Application Domains in the Research Papers at BENEVOL: A Retrospective

Andrea Capiluppi
Dept of Computer Science
Brunel University London, United Kingdom

Nemitari Ajienka
Dept of Computer Science
Edge Hill University, United Kingdom

Bilyaminu Auwal Romo
Dept of Engineering and Digital Technologies
Coventry University, United Kingdom

Abstract

Research on empirical software engineering has increasingly used the data that is made available in online repositories, specifically Free/Libre/Open Source Software projects (FLOSS). The latest trends for researchers is to gather "as much data as possible" to (i) prevent bias in the representation of a small sample, (ii) work with a sample as close as the population itself, and (iii) showcase the performance of existing or new tools in treating vast amount of data.

The effects of harvesting enormous amounts of data have been only marginally considered so far: data could be corrupted; repositories could be forked; and developer identities could be duplicated. In this paper we posit that there is a fundamental flaw in harvesting large amounts of data, and when generalising the conclusions: the application domain, or context, of the analysed systems must be the primary factor for the cluster sampling of FLOSS projects.

This paper presents two contributions: first, we analyse a collection of 100 BENEVOL papers that appeared showing whether (and how much) FLOSS data has been harvested, and how many times the authors flagged an issue in their different application domains. Second, we discuss the implications of using 'application domain' as the clustering factor in the sampling of FLOSS data, and the generalisations within and outside the clusters.

Index terms— FLOSS, application domains, BENEVOL papers

1 Introduction

The use of open, available data has been a welcomed accelerator in the software engineering research field. Data on the processes and products available via an Open Source approach has led to an increasingly large number of workshops, conferences, papers and research attempts to describe the phenomenon. Researchers gathered initially in 2001 around the Open Source Software (OSS) workshops, held annually in co-location with the ICSE series of conferences. Before the OSS workshop spawned into the OSS conference in 2005, the BENEVOL community started to group together researchers from the Software Evolution domain. Its initial focus was '*(...) to bring researchers to identify and discuss important principles, problems, techniques and results related to software evolution research and practice*'[1].

While the goal of a few BENEVOL papers has been to achieve the generality of the results [1], the domain, context and uniqueness of a software system have not been considered very often by empirical software engineering research. As in the example reported in [2], the extensive study of all JSON parsers available would

[1] `https://smartcare.be/events/benevol-04-workshop`

In: D. Di Nucci, C. De Roover (eds.): Proceedings of the 18th Belgium-Netherlands Software Evolution Workshop, Brussels, Belgium, 28-11-2019, published at http://ceur-ws.org

find similarities between them or common patterns. That type of study would focus on one particular language (JSON), one specific domain (parsers) and inevitably draw limited conclusions. On the other hand, considering the "parsers" domain (but without focusing on one single language) would show the common characteristics of developing that type of systems, and irrespective of their language.

The underlying vision of this paper is to open a proper debate on the importance of context for any software system, and the uniqueness of its application domain. This position paper stems from the work of several prominent researchers who called the community to 'go deeper, not wider' (Michael Godfrey at MSR 2017) and 'minding the mine, mining the mind' [3]. We posit that past empirical investigations using FLOSS systems have been mostly blind to these aspects (i.e., context and domain), establishing similarities between vastly different systems if they shared a common pattern in one measured attribute. Using an extreme example, one could establish a similarity between the coupling of a 'hammock' and a 'bridge' due to the fact that both are held at the sides.

The purpose of this paper is to share some findings about a selection of papers discussed during the last few years of BENEVOL workshops. The focus is specifically based on BENEVOL papers that have used FLOSS data. The context of our analysis is the diversity of FLOSS projects under study, and how that was reflected by researchers in their findings. Some 100 papers are analysed in terms of whether FLOSS projects are used, how many, and whether considerations of application domains have been used to inform the sampling of FLOSS projects, or the validity of the conclusions. We assume that domains are relevant as a fundamental construct for any empirical software engineering research [4].

2 Related Work

The vast literature on FLOSS systems of the last 10 years has been possible also due to a series of guidelines on how to perform quantitative, empirical analysis on FLOSS processes and products. When SourceForge[2] was considered as the *de-facto* FLOSS forge, a well received research paper shared more than one insight on the most common mistakes to avoid when mining data and results from the projects hosted there [5]. Among other more technical issues of mining this specific forge, this paper actively warned against an inaccurate 'screening' of projects into samples: reducing a population to, say, 'FLOSS projects with more than 7 developers' would inevitably reduce the variables for the analysis, but the 'number of developers' variable cannot be used as a dependent or independent variable for any model or analysis.

The acknowledgement of GitHub as the newly established central focus for FLOSS development generated a similar requirement, in terms of shared guidelines to avoid common mining mistakes [6]. Differently from [5], the 2014 paper mostly focused on the technical aspects of GitHub, and how the collected metrics could skew the results, due to the inner workings of the Git toolset, and the different approach to FLOSS development observed on GitHub (forking, non-software development, inactivity of projects). Neither [5] or [6] warned about the variability of FLOSS projects, the importance of their context, or the uniqueness of their domains.

Outside of the FLOSS literature, the diversity and context of software systems have received some attention in the past [4, 7]. The phrase "large scale" has been frequently used in empirical software engineering research to denote the magnitude of the analyzed case study or studied software sample. Notwithstanding, Nagappan *et al.* argue that analyzing a high number of projects is not always necessary [2]. But what is even more important is the selection of the projects studied.

Interesting patterns valuable to researchers and practitioners are often identified in domain-based analysis of software projects. Results from one domain might not be applicable in another. As such, it is important for results to be representative.

Software categorization or domain clustering has gained importance over the years. For example, the knowledge of software trends in a particular domain can assist developers in the search for domain-specific reusable components [8]. Tian *et al.* [9] proposed a technique based on Latent Dirichlet Allocation for automatic software categorization in open-source software repositories.

According to Haefliger *et al.* [10], "domain analyses, documentation, and quality standards enhance the ability to reuse software components". However, our survey of past BENEVOL papers that have analyzed OSS projects demonstrates that software domains have not been considered in most of the past software engineering studies.

3 A Survey of BENEVOL Papers: 2012 to 2018

In order to show how FLOSS data has been used and analysed by the BENEVOL community, we report here an investigation of the research papers appeared in the last 5 years of the BENEVOL event. An overall 101 papers have been considered in this study: we share the raw data in the spreadsheet at `https://tinyurl.`

[2] `https://sourceforge.net/`

com/y69wkadr.

Each paper was read by one of the co-authors, and summarised along the following points:

- *Use of FLOSS systems* (yes/no): at first we checked whether FLOSS projects are used in the paper at all. This served as an indicator of the pervasiveness of FLOSS projects in the literature produced by BENEVOL papers.
- *Number of FLOSS systems used*: in second instance, we trawled through the paper, annotating where the authors mentioned how many FLOSS systems were used. In the case of full papers, the abstract, introduction, methodology and conclusion were read for that purpose.
- *Analysis of application domains*: thirdly, we considered the methodology, results and conclusion of each paper, along with the threats to validity, looking for considerations of application domains. We checked if the authors considered this attribute in the sampling of FLOSS projects, whether they limited their results against this axis, or whether it was considered a specific threat to validity. This attribute was coded as either {*yes* — *no*}.

The contributions to the 2016 edition of BENEVOL are not available online, so they had to be excluded from our analysis. The spreadsheet with the categorisation of the papers has been made available for inspection under the following link: `https://tinyurl.com/y69wkadr`.

3.1 BENEVOL use of FLOSS Systems

In this section we provide the first point of our analysis: '*how many BENEVOL papers have used FLOSS systems in their analyses?*'. As visible in the two plots of Figure 1, researchers (and accepted BENEVOL papers) have steadily used FLOSS systems for their papers. The first plot shows the absolute numbers of accepted BENEVOL submissions that use one or more FLOSS projects.

The bottom plot of Figure 1 shows the ratio of FLOSS and non-FLOSS papers in the BENEVOL sample of papers. It is getting increasingly more common to use one or more commercial software systems, or a combination of FLOSS and non-FLOSS projects.

3.2 Number of FLOSS systems used in BENEVOL

In this section we report on the number of FLOSS systems evaluated by BENEVOL papers. For this purpose, we analysed the methodology description, or the

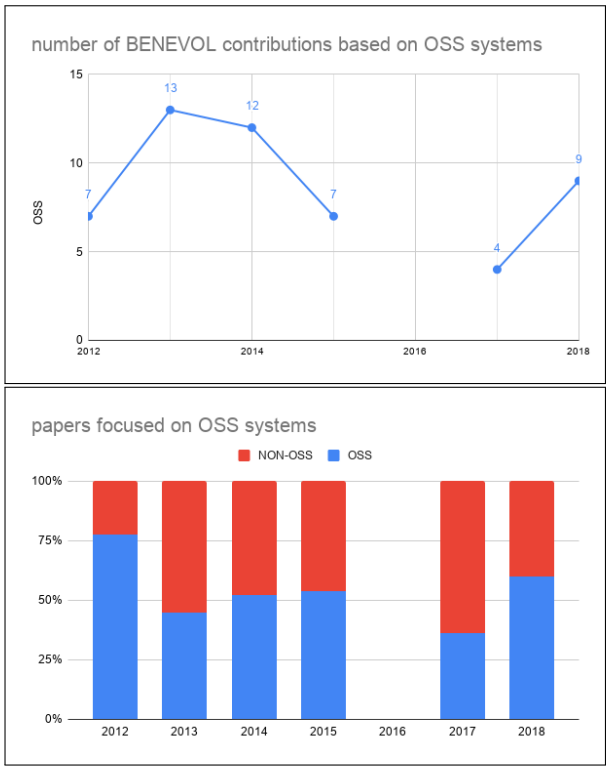

Figure 1: Papers using FLOSS (above) and use of FLOSS and non-FLOSS projects (below) in the BENEVOL (between 2012 and 2018)

empirical approach, of each paper to determine how many FLOSS systems were reported in the study. Figure 2 displays the cumulative number of FLOSS systems used in BENEVOL papers, per year. The median number of systems has increased from one analysed OSS system in 2012 to 1,127 systems in 2018.

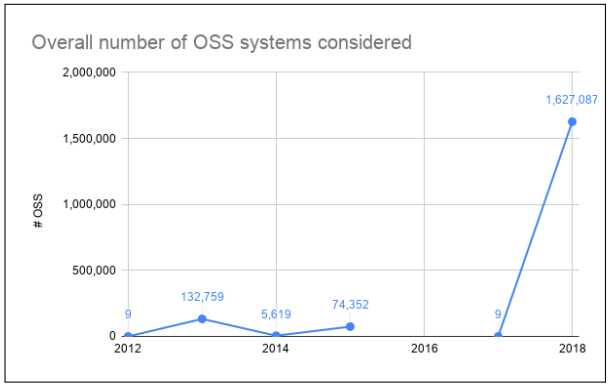

Figure 2: Cumulative number of FLOSS projects per year

The exponential number of FLOSS systems being used by BENEVOL papers has been accelerated by many factors: (i) availability of open forges (FreshMeat, SourceForge, Savannah, Apache FSF, GitHub and many others); (ii) common, shared toolsets to per-

form the analyses; (iii) guidelines on how to effectively use forges.

Below we give a summary of findings to assess the trends observed in the number of FLOSS systems analysed by the BENEVOL papers.

Growth of sample sizes

The trend that we observed throughout the subsequent years of the BENEVOL contributions is, fundamentally, summarised as 'the more the better'. Authors have started to include larger and larger FLOSS samples to their papers. We can assume that this pattern has been followed in order to achieve the generality of a paper's findings. At the last edition of available BENEVOL contributions (BENEVOL 2018), over one million FLOSS systems were considered for investigation, jointly by the accepted papers.

Uncertainty on sample sizes

Several BENEVOL papers use ecosystems [11], or umbrella projects [12], as their cases studies, whereas other papers either take a subset of those super-projects, or explicitly declaring the number of subprojects (e.g., Scala [13] or Python [14] projects) that they analysed. This means that our final figures are mostly *lower bounds* of the actual number of FLOSS systems being used by the BENEVOL community.

Ecosystems vs time of analysis

Several BENEVOL papers have used umbrella projects (for example, Gnome). In most cases we considered them as single FLOSS systems: depending on the time of the analysis, these larger projects can contain a variable number of sub-projects. This makes it difficult to define the status of the super-project, in terms of number of its sub-projects, as well as their domains. This also makes it difficult to replicate those studies, as well as their results and conclusions.

Sampling and Pruning

Throughout the editions of BENEVOL, sampling of a few systems (see for instance [15] or [16]) has given way to whole-forge analyses. Also, there seems to be a general view that 'pruning' a sample is a good idea for removing outliers, or for promoting quality. This has an effect on the sample studied, and the representativeness of the population as a whole.

3.3 Application domains and FLOSS projects

The third analysis was based on the application domains of the systems considered in the empirical study. For all the papers (not only for those using FLOSS projects), we tried to establish whether the authors considered the results, findings or discussion as constrained by the type of system (e.g., its domain). This included checking how the threats to external validity (if any) addressed limited the conclusion to the domain(s) under investigation.

We grouped the papers into two categories (and plotted them accordingly per year):

1. papers that *directly* considered application domains as drivers in the variability of the results (stack "YES" in Figure 3);
2. papers that didn't considered application domains as drivers (stack "NO" in Figure 3).

The results of this analysis are shown in Figure 3: a ratio (%) is used to separate the papers in the two categories. It is clear from the visualisation that BENEVOL papers do not generally acknowledge the variability of results as driven by the domains of the systems involved. Earlier papers (especially from the 2012 batch) have a good cover of domains in the evaluation of the results, but this is not reflected in the later editions of BENEVOL.

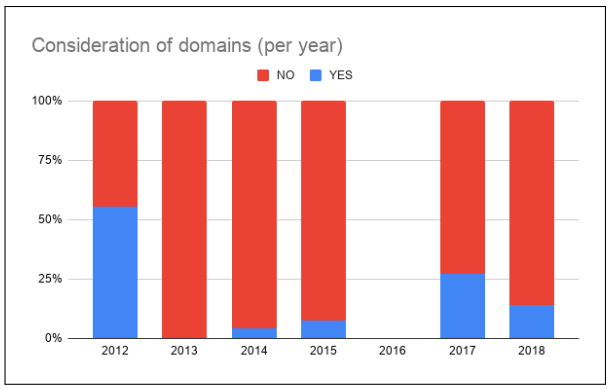

Figure 3: Application domains in papers using FLOSS projects

As visible in the Figure, the majority of findings on FLOSS, as reported by BENEVOL papers, do not mention application domains. In some cases, researchers have acknowledged the variability of the results [17, 18], and hinted that other factors could play a role in such variability. We considered as a "limited" acknowledgment of the relevance of the application domain when authors mentioned the diversity of the systems under study.

4 BENEVOL: FLOSS and Domains

The birdseye view on the type of BENEVOL contributions (Sections 3.1, 3.2 and 3.3 above) reveals some interesting trends when dealing with FLOSS projects. Below we discuss in more detail whether FLOSS papers were analysed ("YES" or "NO"), and whether

domains were considered in the analysis ("YES" or "NO").

4.1 FLOSS: YES, Domains: YES

So far in the BENEVOL series, few papers explicitly addressed the importance of domains when analysing systems, or when discussing findings. An interesting perspective is given in [19], since it considers a very specific type of systems, the 'cross-system packages'. These systems are likely to show similar characteristics since they are supposed to act as vectors to an from the overarching system.

By drawing on the importance of the application domains in this paper [20], the authors signify the importance of domain analysis when creating a theoretical and practical framework that supports the development and the evolution of adaptive data-intensive software systems for ubiquitous environments in their study. Thus, they focus on data and in particular on the problem of finding the most suitable portion of data that have to be provided by the application in the of context of 'self-adaptive system'.

Likewise in the 11^{th} edition of BENEVOL (2012), [21] examined the impact and role of social media on software development. The authors argued that "social media is poised to bring about a paradigm shift in software engineering research" particularly in OSS community.

In the 2014 edition, only one BENEVOL study focusing on OSS projects implicitly highlighted the need to investigate projects from various domains [22]. The authors studied an OSS project called *DrJava* and implicitly mentioned domains but did not investigate multiple projects clustered into several domains. According to the authors, "we chose an IDE since they contain elements of multiple domains. The IDE project was taken from the Qualitas Corpus and it consists of 3000 revisions since 2000 and the system grew from 30K SLOC in 2003 to 200K SLOC in 2013.

We concluded that application domains are not well represented or studied in the papers that use FLOSS data.

4.2 FLOSS: YES, Domains: NO

The vast majority of BENEVOL contributions, based on FLOSS systems, do not consider domains as one of the factors to take in consideration. An interesting example of this approach is given in [1], where the authors pose that '... *(to) gather as much as possible should be the aim of empirical software engineering*'.

More in general, the approach of researchers is to focus on specific languages or source code models (see for instance the paper in [23], focused on all available meta-models from GitHub), hence representing convenience sampling. For example in the study on control flow, Landman *et al.,* [24] focused on the Sourcerer Corpus which contains 18K (13K non empty) Java projects. In an empirical analysis of the maintainability of CRAN packages, Claes *et al.,* [25] presented early results on analysing the dependencies of the CRAN R packages repository.

We concluded that most of the papers studied from the BENEVOL series do not consider the application domains as an important factor for software analysis or evolution.

4.3 FLOSS: NO, Domains: YES

A few of the papers that we analysed are not based on FLOSS systems, but more in general on commercial, or in-house software. In a few cases, we observed that the authors actually considered the limitations of their case studies to the one domain that was investigated.

As a few of such examples, we noted a paper based on a banking system [26]; and one focused on the specific features of home-automation system [27]. Both these papers clearly acknowledged the limitations given by the chosen application domains that their systems are based on. In other cases, the authors specifically focused on one domain (for example, GIS systems [28], or the larger business domain [29]).

In general, the BENEVOL papers using non-OSS software as their case studies do not use the domains to aggregate results. Nonetheless a few BENEVOL contributions have shown a clear pathway into not generalising the findings to all domains.

5 Conclusion

This paper analysed how open source software has been used by the BENEVOL contributions between 2012 and 2018. We showed the increasing number of BENEVOL contributions that used FOSS projects for their analyses.

Although the majority of contributions do not acnowledge the importance of domains when discussing the findings, there is an increasing number of papers that limit the results, or the data sampling, to specific domains. We believe that one of the major challenges for empirical software engineering is to better understand the role of domains, especially in the evolution of software systems. We propose for papers that empirically analyse software systems to acknowledge such challenge in a 'threat to domain validity'.

References

[1] Antoine Pietri and Stefano Zacchiroli. Towards universal software evolution analysis. In *BENEVOL*, pages 6–10, 2018.

[2] Meiyappan Nagappan, Thomas Zimmermann, and Christian Bird. Diversity in software engineering research. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, pages 466–476. ACM, 2013.

[3] A. J. Ko. Mining the mind, minding the mine: grand challenges in comprehension and mining. In Andy Zaidman, Yasutaka Kamei, and Emily Hill, editors, *Proceedings of the 15th International Conference on Mining Software Repositories, MSR 2018, Gothenburg, Sweden, May 28-29, 2018*, page 118. ACM, 2018.

[4] 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.

[5] James Howison and Kevin Crowston. The perils and pitfalls of mining sourceforge. In *Proceedings of the International Workshop on Mining Software Repositories (MSR 2004*. Citeseer, 2004.

[6] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M German, and Daniela Damian. The promises and perils of mining github. In *Proceedings of the 11th working conference on mining software repositories*, pages 92–101. ACM, 2014.

[7] Carmine Vassallo, Sebastiano Panichella, Fabio Palomba, Sebastian Proksch, Andy Zaidman, and Harald C Gall. Context is king: The developer perspective on the usage of static analysis tools. In *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 38–49. IEEE, 2018.

[8] Yunwen Ye and Gerhard Fischer. Reuse-conducive development environments. *Automated Software Engineering*, 12(2):199–235, 2005.

[9] Kai Tian, Meghan Revelle, and Denys Poshyvanyk. Using latent dirichlet allocation for automatic categorization of software. In *6th IEEE International Working Conference on Mining Software Repositories, 2009. MSR'09.*, pages 163–166. IEEE, 2009.

[10] Stefan Haefliger, Georg Von Krogh, and Sebastian Spaeth. Code reuse in open source software. *Management Science*, 54(1):180–193, 2008.

[11] Tom Mens, Bram Adams, and Josianne Marsan. Towards an interdisciplinary, socio-technical analysis of software ecosystem health. *arXiv preprint arXiv:1711.04532*, 2017.

[12] Maëlick Claes. Applying biological evolution to software ecosystems a case study with gnome.

[13] Yunior Pacheco, Jonas De Bleser, Tim Molderez, Dario Di Nucci, Wolfgang De Meuter, and Coen De Roover. Mining extension point patterns in scala. In *BENEVOL*, pages 16–20, 2018.

[14] José Javier Merchante and Gregorio Robles. From python to pythonic: Searching for python idioms in github.

[15] Ward Muylaert and Coen De Roover. Untangling source code changes using program slicing. In *BENEVOL*, pages 36–38, 2017.

[16] Jie Tan, Mircea Lungu, and Paris Avgeriou. Towards studying the evolution of technical debt in the python projects from the apache software ecosystem. In *BENEVOL*, pages 43–45, 2018.

[17] Zeeger Lubsen, Andy Zaidman, and Martin Pinzger. Using association rules to study the co-evolution of production & test code. In *Mining Software Repositories, 2009. MSR'09. 6th IEEE International Working Conference on*, pages 151–154. IEEE, 2009.

[18] Christian Rodríguez-Bustos and Jairo Aponte. How distributed version control systems impact open source software projects. In *Mining Software Repositories (MSR), 2012 9th IEEE Working Conference on*, pages 36–39. IEEE, 2012.

[19] Eleni Constantinou, Alexandre Decan, and Tom Mens. Breaking the borders: an investigation of cross-ecosystem software packages. *arXiv preprint arXiv:1812.04868*, 2018.

[20] Marco Mori and Anthony Cleve. A framework to support the development and evolution of self-adaptive data-intensive systems. In *11th edition of the BElgian-NEtherlands software eVOLution symposium (BENEVOL 2012)*, 01 2012.

[21] Maëlick Claes. Applying biological evolution to software ecosystems a case study with gnome. In *11th edition of the BElgian-NEtherlands software eVOLution symposium (BENEVOL 2012)*, 01 2012.

[22] Davy Landman, Alexander Serebrenik, and Jurgen Vinju. The relationship between cc and sloc: a preliminary analysis on its evolution. In *Benevol 2014 (Seminar on Software Evolution in Belgium and the Netherlands, Amsterdam, The Netherlands, November 27-28, 2014)*, pages 29–30. Centrum voor Wiskunde en Informatica, 2014.

[23] Önder Babur, Loek Cleophas, and Mark van den Brand. Metamodel clone detection with samos. *BENEVOL*, 2018.

[24] Davy Landman, Alexander Serebrenik, and Jurgen Vinju. Control flow in the wild a first look at 13k java projects. *BENEVOL 2013*, page 35, 2013.

[25] Maëlick Claes, Tom Mens, and Philippe Grosjean. Towards an empirical analysis of the maintainability of cran packages. *BENEVOL 2013*, page 42.

[26] Elvan Kula, Ayushi Rastogi, Hennie Huijgens, and Arie van Deursen. Characterizing rapid releases in a large banking company: A case study. In *BENEVOL*, pages 56–60, 2018.

[27] Tim Molderez, Coen De Roover, and Wolfgang De Meuter. Towards a domain-specific language for automated network management. In *BENEVOL*, pages 39–43, 2017.

[28] Cosmin Tomozei, Iulian Furdu, and Simona-Elena Vârlan. Gis sdks dynamics echoed by social requirements transformations. In *BENEVOL*, pages 22–25, 2017.

[29] Gururaj Maddodi and Slinger Jansen. Responsive software architecture patterns for workload variations: A case-study in a cqrs-based enterprise application. In *BENEVOL*, page 30, 2017.