Empir Software Eng
DOI 10.1007/s10664-013-9244-1

On the variation and specialisation of workload—A case study of the GNOME ecosystem community

Bogdan Vasilescu · Alexander Serebrenik · Mathieu Goeminne · Tom Mens

Abstract 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

Communicated by: Margaret-Anne Storey

B. Vasilescu · A. Serebrenik
MDSE, Eindhoven University of Technology, PO Box 513, 5600 MB, Eindhoven, The Netherlands

B. Vasilescu
e-mail: b.n.vasilescu@tue.nl

A. Serebrenik
e-mail: a.serebrenik@tue.nl

M. Goeminne · T. Mens (✉)
COMPLEXYS Research Institute, Université de Mons, Place du Parc 20, 7000 Mons, Belgium
e-mail: tom.mens@umons.ac.be

M. Goeminne
e-mail: mathieu.goeminne@umons.ac.be

Published online: 16 February 2013

contributors tend to be more involved in the coding activity in a limited number of projects.

Keywords Open source · Software ecosystem · Metrics · Developer community · Case study

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 (Weber 2004; Bonaccorsi et al. 2006).

In this paper, 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 (Bettenburg and Hassan 2010; Terceiro et al. 2010). 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 et al. (2006) and Hindle et al. (2007), one can distinguish different activity types such as coding, development documentation, building, testing, and so on. Both German (2003) 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." Waugh (2007)
>
> "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." Stone (2004)

[1] www.gnome.org

This is why we use the GNOME ecosystem as a case study in this paper. 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 article is structured as follows. Section 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 3 and 4 report on the statistical evaluation carried out for each research goal, and discuss the results. Section 5 presents the threats to validity, Section 6 reviews related work, Section 7 discusses future work, and Section 8 concludes.

2 Methodology

2.1 Research Goals

Following Lungu et al. (2010) 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, 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 2.5. As explained in the introduction, and as observed in an earlier exploratory study (Mens and Goeminne 2011), 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 3 will study the first research goal, and Section 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 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., `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 2.1 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 (Robles et al. 2009). 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 commiter 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,

[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.

[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

Table 1 Variation of Git project characteristics across 1316 GNOME projects

	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

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

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 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 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 Fig. 1 visualise the distribution of these results.

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

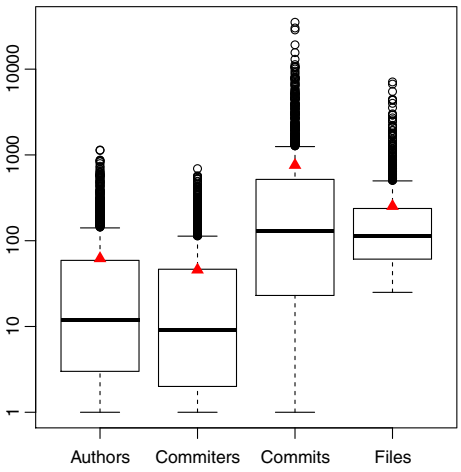


Fig. 1 Boxplots showing the variation of Git characteristics from Table 1 (log *y*-axis). *Red triangles* show the mean value

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 (Robles and González-Barahona 2005; Christen 2006; Bird et al. 2006; Goeminne and Mens 2011a; Kouters et al. 2012; Iqbal and Hausenblas 2012). 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 (Robles and González-Barahona 2005)). 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 (Goeminne and Mens 2011b). 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 Fig. 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 (Levenshtein 1966)), phonetic encoding (e.g., soundex (Knuth 1973)), or a combination of both (e.g., editex (Zobel and Dart 1996)). Christen (2006) provides an overview of such similarity measures. Although numerous name similarity measures exist and have available implementations, e.g., as part of Febrl (Christen et al. 2004), computing such measures is often computationally expensive. Moreover, there is no single best technique available. Therefore, we applied the Bertillonage approach proposed by Davies et al. (2011) 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 (2006) we required a limited subset (to ensure fast computation), well-balanced in terms of complemen-

[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 (Christen et al. 2004).

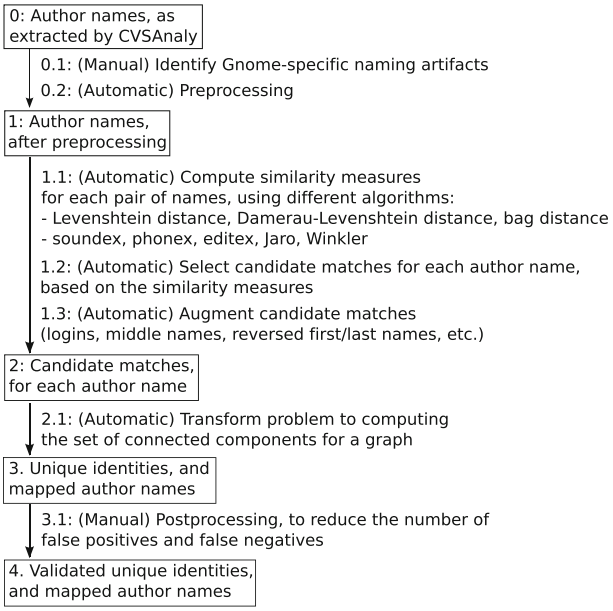


Fig. 2 Identity matching steps

tary 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 (2006), 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 6,982 different author names across all considered GNOME projects. After name matching, only 5,155 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 paper, whenever we use the term *author*, we refer to the unique identities obtained as a result of the identity matching process.

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. (2006) and Hindle et al. (2007), who distinguish between different activity types based on the file names and extensions. Hindle et al. (2007) distinguished between four types of files: source, test, build and documentation. Robles et al. (2006) proposed 8 different activity types. We expand upon the classification by Robles et al. (2006) 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 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, 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

Table 2 Excerpt of the rules (t, e) used to identify the activity types from the file paths and file names

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`

The regular expressions follow the traditional POSIX Basic Regular Expression syntax (ISO/IEC/IEEE 2009). 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

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. (2011) 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 article.

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 3 and 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 2.4), 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 2.4.

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

$$APTW(p, a, t) = \text{number of touches to files of activity type } t \text{ by author } a \text{ for project } p \text{ over its entire history.} \quad (1)$$

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:

$$APTI(p,a,t) = \begin{cases} 1, \text{ if } APTW(p,a,t) > 0; \\ 0, \text{ otherwise.} \end{cases} \quad (2)$$

Using these two basic metrics, we can derive higher-level aggregate metrics. Figure 3 presents the workload metrics that are derived from $APTW$, while Fig. 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 3 and 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 why we first compute the maximum over all types, and then compute the sum over all authors.

Tables 3 and 4 and Figs. 3 and 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 (Gini 1921). We have used it to define 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 (Gini 1921)

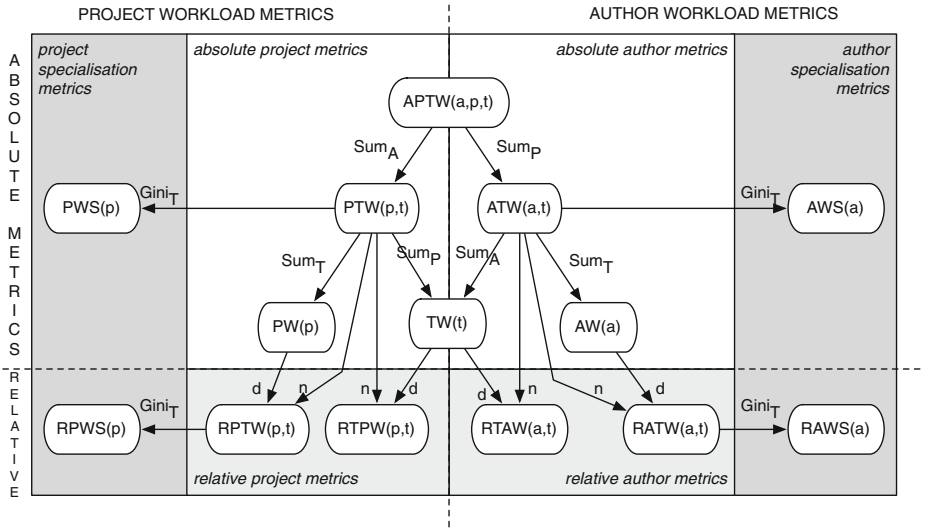


Fig. 3 Workload metrics. The following naming convention is adopted for the metric acronyms: **A** = Author; **P** = Project; **T** = activity Type; **W** = Workload; **R** = Relative; **S** = Specialisation. Relative metrics are defined as a fraction $\frac{n}{d}$ and represent a percentage (i.e., a value between 0 and 1). $Gini_T$ denotes the application of the *Gini* inequality index (Gini 1921) over all activity types. Similarly, Sum_T, Sum_A and Sum_P aggregate values through summation

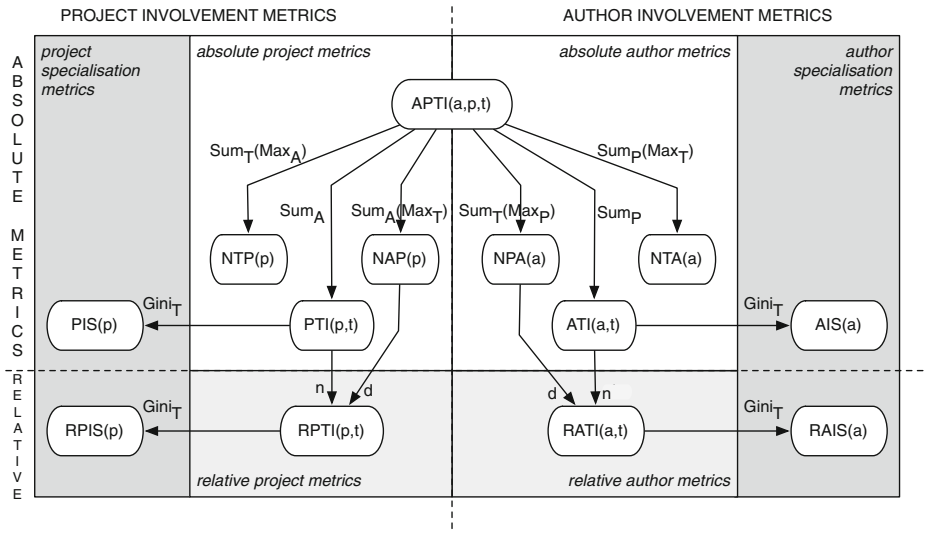


Fig. 4 Involvement metrics. The same naming convention is followed as in Fig. 3, except that we now use **I** for *involvement* and **N** for *number of*

is one of the many inequality indices commonly applied in econometrics to study inequality of income or welfare distributions (Cowell 2000; Cowell and Jenkins 1995). As opposed to traditional aggregation techniques such as mean or median, inequality indices provide reliable results for highly-skewed distributions. Similarly

Table 3 Definitions of $APTW$-based project-level workload metrics

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	$\frac{PTW}{TW(t)}$
$PWS(p)$	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	$Gini_{t_k \in T}(RPTW(p, t_k))$

(The author-level workload metrics are defined similarly.)

Table 4 Definitions of $APTI$-based project-level involvement metrics

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	$Gini_{t_k \in T}(RPTI(p, t_k))$

(The author-level involvement metrics are defined similarly.)

to such traditional aggregation techniques, inequality indices do not require complex application procedures. The Gini index is defined based on the Lorenz curve (Lorenz 1905), and ranges between 0 and $1 - \frac{1}{n}$ (Allison 1978), 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 (Cowell 2000; Theil 1967; Serebrenik and van den Brand 2010; Mordal et al. 2012), but Gini has been shown to convey the same information as the other applicable inequality indices (Vasilescu et al. 2011a, b).

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. (2012)) or specialisation of projects towards a particular author $Gini_{p \in P}(\sum_{t \in T} APTW(p, a, t))$.

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 (R Development Core Team 2010), including packages such as `ineq` to calculate the Gini index (Zeileis 2009), `Matching` to perform the bootstrapped Kolmogorov-Smirnov test (Sekhon 2011), `agricolae` to determine the Kendall

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

correlation coefficient (de Mendiburu 2010), and `nparcomp` to compute relative contrast effects when comparing two distributions (Konietschke 2012).

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 1895) are sensitive only to a linear relation between two variables. Rank coefficients (Kendall 1938; Spearman 1904) 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 (Noether 1981). We account for ties as described by Press et al. (2002). 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 $\bar{R}^2$ (Theil 1971, 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* $x^{-\alpha}$ (Louridas et al. 2008) or are *log-normal* (Baxter et al. 2006; Little 2006), in this paper 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 (Sekhon 2011) 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 et al (2009) 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 et al. (2009), 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 (1989) or a slightly modified likelihood ratio test (Clauset et al. 2009).

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. (2006). The only exception is when we compare specialisation of projects and contributors in few activity types (i.e., Figs. 8 and 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.

2.7 $\widetilde{\mathbf{T}}$ 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 (Holander and Wolfe 1973). The second step uses the *t*-test or the rank-based Wilcoxon-Mann-Whitney test (Wilcoxon 1945), with Bonferroni correction (Dunn 1961; Sheskin 2007). Unfortunately, the global test null hypothesis may be rejected while none of the sub-hypotheses are rejected, or vice versa (Gabriel 1969). Moreover, simulation studies suggest that the Wilcoxon-Mann-Whitney test is not robust to unequal population variances, especially in the unequal sample size case (Zimmerman and Zumbo 1992). 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 (2002) 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 $\widetilde{\mathbf{T}}$ (Konietschke et al. 2012). Moreover, $\widetilde{\mathbf{T}}$ is robust against unequal population variances.

[8] As explained in Section 2.4 we do not include the `unknown` activity type.

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 (Kurtz et al. 1965) (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.

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 (Brunner and Munzel 2002). 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 (Dunnett 1955). Matrix C_T expresses all pairwise comparisons (up to symmetry) and is known as "all pairs" or Tukey-type contrast (Tukey 1951). 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:

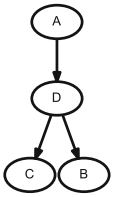
- First, for each pair of groups we analyse the 95 % confidence interval to test whether the corresponding null sub-hypothesis can be rejected. If the lower boundary of the interval is greater than zero for groups A and B, then we claim that the metric value is higher in A than in B. Similarly, if the 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 (Aho et al. 1972), 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.

Table 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

Activity type	Developers																			
A	2	2	2	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	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	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



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 Akritas et al. (1997). Table 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 the $\widetilde{\mathbf{T}}$ procedure for the Tukey-type contrast and 95 % confidence level. Results of the $\widetilde{\mathbf{T}}$ procedure are summarized in Table 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 Table 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 (Antoniol et al. 2005; Khomh et al. 2009). However, Wilcoxon-Mann-Whitney is not robust against differences in variance (Zimmerman and Zumbo 1992; Brunner and Munzel 2000). The $\widetilde{\mathbf{T}}$ procedure as described above cannot be applied to comparison of two distributions (Konietschke et al. 2012). 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 (Brunner and Munzel 2000), 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 (Antoniol et al. 2005; Khomh et al. 2009), we also report the results of this test.

3 Goal 1: How Does Workload Vary Across Projects?

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?

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 (Taube-Schock et al. 2011). 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 (Clauset et al. 2009). Next, we consider the log-normal distribution. Since the data contains ties we opt for the bootstrapped Kolmogorov-Smirnov test (Sekhon 2011). 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 Fig. 5.

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

Figure 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

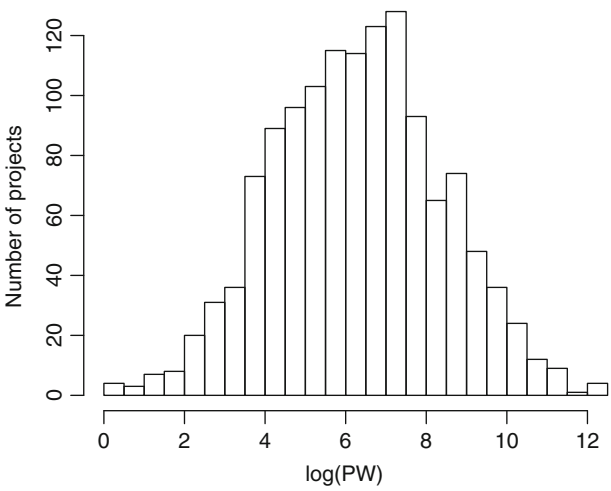


Fig. 5 The workload $PW(p)$ is distributed log-normally

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.

3.2 Which Projects are More Active?

3.2.1 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 Fig. 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 ($e^{0.64562} \simeq 1.9$).

> 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 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*.

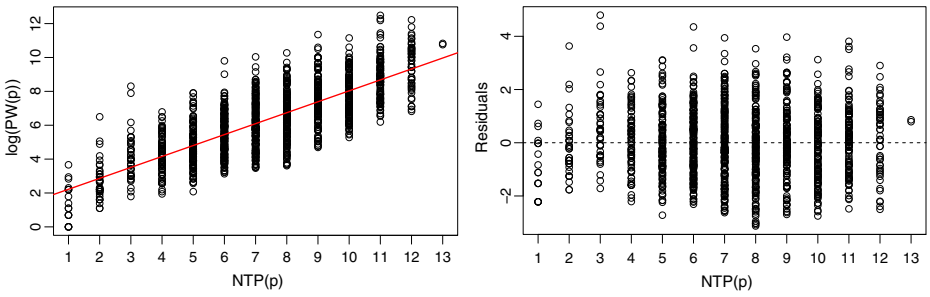


Fig. 6 *Left* observed linear relation between $NTP(p)$ and $\log PW(p)$ (regression line drawn in red). *Right* residuals plot

3.2.2 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.

3.3 How Specialised are Projects Towards Different Activity Types?

Let us first explore how the ecosystem workload varies across the different activity types. Figure 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.

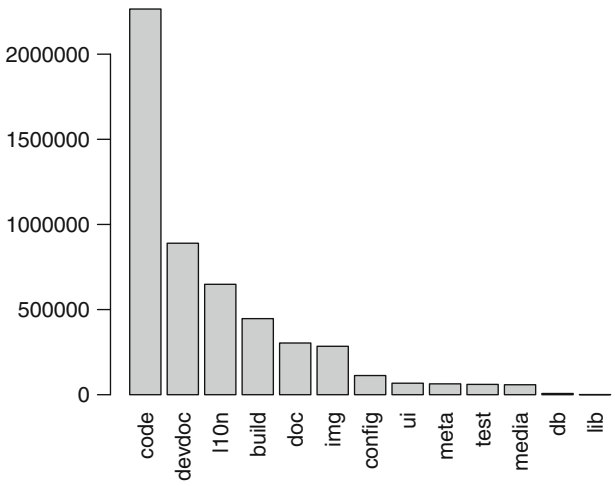


Fig. 7 Type workload: activity types `code`, `devdoc`, `l10n`, and `build` account for the highest workload share in the ecosystem

3.3.1 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 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. (2009), 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. (2009). 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

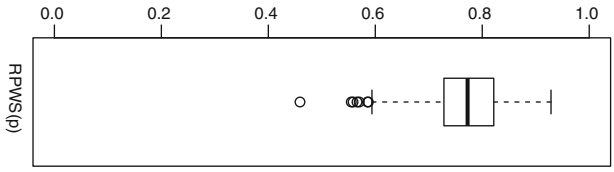


Fig. 8 Relative project workload specialisation RPWS(p): Most projects concentrate their workload in few activity types

activity types. As explained in Section 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.

*3.3.2 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 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, Fig. 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

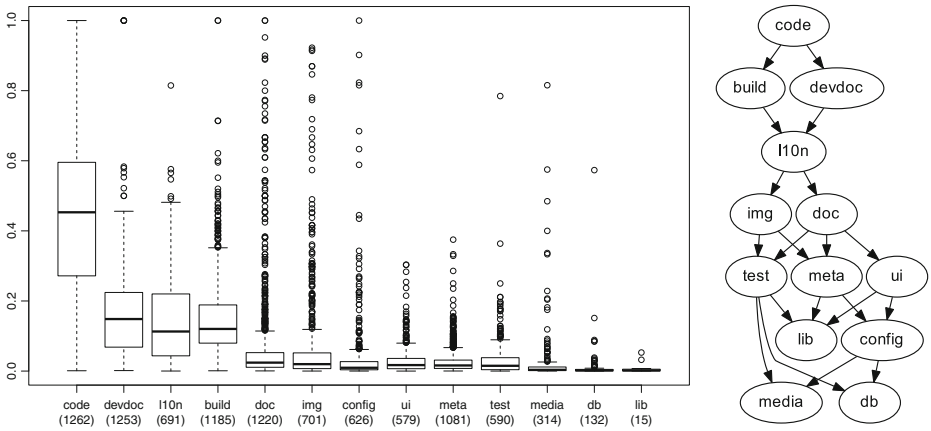


Fig. 9 Boxplots for relative project workload per activity type t. Per boxplot, zero values are excluded. `code` is the predominant activity type at project level: in most projects that contain coding, it represents around 50 % of the workload. `devdoc`, `l10n`, and `build` each account for 10–20 % of 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 2.6)

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.

The variation of $RPTI(p,t)$ across projects per activity type t is illustrated in Fig. 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 Fig. 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

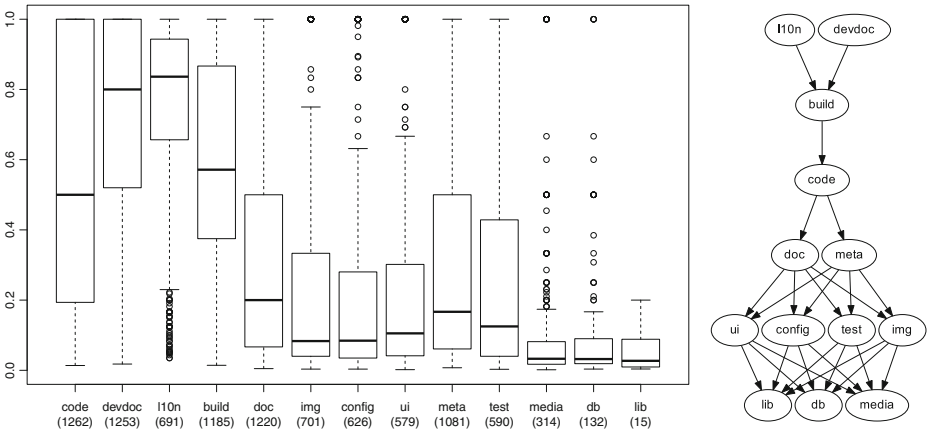


Fig. 10 Boxplots for relative project involvement per activity type t. Per boxplot, zero values are excluded. `l10n` attracts the highest fraction of the project authors. `db` activities are performed only by a small fraction of the project authors. The multiple contrast test procedure $\widetilde{\mathbf{T}}$ 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 2.6)

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.

3.4 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.

3.4.1 Which Project Characteristics are Observed When it is Specialised Towards Few Activity Types?

In Section 3.3.1 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.

3.4.2 To What Extent Does Project Community Size Relate to the Workload (Share) for a Particular Activity Type?

In Section 3.2.2 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 3.3.2. The results of the correlation tests are visually summarised in Fig. 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.

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.

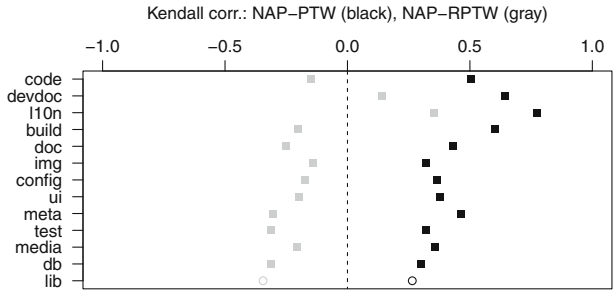


Fig. 11 As more authors become involved in the projects, the workload increases the most in `l10n`, `devdoc`, `build`, and `code` (*black*). In terms of shares, it is the workload in *localization* that increases the most relative to those in other activity types (*gray*)

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 (2004), 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.

3.4.3 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 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

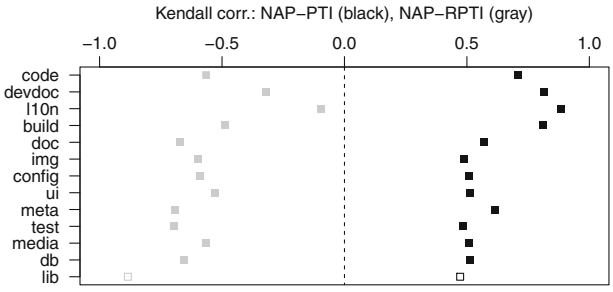


Fig. 12 As more authors become involved in the projects, they mostly contribute to `l10n`, `devdoc`, `build`, and `code` (*black*). The percentage of developers involved in `l10n` does not decrease as more developers become involved in the projects (*gray*)

($\tau = -0.88$), 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 3.4, 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.

3.5 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.

4 Goal 2: How Does Workload Vary Across Authors?

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?

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 (Fig. 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 (Clauset et al. 2009). 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 (Sekhon 2011) 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 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 ($\log 185874 \simeq 12.13$).

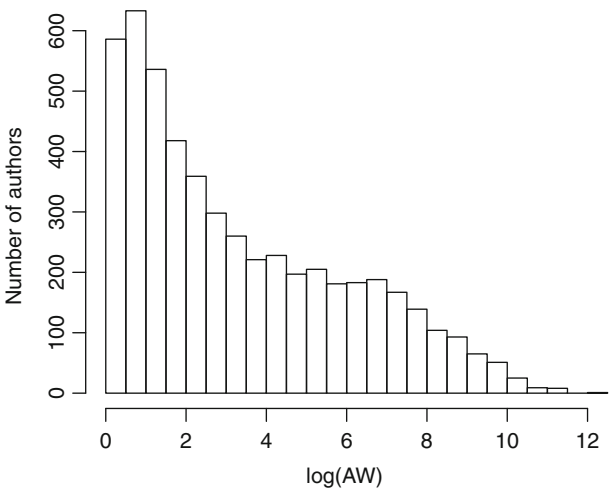


Fig. 13 Distribution of author workload AW is heavy-tailed but does not follow a power law or log-normal distribution

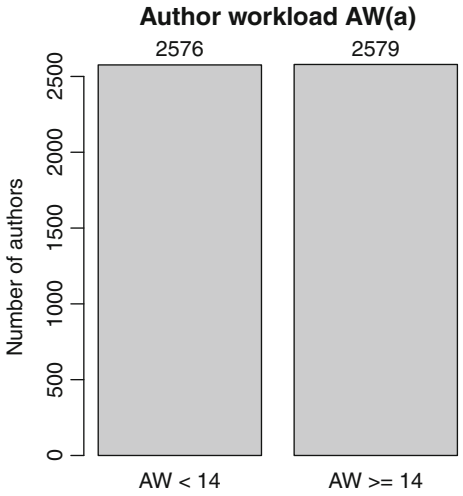


Fig. 14 Approximately half of the authors performed less than 14 touches to GNOME files

This conclusion is concurrent with the observation by Neary and David (2010) that the top 40 developers have made 31 % of all changes, while the most prolific 5 % of developers have made 65 % of all changes.

4.2 Which Kind of Authors are More Active?

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 $AW(a)$ ($\tau = 0.737$).

The scatter plot of Fig. 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

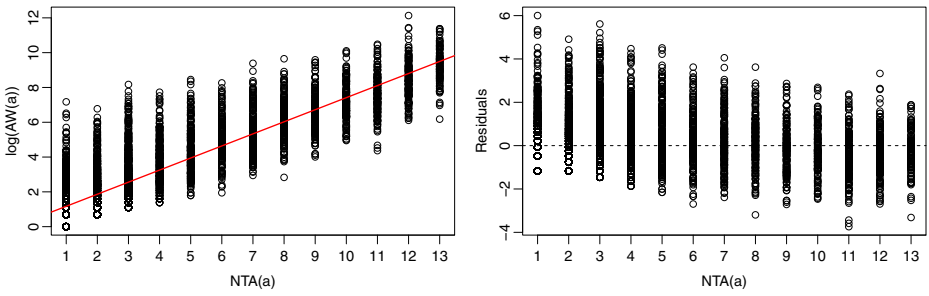


Fig. 15 *Left* observed linear relation between $NTA(a)$ and $\log AW(a)$ (regression line drawn in red). *Right* residuals plot

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 15 also reveals that 1452 (28 %) authors are involved in a single activity type. In Section 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.

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.

4.3 How Specialized are Authors Towards Different Activity Types?

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 3.3. 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 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 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

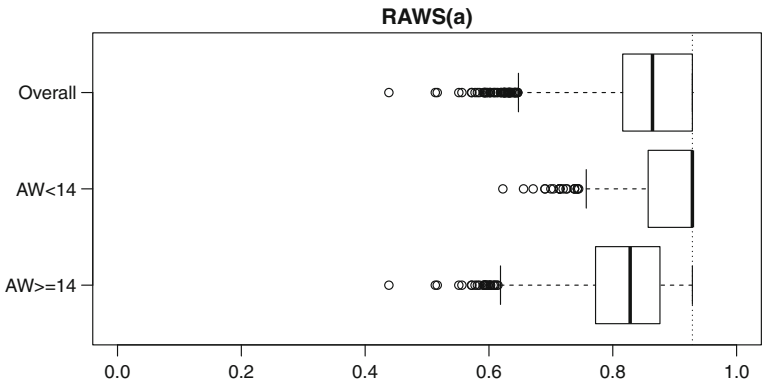


Fig. 16 Relative author-workload specialisation $RAWS(a)$

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 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, Fig. 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.

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 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 Figs. 7 and 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 GNOMEcommunity. A similar finding has been reported for KDE by Robles et al. (2006).

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 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." (Villa 2007)

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 Fig. 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 $RATW(a,$ `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 $RATW(a,$ `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 $RATW(a,$ `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

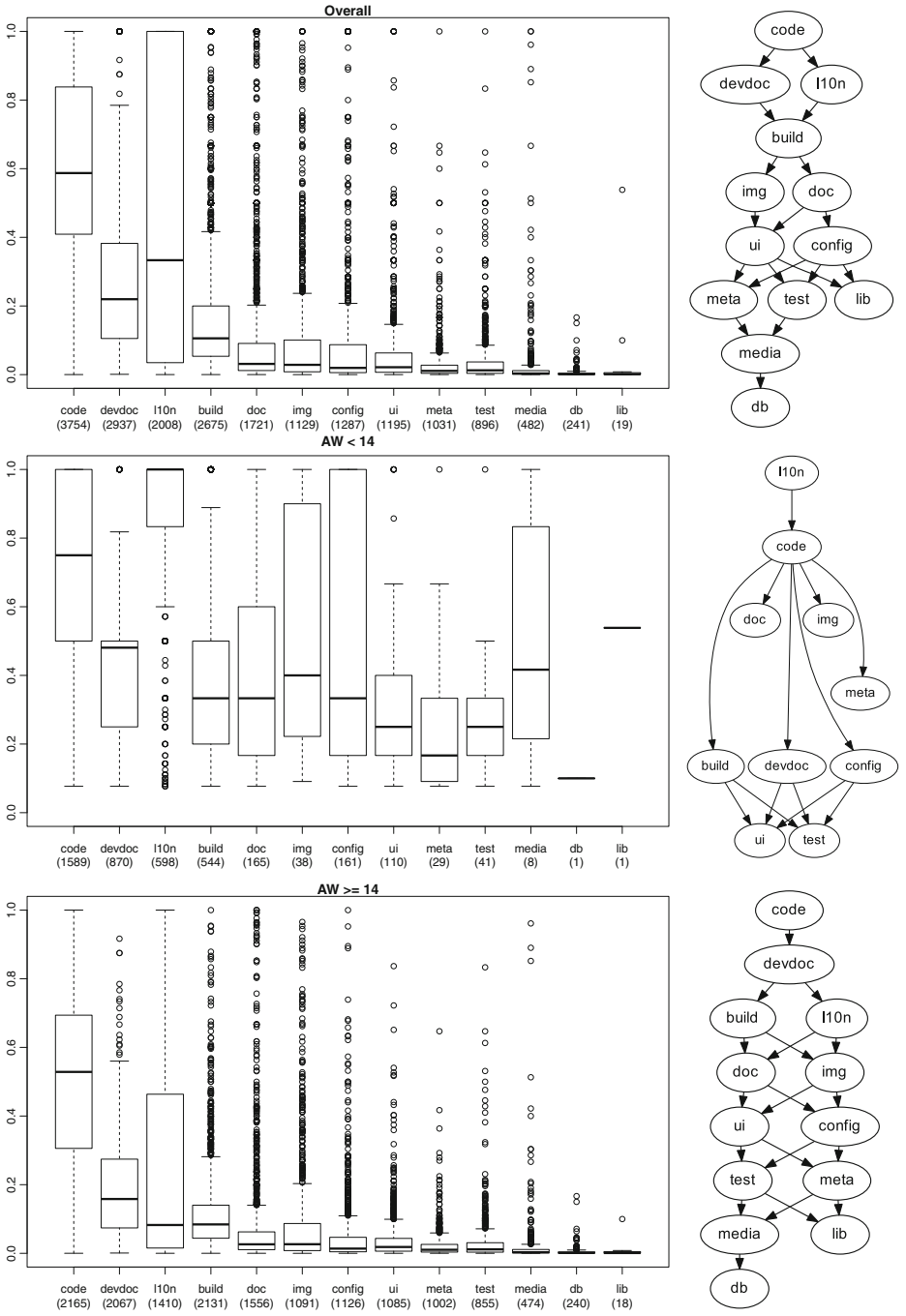


Fig. 17 Boxplots for $RATW(a,t)$ per activity type t. Zero values are excluded. By definition of $RATW(a,t)$ ($AW(a)$ appears in the denominator), the lower whiskers in the $AW < 14$ boxplots cannot be lower than 1/13. For $AW < 14$ the $\widetilde{\mathbf{T}}$-graph does not include `db` and `lib` since the $\widetilde{\mathbf{T}}$ procedure is not applicable for groups of size one

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. (2011).

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 Fig. 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 Fig. 18–left). While this classification is only qualitative, it is confirmed by our quantitative analysis (Fig. 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 touches per author, so more fine-grained than the number of commits per committer used by Neu et al. (2011). 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. (2011): coders are typically very active contributors involved in a relatively small number of projects, while translators are

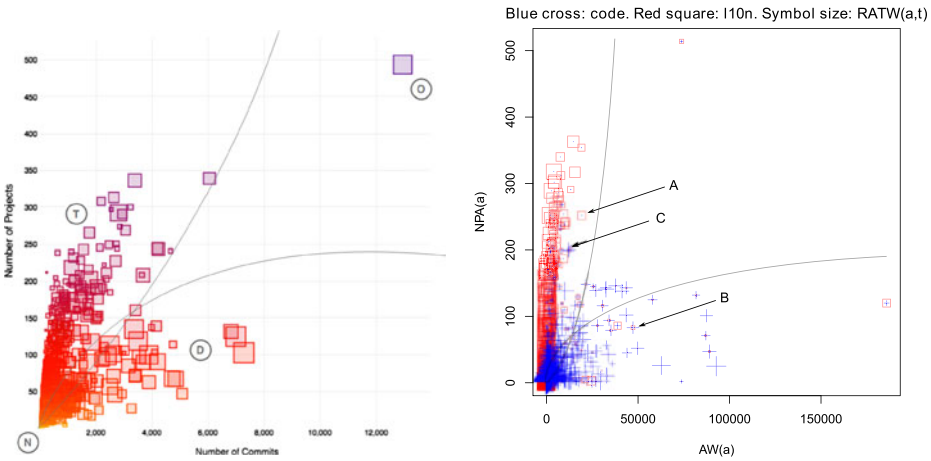


Fig. 18 *Left* visualization by Neu et al. (2011): each square depicts a committer; the number of projects is encoded both on the y-axis and in the colour of each square; the size of a square corresponds to the lifetime in days of a committer within Gnome; Ⓣ: translators, Ⓓ: developers, Ⓞ: outlier, Ⓝ: 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

less active but are involved in more projects. Examples of potential misclassifications 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 inline 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." Rose (2007)

> "As you have pointed out yourself, translators are usually not hackers/coders." Rose (2001)

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 Fig. 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 Fig. 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 Fig. 19. Visual comparison of both sets of boxplots reveals striking differences: in

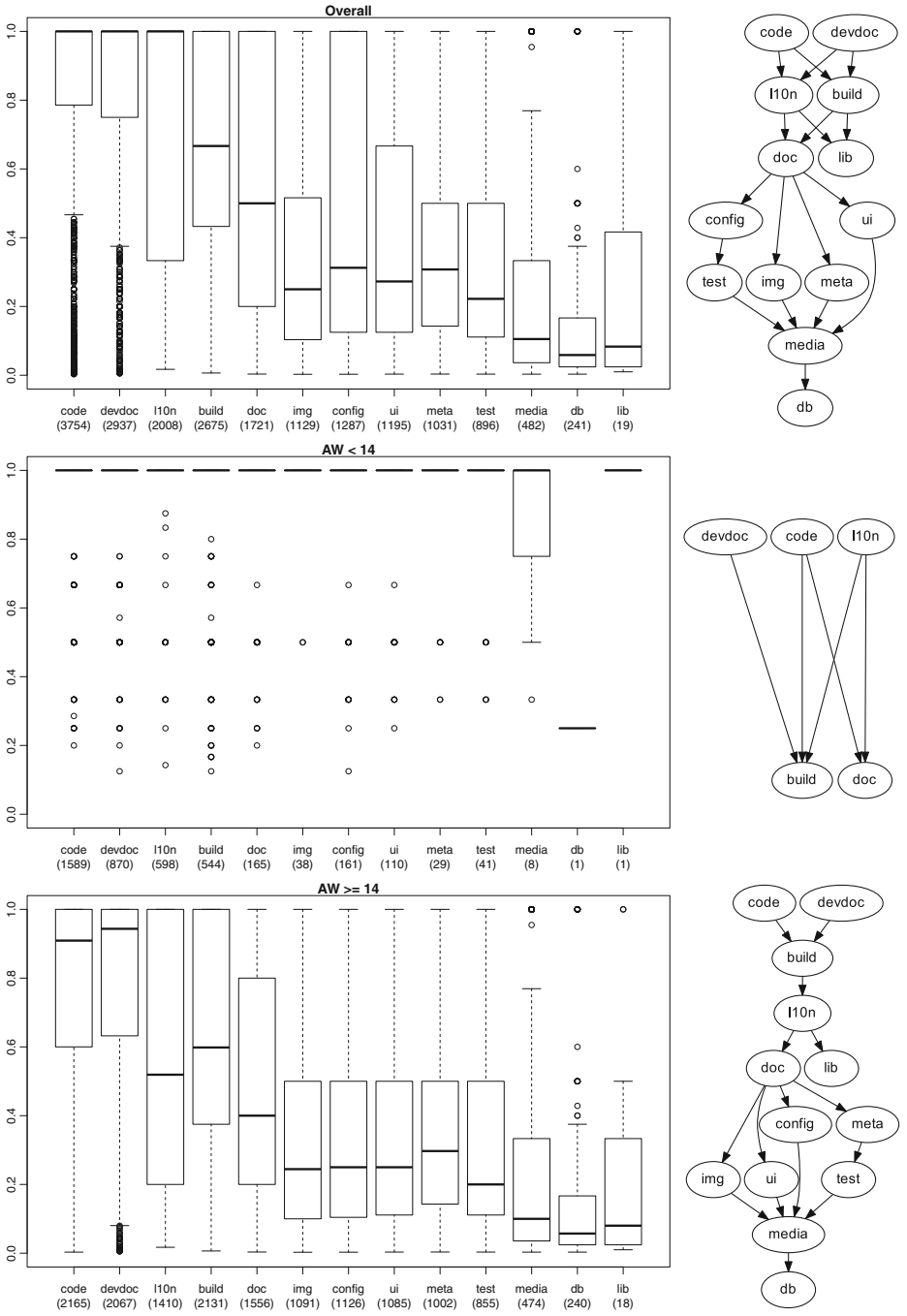


Fig. 19 Boxplots for $RATI(a,t)$ per activity type t. Zero values are excluded. For $AW < 14$ the $\widetilde{\mathbf{T}}$-graph does not include `db` and `lib` since the $\widetilde{\mathbf{T}}$ procedure is not applicable for groups of size one

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 2,576) of the occasional contributors only participate in a single project.

On the other hand, the results (and $\tilde{\mathbf{T}}$-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 ($AW = 1{,}459$) 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.

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.

4.4.1 Which Characteristics of an Author are Observed When She is Specialised Towards Few Activity Types?

In Fig. 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 3.4.1, 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.

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

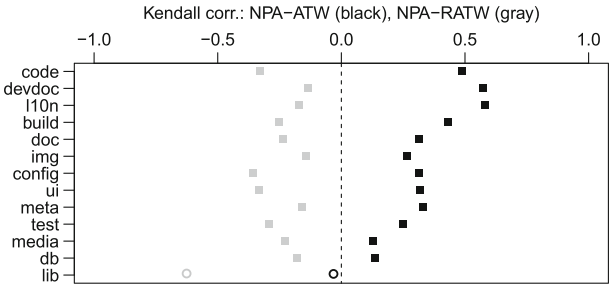


Fig. 20 As they are involved in more projects, more authors concentrate their workload in `l10n` and `devdoc` than in other activity types (*black*). In addition, the share of their workload in `code` decreases in comparison to that in `l10n` (*gray*)

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 4.3.2, we discard the authors for which $ATW(a,t) = 0$. Figure 20 visually summarises the results of the correlation tests for ATW (black) and $RATW$ (gray), using the same visual conventions as in Figs. 11 and 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 Fig. 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.

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 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 $ATI(a,$ `code`$)$ and $ATI(a,$ `l10n`$)$ both match increases in $NPA(a)$ (high positive correlation), it is only $RATI(a,$ `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. (2011) 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`,

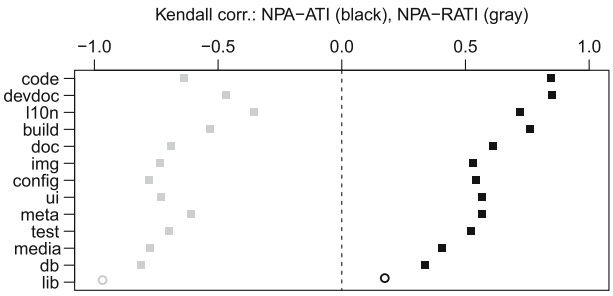


Fig. 21 As they are involved in more projects, more authors contribute to `code` and `devdoc` in their new projects than to `l10n` (*black*). In addition, the share of their projects in which they `code` decreases in comparison to `l10n` (*gray*)

the Web application used to manage the localization of GNOME[9] (cf. description of `intltool`, one of the predecessors of `Damned Lies` in Souphavanh and Karoonboonyanan (2005, p. 44)).

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 four 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.

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

[9] l10n.gnome.org

author identification technique described in Section 2.3. As recognised by Goeminne and Mens (2011a), *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 paper. 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 (2011a).

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. (2006). 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 (D'Ambros and Lanza 2009; Valverde 2007). A similar proxy of the development effort has been used by German (2003).

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 1,358 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 (Wohlin et al. 2000) could not have affected the results of our study. Furthermore, we have paid special attention to the appropriate use of statistical machinery (Sheskin 2007) (cf. Sections 2.6 and 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 paper 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 Gnome Live! translation project or `Damned Lies`, the Web application used to manage the localization of Gnome.

[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.

6 Related Work

In the case study of this article 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 2.6 and 2.7, while identity matching approaches are discussed in Section 2.3. In Section 6.1 we discuss existing results related to studies of developers and their activities, and in Section 6.2 we discuss studies of the Gnome ecosystem.

6.1 Studies of Open Source Software Contributors

Many researchers have investigated the roles developers play in open source software projects. Capiluppi et al. (2003) 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 (2009) 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. (2002) 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. (2002), Nakakoji et al. (2002) 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 (2005). 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 (Moon and Sproull 2000).

[11] freecode.com

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 involvement in other projects to bug reporting. However, as shown by Poncin et al. (2011), integration of the refined classification proposed by Nakakoji et al. (2002) necessitates additional analysis of bug tracker information. Yu and Ramaswamy (2007) made a similar distinction between core and associate project members, but unlike Nakakoji et al. (2002) 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. (2006), 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.

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. Linstead and Baldi (2009) and Schackmann and Lichter (2009) mined the Gnome Bugzilla database, and Shihab et al. (2009) mined the Gnome Internet Relay Chat (IRC) meeting channels. In this paper, we investigated a different data source, namely the version control repositories. Lungu et al. (2009) 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. Casebolt et al. (2009) 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 (Hindle et al. 2010). There were two contributions to this mining challenge that used Gnome as a case study. Krinke et al. (2010) focused on the reuse and cloning of code between the different Gnome projects. Luijten et al. (2010) 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 paper.

Similarly to Neu et al. (2011) 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. (2011) 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 (2002) 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 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 (Taube-Schock et al. 2011). A similar distribution seems to be suggested by partial data on percentages of modification requests per developer, as reported by German (2003). As opposed to our work, Koch and Schneider (2002) 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 (2002) follow a holistic approach, we augment such results with more fine-grained analyses of individual Gnome projects. Similarly to Koch and Schneider (2002), Gousios et al. (2008) 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. (2011) 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. (2011) 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. (2006) studied relations between the Gnome developers by means of social network analysis, while Ernst and Mylopoulos (2010) studied perception of software quality requirements in some of the Gnomeprojects.

7 Future Work

Our research can be extended in numerous ways.

While in this article 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 (Cowell and Jenkins 1995; Serebrenik and van den Brand 2010; Serebrenik et al. 2011).

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 (Goeminne and Mens 2013).

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 (Gousios et al. 2008; Poncin et al. 2011). 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 (Capiluppi et al. 2012b). These measures can be used both by software developers looking for a job and by recruiters evaluating suitability of such candidates (Capiluppi et al. 2012a).

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.

8 Conclusion

This article 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 Gnomecase study, a large and well-known open source ecosystem and associated community (with over 1,300 different projects and over 5,000 active authors).

By analyzing 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 Gnomecommunity 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 Gnomeactivity 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, parameterised 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.

Acknowledgements We thank Javier Perez and Romuald Deshayes for proofreading a draft version of this article. We are also grateful to Dr. Koo Rijpkema for a number of discussions on certain aspects of statistical analysis and Dr. Frank Konietschke for providing us with the (yet to be published) implementation of the $\widetilde{\mathbf{T}}$ procedure. Moreover, we thank the anonymous reviewers for their numerous remarks that helped us to improve the article significantly.

This research has been partially supported by research projects FRFC 2.4515.09 financed by Fonds de la Recherche Scientifique (F.R.S-FNRS), ARC AUWB-08/12-UMH-3 and AUWB-12/17-UMONS-3 financed by the Ministère de la Communauté française—Direction générale de l'Enseignement non obligatoire et de la Recherche scientifique (Belgium), and NWO 600.065.120.10N235 financed by the Dutch Science Foundation (Nederlandse Organisatie voor Wetenschappelijk Onderzoek, NWO). Part of this research has been carried out during the second author's stay at the Université de Mons, supported by grant BSS-2012/V 6/5/015 of the Fonds de la Recherche Scientifique (F.R.S-FNRS).

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	.*\.((s|x|g|p|(gt))?)htm(l?)		.*/translators	.*/contributors
	.*/doc(-?)book(s?)/.*	.*\.page	.*/info	.*\.1
doc	.*\.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
	.*\.eps	.*\.vcf	.*/doc(s?)/.*	.*/thanks
	.*\.xsd	.*\.schemas	.*/help(s?)/.*	.*/authors
	.*\.texi	.*\.doc	.*/bugs	.*\.docx
Image	.*\.eps	.*\.ppm	.*\.icns	.*\.eps
	.*\.pgm	.*\.jpg	.*\.chm	.*\.xbm
img	.*\.jpeg	.*\.bmp	.*\.chm	.*\.vdx
	.*\.gif	.*\.sgv(z?)	.*\.nsh	.*\.ico
	.*\.xcf			
Localization	.*/.potfiles\.in(~?)	.*\.i18ns(~?)	.*\.pot(~?)	/po/.*
	/strings.properties	.*\.mo(~?)	.*\.linguas	.*\.wxl
l10n	.*\.gmo(~?)	.*\.resx(~?)	.*/locale(s?)/.*	.*\.po(~?)
	.*\.charset(~?)			
User interface	.*\.glade(\d?)((\.bak)?)(~?)		.*\.desktop	.*\.xul(~?)
	.*\.gladed(\d?)((\.bak)?)(~?)		.*\.ui	.*\.xpm
ui	.*\.gladep(\d?)((\.bak)?)(~?)		.*\.theme	
Multimedia	.*\.mp3	.*\.mp4	.*/media(s?)/.*	.*\.pfm
	.*\.mpv	.*\.mml	.*/font(s?)/.*	.*\.gnect
media	.*\.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	.*\.dmg(~?)	.*\.swg(~?)	.*\.so(~?)	.*\.i(~?)
	.*\.o(~?)	.*\.exe(~?)	.*\.oafinfo(~?)	.*\.pyd(~?)
code	.*\.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(~?)	.*\.hẗemplate((\.swp)?)(~?)	

Activity type acronym	Regular expressions			
	.*\.js((\.swp)?)(~?)	.*\.rb((\.swp)?)(~?)	.*\.ctemplate((\.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)?)(~?)	.*\.ctmpl((\.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((ṡwp)?)(~?)	
Meta	.*\.svn(.*)	.*\.git(.*)	.*\.doap	.*\.mdp
	.*\.cvs(.*)	.*\.bzr(.*)	.*\.mds	.*\.vbg
meta	.*\.sln			
Configuration	.*\.conf	.*\.cfg	.*\.anjuta	.*\.dsw
	.*\.gnorba	.*\.project	.*\.pgp(~?)	.*\.ini
config	.*\.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	.*\.cmake	.*/install-sh	.*/build/.*	.*\.ezt
	.*\.cbp	.*\.pch	.*/pkg-info	.*\.wxilib
build	.*\.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	.*readme.*	.*/changelog.*	.*\.dia(~?)	.*\.ical
documentation	.*/changes	.*/status	.*/fixme	.*\.doxi
devdoc	.*/todo.*	.*/hacking.*	.*/news.*	.*/roadmap
	.*\.rst	.*/devel(-?)doc(s?)/.*		
Database	.*\.sql	.*\.sqlite	.*\.mdb	.*\.yaml
	.*\.sdb	.*\.dat	.*\.yaml	.*\.json
db	.*\.db	.*/berkeleydb.*/.*		
Testing	.*\.test(s?)/.*	.*/.*test\..*	.*/test.*\..*	
test				
Library	.*/library/.*	.*/libraries/.*		
lib				

References

Aho AV, Garey MR, Ullman JD (1972) The transitive reduction of a directed graph. SIAM J Comput 1(2):131–137

Akritas M, Arnold S, Brunner E (1997) Nonparametric hypotheses and rank statistics for unbalanced factorial designs. J Am Stat Assoc 92:258–265

Allison PD (1978) Measures of inequality. Am Sociol Rev 43(6):865–880

Antoniol G, Di Penta M, Harman M (2005) Search-based techniques applied to optimization of project planning for a massive maintenance project. In: Int conf softw maint. Inst Electr Electron Eng, pp 240–249

Baxter G, Frean M, Noble J, Rickerby M, Smith H, Visser M, Melton H, Tempero E (2006) Understanding the shape of Java software. SIGPLAN Not 41(10):397–412
Bettenburg N, Hassan AE (2010) Studying the impact of social structures on software quality. In: Int conf program comprehension. Inst Electr Electron Eng, pp 124–133
Bird C, Gourley A, Devanbu PT, Gertz M, Swaminathan A (2006) Mining email social networks. In: Min softw repos. Assoc comput mach, pp 137–143
Bonaccorsi A, Giannangeli S, Rossi C (2006) Entry strategies under competing standards: hybrid business models in the open source software industry. Manag Sci 52(7):1085–1098
Brown BM, Hettmansperger TP (2002) Kruskal-Wallis, multiple comparisons and Efron dice. Aust N Z J Stat 44(4):427–438
Brunner E, Munzel U (2000) The nonparametric Behrens-Fisher problem: asymptotic theory and a small-sample approximation. Biom J 42(1):17–25
Brunner E, Munzel U (2002) Nichtparametrische Datenanalysen: Unverbundene Stichproben. Statistik und ihre Anwendungen, Springer
Capiluppi A, Lago P, Morisio M (2003) Characteristics of open source projects. In: Conf softw maint reengineering. Inst Electr Electron Eng, pp 317–327
Capiluppi A, Serebrenik A, Singer L (2012a) Assessing technical candidates on the social web. IEEE Software 30(1):45–51
Capiluppi A, Serebrenik A, Youssef A (2012b) Developing an h-index for OSS developers. In: Min softw repos. Inst Electr Electron Eng, pp 251–254
Casebolt JR, Krein JL, MacLean AC, Knutson CD, Delorey DP (2009) Author entropy vs. file size in the GNOME suite of applications. In: Min softw repos. Inst Electr Electron Eng, pp 91–94
Christen P (2006) A comparison of personal name matching: Techniques and practical issues. In: Int conf data min. Inst Electr Electron Eng, pp 290–294
Christen P, Churches T, Hegland M (2004) Febrl—a parallel open source data linkage system. In: Adv knowl discov data min. Lect Not Comput Sci, vol 3056. Springer, pp 638–647
Clauset A, Shalizi CR, Newman MEJ (2009) Power-law distributions in empirical data. SIAM Rev 51:661–703
Cowell FA (2000) Measurement of inequality. In: Handbook of income distribution. Handbooks in economics, vol 1. Elsevier, pp 87–166
Cowell FA, Jenkins SP (1995) How much inequality can we explain? A methodology and an application to the United States. Econ J 105(429):421–430
D'Ambros M, Lanza M (2009) Visual software evolution reconstruction. J Softw Maint Evol 21:217–232
Davies J, German D, Godfrey M, Hindle A (2011) Software bertillonage: finding the provenance of an entity. In: Min softw repos. Assoc comput mach, pp 183–192
Dinh-Trong T, Bieman J (2005) The FreeBSD project: a replication case study of open source development. Trans Softw Eng, Inst Electr Electron Eng 31(6):481–494
Dunn OJ (1961) Multiple comparisons among means. J Am Stat Assoc 56:52–64
Dunnett CW (1955) A multiple comparison procedure for comparing several treatments with a control. J Am Stat Assoc 50(272):1096–1121
Ernst N, Mylopoulos J (2010) On the perception of software quality requirements during the project lifecycle. In: Requirements engineering: foundation for software quality. Lect Not Comput Sci, vol 6182. Springer, pp 143–157
Gabriel KR (1969) Simultaneous test procedures—some theory of multiple comparisons. Ann Math Stat 40(1):224–250
German DM (2003) The GNOME project: a case study of open source, global software development. Softw Process Improv Pract 8(4):201–215
German DM (2004) Using software trails to reconstruct the evolution of software. J Softw Maint Evol 16(6):367–384
Gini C (1921) Measurement of inequality of incomes. Econ J 31:124–126
Goeminne M, Mens T (2011a) A comparison of identity merge algorithms for software repositories. Sci Comput Program. Available online 1 Dec 2011, ISSN 0167-6423. doi:10.1016/j.scico.2011.11.004.
Goeminne M, Mens T (2011b) Evidence for the Pareto principle in open source software activity. In: Int workshop softw qual maintainab
Goeminne M, Mens T (2013) Analysing ecosystems for open source software developer communities. In: Software ecosystems: analyzing and managing business networks in the software industry. Palgrave-MacMillan

Gousios G, Kalliamvakou E, Spinellis D (2008) Measuring developer contribution from software repository data. In: Min softw repos. Assoc comput mach, pp 129–132
Hindle A, Godfrey MW, Holt RC (2007) Release pattern discovery: A case study of database systems. In: Int conf softw maint. Inst Electr Electron Eng, pp 285–294
Hindle A, Herraiz I, Shihab E, Jiang ZM (2010) Mining challenge 2010: FreeBSD, GNOME desktop and Debian/Ubuntu. In: Min softw repos. Inst Electr Electron Eng, pp 82–85
Holander M, Wolfe DA (1973) Nonparametric statistical methods. Wiley
Iqbal A, Hausenblas M (2012) Integrating developer-related information across open source repositories. In: 13th Int Conf Information reuse and integration (IRI), 2012 Inst Electr Electron Eng, pp 69–76
ISO/IEC/IEEE (2009) Standard 9945:2009 information technology—portable operating system interface (posix) base specifications. Issue 7
Jergensen C, Sarma A, Wagstrom P (2011) The onion patch: migration in open source ecosystems. In: Gyimóthy T, Zeller A (eds) SIGSOFT found softw eng. Assoc comput mach, pp 70–80
Kendall MG (1938) A new measure of rank correlation. Biometrika 30(1–2):81–93
Khomh F, Di Penta M, Guéhéneuc YG (2009) An exploratory study of the impact of code smells on software change-proneness. In: Work conf reverse eng. Inst Electr Electron Eng, pp 75–84
Knuth D (1973) The art of computer programming, vol 3. Sorting and searching. Addison Wesley
Koch S, Schneider G (2002) Effort, co-operation and co-ordination in an open source software project: GNOME. Inf Syst J 12(1):27–42
Konietschke F (2012) nparcomp. Reference manual
Konietschke F, Hothorn LA, Brunner E (2012) Rank-based multiple test procedures and simultaneous confidence intervals. Electron J Stat 6:738–759
Kouters E, Vasilescu B, Serebrenik A, van den Brand MGJ (2012) Who's who in Gnome: using LSA to merge software repository identities. In: Int conf softw maint. Inst Electr Electron Eng, pp 592–595
Krinke J, Gold N, Jia Y, Binkley D (2010) Cloning and copying between GNOME projects. In: Min softw repos. Inst Electr Electron Eng, pp 98–101
Kurtz TE, Link RF, Tukey JW, Wallace DL (1965) Short-cut multiple comparisons for balanced single and double classifications: part 2. Derivations and approximations. Biometrika 52(3–4):485–498
Levenshtein VI (1966) Binary codes capable of correcting deletions, insertions, and reversals. Sov Phys Dokl 10(8):707–710
Linstead E, Baldi P (2009) Mining the coherence of GNOME bug reports with statistical topic models. In: Min softw repos. Inst Electr Electron Eng, pp 99–102
Little T (2006) Schedule estimation and uncertainty surrounding the cone of uncertainty. IEEE Software 23(3):48–54
Lopez-Fernandez L, Robles G, Gonzalez-Barahona J, Herraiz I (2006) Applying social network analysis techniques to community-driven libre software projects. Int J Inf Technol Web Eng 1(3):27–48
Lorenz MO (1905) Methods of measuring the concentration of wealth. J Am Stat Assoc 9(70): 209–219
Louridas P, Spinellis D, Vlachos V (2008) Power laws in software. Assoc Comput Mach: Trans Softw Eng Meth 18:2:1–2:26;
Luijten B, Visser J, Zaidman A (2010) Assessment of issue handling efficiency. In: Min softw repos. Inst Electr Electron Eng, pp 94–97
Lungu M, Malnati J, Lanza M (2009) Visualizing GNOME with the small project observatory. In: Min softw repos. Inst Electr Electron Eng, pp 103–106
Lungu M, Lanza M, Gîrba T, Robbes R (2010) The small project observatory: visualizing software ecosystems. Sci Comput Program 75:264–275
de Mendiburu F (2010) Agricolae. Practical manual. Faculty of Economics and Planning, La Molina National Agrarian University, La Molina, Lima, Peru
Mens T, Goeminne M (2011) Analysing the evolution of social aspects of open source software ecosystems. In: Int workshop softw ecosystems, CEUR-WS, pp 1–14
Mockus A, Fielding RT, Herbsleb JD (2002) Two case studies of open source software development: Apache and Mozilla. Assoc Comput Mach: Trans Softw Eng Meth 11(3):309–346
Moon JY, Sproull L (2000) Essence of distributed work: The case of Linux kernel. First Monday 5(11). http://firstmonday.org/issues/issue5_11/moon/index.html. Accessed December 2011
Mordal K, Anquetil N, Laval J, Serebrenik A, Vasilescu B, Ducasse S (2012) Software quality metrics aggregation in industry. J Softw Evol Proc. doi:10.1002/smr.1558

Nakakoji K, Yamamoto Y, Nishinaka Y, Kishida K, Ye Y (2002) Evolution patterns of open-source software systems and communities. In: Int workshop princ softw evol. Assoc comput mach, pp 76–85
Neary D, David V (2010) The GNOME census: who writes GNOME? In: GNOME users and developers European conference
Neu S, Lanza M, Hattori L, D'Ambros M (2011) Telling stories about GNOME with complicity. In: Intl workshop vis softw underst anal. Inst Electr Electron Eng, pp 1–8
Noether GE (1981) Why Kendall tau? Teach Stat 3(2):41–43
Pearson K (1895) Note on regression and inheritance in the case of two parents. Royal Soc Proc 58:240–242
Poncin W, Serebrenik A, van den Brand MGJ (2011) Process mining software repositories. In: Conf softw maint reengineering. Inst Electr Electron Eng, pp 5–14
Press WH, Teukolsky SA, Vetterling WT, Flannery BP (2002) Numerical recipes in C/C++: the art of scientific computing code. Cambridge University Press
R Development Core Team (2010) R: a language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria
Robles G, González-Barahona JM (2005) Developer identification methods for integrated data from various sources. In: Min softw repos. Assoc comput mach, pp 106–110
Robles G, Gonzalez-Barahona JM, Merelo JJ (2006) Beyond source code: the importance of other artifacts in software development (a case study). J Syst Softw 79(9):1233–1248
Robles G, González-Barahona JM, Izquierdo-Cortazar D, Herraiz I (2009) Tools for the study of the usual data sources found in libre software projects. Int J Open Source Softw Process 1(1):24–45
Rose C (2001) Re: Handling Translations. https://mail.gnome.org/archives/gnome-web-list/2001-August/msg00073.html. Accessed December 2011
Rose C (2007) Re: Git vs SVN (was: can we improve things?). https://mail.gnome.org/archives/foundation-list/2007-September/msg00050.html. Accessed December 2011
Schackmann H, Lichter H (2009) Evaluating process quality in GNOME based on change request data. In: Min softw repos. Inst Electr Electron Eng, pp 95–98
Sekhon JS (2011) Multivariate and propensity score matching software with automated balance optimization: the matching package for R. J Stat Softw 42(7):1–52
Serebrenik A, van den Brand MGJ (2010) Theil index for aggregation of software metrics values. In: Int conf softw maint. Inst Electr Electron Eng, pp 1–9
Serebrenik A, Vasilescu B, van den Brand MGJ (2011) Similar tasks, different effort: Why the same amount of functionality requires different development effort? In: 10th Belg-Neth softw evol semin, pp 4–5
Sheskin DJ (2007) Handbook of parametric and nonparametric statistical procedures, 4th edn. Chapman & Hall
Shibuya B, Tamai T (2009) Understanding the process of participating in open source communities. In: Emerg trends in free/libre/open-source softw. Inst Electr Electron Eng, pp 1–6
Shihab E, Jiang ZM, Hassan A (2009) On the use of internet relay chat (IRC) meetings by developers of the GNOME GTK+ project. In: Min softw repos. Inst Electr Electron Eng, pp 107–110
Souphavanh A, Karoonboonyanan T (2005) Free/open source software: localization. United Nations Asia Pacific Development Information Programme
Spearman C (1904) The proof and measurement of association between two things. Am J Psychol 15(1):72–101
Stone D (2004) Re: [fdo] Re: on translation regressions due to freedesktop.org dependencies. https://mail.gnome.org/archives/gnome-i18n/2004-July/msg00146.html. Accessed December 2011
Taube-Schock C, Walker RJ, Witten IH (2011) Can we avoid high coupling? In: Eur conf object-oriented program. Lect not comp sci, vol 6813. Springer, pp 204–228
Terceiro A, Rios LR, Chavez C (2010) An empirical study on the structural complexity introduced by core and peripheral developers in free software projects. In: Braz symp softw eng. Inst Electr Electron Eng, pp 21–29
Theil H (1967) Economics and information theory. North-Holland
Theil H (1971) Principles of econometrics. John Wiley
Tsay JT, Dabbish L, Herbsleb J (2012) Social media and success in open source projects. In: Comp support coop work companion. Assoc comput Mach. New York, NY, USA, pp 223–226
Tukey JW (1951) Quick and dirty methods in statistics, part II. Simple analysis for standard designs. In: Am soc qual control, pp 189–197

Valverde S (2007) Crossover from endogenous to exogenous activity in open-source software development. Europhys Lett 77(2):20,002
Vasa R, Lumpe M, Branch P, Nierstrasz OM (2009) Comparative analysis of evolving software systems using the Gini coefficient. In: Int conf softw maint. Inst Electr Electron Eng, pp 179–188
Vasilescu B, Serebrenik A, van den Brand MGJ (2011a) By no means a study on aggregating software metrics. In: Workshop emerg trends softw metr. Assoc comput Mach, pp 23–26
Vasilescu B, Serebrenik A, van den Brand MGJ (2011b) You can't control the unfamiliar: A study on the relations between aggregation techniques for software metrics. In: Int conf softw maint. Inst Electr Electron Eng, pp 313–322
Villa L (2007) Re: GNOME Project Organogram. https://mail.gnome.org/archives/marketing-list/2007-February/msg00027.html. Accessed December 2011
Vuong QH (1989) Likelihood ratio tests for model selection and non-nested hypotheses. Econometrica 57(2):307–333
Waugh J (2007) GNOME community celebrates 10 years of software freedom, innovation and industry adoption. https://mail.gnome.org/archives/gnome-announce-list/2007-August/msg00048.html. Accessed December 2011
Weber S (2004) The success of open source. Harvard University Press
Wilcoxon F (1945) Individual comparisons by ranking methods. Biom Bull 1(6):80–83
Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslén A (2000) Experimentation in software engineering: an introduction. Kluwer
Yu L, Ramaswamy S (2007) Mining CVS repositories to understand open-source project developer roles. In: Min softw repos. Inst Electr Electron Eng, p 8
Zaidman A, Rompaey BV, van Deursen A, Demeyer S (2011) Studying the co-evolution of production and test code in open source and industrial developer test processes through repository mining. Empir Softw Eng 16(3):325–364
Zeileis A (2009) ineq: Measuring Inequality, concentration, and poverty. R Foundation for Statistical Computing
Zimmerman DW, Zumbo BD (1992) Parametric alternatives to the Student t test under violation of normality and homogeneity of variance. Percept Mot Skills 74(3(1)):835–844
Zobel J, Dart P (1996) Phonetic string matching: lessons from information retrieval. In: Int conf res and dev inf retr. Assoc comput mach, pp 166–172

Bogdan Vasilescu is a PhD student at the Eindhoven University of Technology, The Netherlands. Previously, he obtained a MSc degree in computer science and engineering from the same university in 2011, and a Dipl.-Ing. degree in systems and computer science from the Petroleum-Gas University of Ploiesti, Romania in 2009. Bogdan's research interests so far revolved around aggregation techniques for software metrics and empirical analyses of software ecosystems and developer communities.

Alexander Serebrenik is an assistant professor of model-driven software engineering at Eindhoven University of Technology. His research interests include software evolution and maintenance, social media, program analysis, and transformation. Serebrenik received a PhD in computer science from Katholieke Universiteit Leuven. He's a member of IEEE and the European Research Consortium for Informatics and Mathematics Working Group on Software Evolution. Contact him at a.serebrenik@tue.nl or on Twitter @aserebrenik.

Mathieu Goeminne is a PhD student at the Software Engineering Lab of the University of Mons, Belgium. After a Master degree in Computer Sciences obtained in 2009, he started to study the evolution of open source ecosystems. His research interests cover the social interactions observed in the communities involved in software development.

Tom Mens directs the Software Engineering Lab at the Department of Computer Science, Faculty of Sciences, University of Mons, Belgium. His research interests are in software evolution, model-driven software engineering, empirical analysis and human-computer interaction. He is involved in several research projects and he published numerous articles in these domains in international scientific conferences and journals. He co-edited the Springer book Software Evolution with Serge Demeyer in 2008 and was program co-chair of CSMR 2011, CSMR 2012 and ICSM 2013. He chairs the ERCIM Working Group on Software Evolution, and is a member of IEEE, the IEEE Computer Society and the European Association of Software Science and Technology (EASST).