Cartographic Schematization.* Faculty of Mathematics and Computer Science, TU/e. 2014-08

A.F.E. Belinfante. *JTorX: Exploring Model-Based Testing.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2014-09

A.P. van der Meer. *Domain Specific Languages and their Type Systems.* Faculty of Mathematics and Computer Science, TU/e. 2014-10

B.N. Vasilescu. *Social Aspects of Collaboration in Online Software Communities.* Faculty of Mathematics and Computer Science, TU/e. 2014-11