



Software Development Metrics With a Purpose

Jesus M. Gonzalez-Barahona, Universidad Rey Juan Carlos

Daniel Izquierdo-Cortázar, Bitergia

Gregorio Robles, Universidad Rey Juan Carlos

A new generation of toolsets that are flexible enough to adapt to the data analytics needs of a given scenario is emerging to analyze free, open source software (FOSS). GrimoireLab is one such toolset that meets many of the needs of foundations, developers, and companies.

The use of metrics to better understand free, open source software (FOSS) development is becoming commonplace. For projects themselves, using metrics to show how they work is a new level of openness and transparency (open development analytics). For stakeholders, such as companies depending strategically on FOSS components, metrics allow for the evaluation and tracking of potential problems in the software supply chain. For developers, metrics may help to detect issues, such as bottlenecks in processes or problems in the onboarding process of new fellows, or find out about the performance in processes, such as fixing bugs or

completing code reviews, and how to improve them.

These different needs and objectives make the approach of “a single metric fits all” impossible: metrics have to be customized to help to reach specific goals, or they are of little use. They also have to be visualized and organized in ways that are useful for understanding the

underlying FOSS projects. That is the main reason why a new generation of toolsets is emerging to analyze FOSS (and software development in general). Instead of focusing on providing a predefined set of metrics, their aim is to be flexible enough to adapt to the data analytics needs of a given scenario, providing a flexible and customizable toolchain that can also interoperate with other analytics tools. In this article, we introduce GrimoireLab, one such toolset, and show how it is being used in different cases (“personas”): FOSS foundations, companies consuming FOSS, and FOSS developers.

ANALYZING DATA FROM FOSS PROJECTS

About 20 years ago, when we started to consider FOSS projects as a matter of research, one of their most



interesting singularities (compared to other kinds of software projects) was the availability of rich data about how they worked: in addition to the source code, they made public their discussions and comments in mailing lists and, in some cases, issues in bug-reporting systems. Today, the public availability of these kinds of data can be considered as usual and, to some extent, expected for almost any FOSS project with any reasonable impact. In the early 2000s, however, this was still a novelty.

Before FOSS projects started to voluntarily share data about how they developed software, data-based research about software development was based on the analysis of just a few projects shared by companies with a handful of researchers under very strict nondisclosure agreements. Stakeholders of software components only knew what their producers wanted to leak about their development practices. Researchers had a lot of trouble producing reproducible studies, or even studies at all, because detailed data about software development were scarce and difficult to share.

FOSS open development changed this landscape completely, allowing for the blooming of research based on mining data from software development repositories and, with time, availability of data useful to get insights about the reliability and risks associated with how components are produced. The Mining Software Repositories Conference (<http://www.msrfconf.org/>), started in 2004, is a good showcase of this evolution.

A common thought in the 2000s was that this wealth of data about FOSS project development could be utilized like data about the conditions of persons is used to learn about their health. In the same way that we can analyze the levels of certain substances in a blood test to determine

FROM THE EDITOR

A hallmark of scientific progress is to add quantitative assessments to otherwise purely qualitative evaluations. In a previous article in this column, we looked at the Community Health Analytics for Open Source Software project, which defined possible metrics for such quantitative assessments. In this article by Barahona et al., we also look at tools for quantitatively assessing open source projects. This way, we can calculate metrics and relate them to actual project success, thereby also evaluating how meaningful these metrics are. Progress! Happy hacking, be open, be safe!—*Dirk Riehle*

the health of the person, we could analyze some parameters of how a project is developed and determine the health of that project. We were interested in finding pre-established parameters whose thresholds pointed to certain “health-related issues” in projects.

we have not been able to identify a set of metrics or key performance indicators useful to assess the health of a project, even for specific domains or kinds of projects. However, we learned that data can be used in other ways (both by researchers and stakeholders).

To some extent, GrimoireLab is the natural evolution of our previous tools, expanding its aim beyond research to also provide commercial services of interest to the industry.

For getting there, we had access to data about thousands of FOSS projects, which would allow for finding correlations and invariances that could be translated into symptoms of problems. We started to create a number of tools to gather data from different kinds of sources, help us in the curation of those data, and aggregate added value. The ultimate goal was to produce automated reports of the state of a project, including an analysis of the key parameters that would help to identify “health” problems that could be corrected or, at least, detected in advance.

With time, we found that reality was much more complex than we had expected. Despite the many years that have passed, as a scientific community,

For example, the recent interest by companies and governments in the software bill of materials (SBOM) as a tool to track the origin of software components, especially FOSS components, opens new opportunities to detect problems related to how those components are being produced. By using the right tools, all of the software supply chain for a certain product (all of the FOSS projects producing the components used in it) can be analyzed, finding relevant metrics about how they are developed: how many people are involved in the process, how they deal with vulnerabilities, when they produced new releases, or how they are attracting new developers (or losing them).

As another example, FOSS foundations are starting to provide open

metrics services that can be used by their members but also by any third party as a new standard of transparency [see, for example, the Linux Foundation “Insights” (<https://insights.lfx.linuxfoundation.org>), Mozilla “Community Report” (<https://report.mozilla.community/>), and Wikimedia Foundation “Community Metrics” (https://www.mediawiki.org/wiki/Community_metrics or the report on the history of the Linux kernel¹). These examples show a trend: it is becoming increasingly clear that, to understand the opportunities and mitigate the risks associated with the use of FOSS components, metrics are one of the tools that can help. In general, better understanding allows for better acting, and that is the reason why software development metrics can be of assistance: they help us better understand how FOSS components are developed and, therefore, make

better decisions about them. However, there is no single list of metrics that is good for all purposes.

GRIMOIRELAB: A TOOLSET FOR SOFTWARE DEVELOPMENT ANALYTICS

To fit this need of producing different metrics specific to various objectives, a new generation of toolsets is emerging. GrimoireLab⁴ is one of them. It is distributed as FOSS, which leads to increased transparency, and it can be used to analyze FOSS and non-FOSS projects as long as they use practices and supporting systems similar to those in FOSS. To some extent, GrimoireLab is the natural evolution of our previous tools, expanding its aim beyond research to also provide commercial services of interest to the industry.

Since September 2017, GrimoireLab is a founding project of Community

Health Analytics for Open Source Software (CHAOSS), an undertaking hosted by the Linux Foundation that is focused on creating analytics and metrics to help define community health for FOSS communities (see Goggins et al.⁵). This, to some extent, means that we walked the full circle, being involved once again in defining FOSS project health. Now, however, this is one of the aims for the toolset, among many others. In fact, if something characterizes GrimoireLab, it is its flexibility and support for many different kinds of analysis: it has been used in studies by research teams but also by open source projects to analyze themselves and companies in industrial environments.

GrimoireLab is composed of a collection of components that can collaborate to analyze software development repositories. See Figure 1 for a glimpse of how all of them fit together.

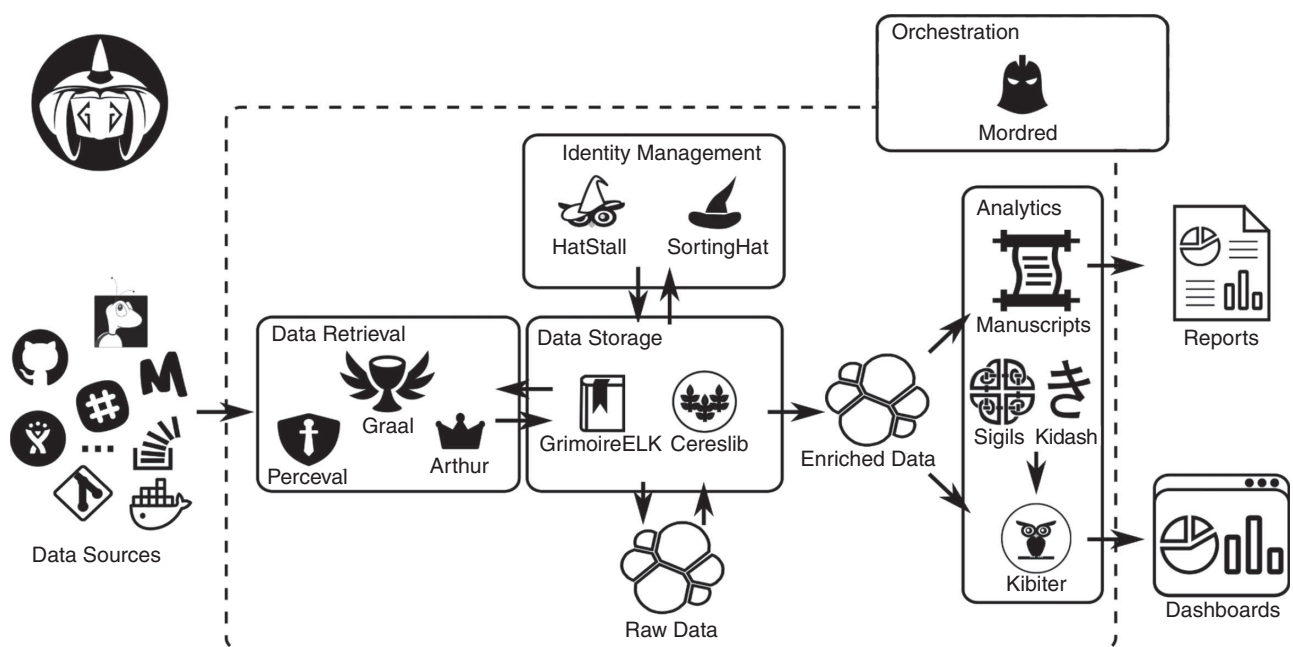


FIGURE 1. The components in the GrimoireLab toolset. Data are first retrieved from data source application programming interfaces by Perceval, with the help of Graal (for running third-party tools on all versions of the source code) or Arthur (to handle job scheduling for large-scale retrieval) if needed, and stored as raw data. Then, they are enriched by GrimoireELK and Cereslib, and identities are merged and made consistent with the help of SortingHat and Hatstall, producing the enriched database. Enriched data are consumed to produce reports and dashboards. Mordred is the component that orchestrates the entire toolchain. (Source: Valerio Cosentino, contributed to the GrimoireLab project; used with permission.)

These components can be used for the following:

- › They can retrieve data, automatically and incrementally, from many kinds of software repositories (data sources).
- › Gathered data can be stored, curated, and enriched. In this process, GrimoireLab also deals with common issues such as identity and affiliation management, merging data from several data sources, and computing process metrics from the data retrieved.
- › The data can be analyzed, producing specific information. For

example, computing thresholds and delays in processes, analyzing the structure of a community and its main contributors, finding relationships in a geographical context, or determining the likely causes of engineering bottlenecks. GrimoireLab tools allow users to deal with several aspects of the development efforts, including community, performance, and activity.

- › They can visualize data in different formats, producing actionable charts and visualizations of several kinds, in which data can be filtered or which can

be used to drill down to explore and find details.

GrimoireLab can also be seen as a black box, consuming data from software development repositories as the input and producing dashboards or reports as the output (see the GrimoireLab dashboards for CHAOSS projects at <https://chaoss.biterg.io>). Both reports and dashboards show metrics and visualizations, but, while the former are static (typically PDF documents), the latter are dynamic (typically web apps that are actionable and running in a browser).



FIGURE 2. The Cauldron community engagement panel for some FOSS projects. At the top left is the number of new pull request submitters over time, the top right shows the evolution of newcomers versus people leaving the community; the bottom left indicates the number of developers attracted (green) versus those who left the project (blue) for different cohorts, and the bottom right shows ratio of the same parameters. PR: pull request; MR: merge request.

GRIMOIRELAB IN PRACTICE

Modern FOSS development uses many different support systems for source code management, issue tracking, continuous integration, asynchronous and synchronous communication, and so on. For each of them, there are usually several options from which FOSS projects select what they feel is more appropriate for them. GrimoireLab provides back ends to extract and store data from more than 25 different systems, including Git, GitHub, GitLab, Bugzilla, Jira, Launchpad, Gerrit, Discourse, mbox archives, Stack Overflow, Jenkins, Internet Relay Chat, Mattermost, Slack, Telegram, Confluence, Mediawiki, and Meetup.

However, all of this diversity also needs some organization. GrimoireLab structures the data for each item it retrieves so that it can be aggregated and filtered with other items, even from different data sources. For example, homogeneous references to dates or authors allow for queries, such as “all activity items between two dates, from this specific author.” Since all of the data can be retrieved and stored together, being able to query in this way is fundamental for building useful dashboards and reports. GrimoireLab can also use structured information about how projects are grouped (for example, for large collections of repositories), how

the many identities of a developer can be merged (such as using authors files maintained by the projects themselves), or specific characteristics of a developer (for instance, to produce metrics for all developers working for a certain company).

Setups based on GrimoireLab can be deployed in several ways. The project provides PyPi Python packages and Docker images ready to be installed and run. Python packages are especially well suited for using some of the tools in isolation. Docker images are more suitable for complete deployments of the toolset: when properly configured, those deployments can automatically retrieve

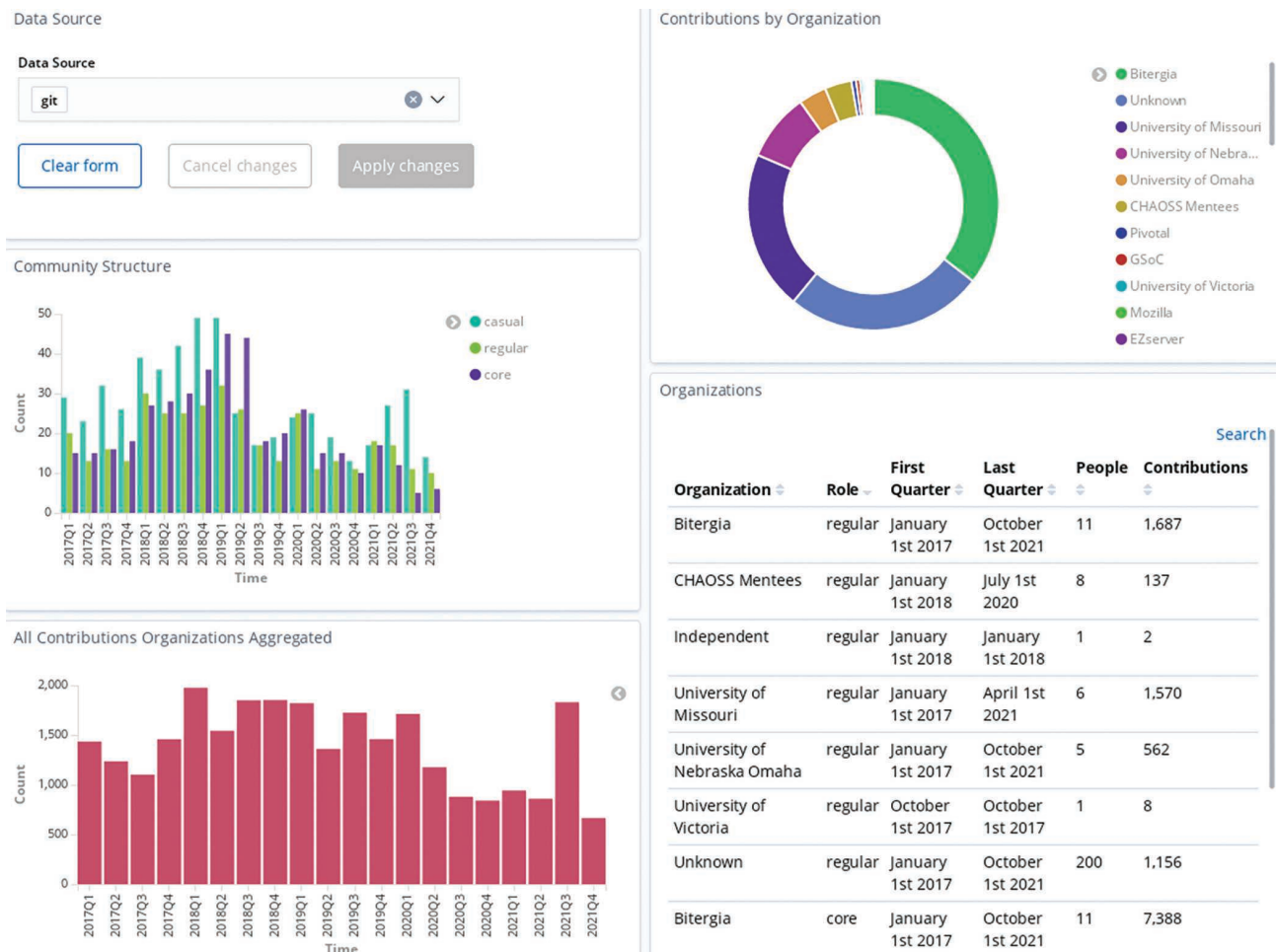


FIGURE 3. An example of a dashboard designed for the FOSS foundation persona. It shows how companies and organizations contribute to a set of projects and the evolution over time. The data are about CHAOSS, shown by GrimoireLab. Q: quarter; GSoC: Google Summer of Code.

data from thousands of projects, store them in a database, analyze them, and produce visualizations—all of this out of the box.

However, it takes some time and certain technical skills to deploy a working system from packages or container images. Most users will prefer to just interact with the resulting visualizations or check the metrics of their interest. For that, they can use Cauldron, the SaaS version of GrimoireLab, which currently supports some of the data sources supported by GrimoireLab: GitHub and GitLab, bare git, Meetup, and Stack Exchange (including Stack Overflow).

Cauldron (Figure 2), which is also FOSS, can be deployed for private use (Cauldron Cloud) or directly used in the public Cauldron.io (<https://cauldron.io>) instance. It has a friendly and simple web-based user interface to select the set of repositories to analyze, or an SBOM document (in Software Package Data Exchange format or as a list of repositories) can be uploaded to analyze all projects in a software supply chain. It automatically retrieves data from all

of them, producing an actionable dashboard that can be used in a browser. Cauldron offers data about the main process and community metrics of the repositories, which, in many situations, is good enough to learn what the user needs, for example, when first approaching a FOSS project. It can also compare the results obtained by several different projects, which is convenient when evaluating similar options.

PERSONAS INTERESTED IN FOSS DEVELOPMENT METRICS

We have summarized how GrimoireLab can show many different aspects of FOSS projects (and of software projects in general) by producing dashboards and reports with different metrics and visualizations. To show how this approach helps in coping with the needs of different stakeholders, we will use the “persona” metaphor. In our case, a persona is an actor with a specific profile, analyzing projects with a specific purpose. We focus on three main personas: FOSS foundations as umbrellas of the FOSS projects they host, acting

as neutral playgrounds; consumers, usually companies, of FOSS components; and individual contributors to FOSS projects.

FOSS foundation persona

Neutrality and transparency are the most important goals of this persona, so it can act as a neutral playground. In this case, data about how the software is being developed constitute a new transparency layer. Data contribute to transparency in the same way that publishing source code, a clear distribution license, or a detailed (and public) decision making process does: by making the project more predictable, letting anyone evaluate the risks and potentials, putting more information in the hands of stakeholders.

FOSS foundations (Figure 3) can use the data in many areas. We mention only some cases in which we have been directly or indirectly involved: code review fairness,³ committer elections (as OPNFV and other Linux Foundation projects do), diversity and inclusion,² or decisions about the maturity of projects (as the Apache

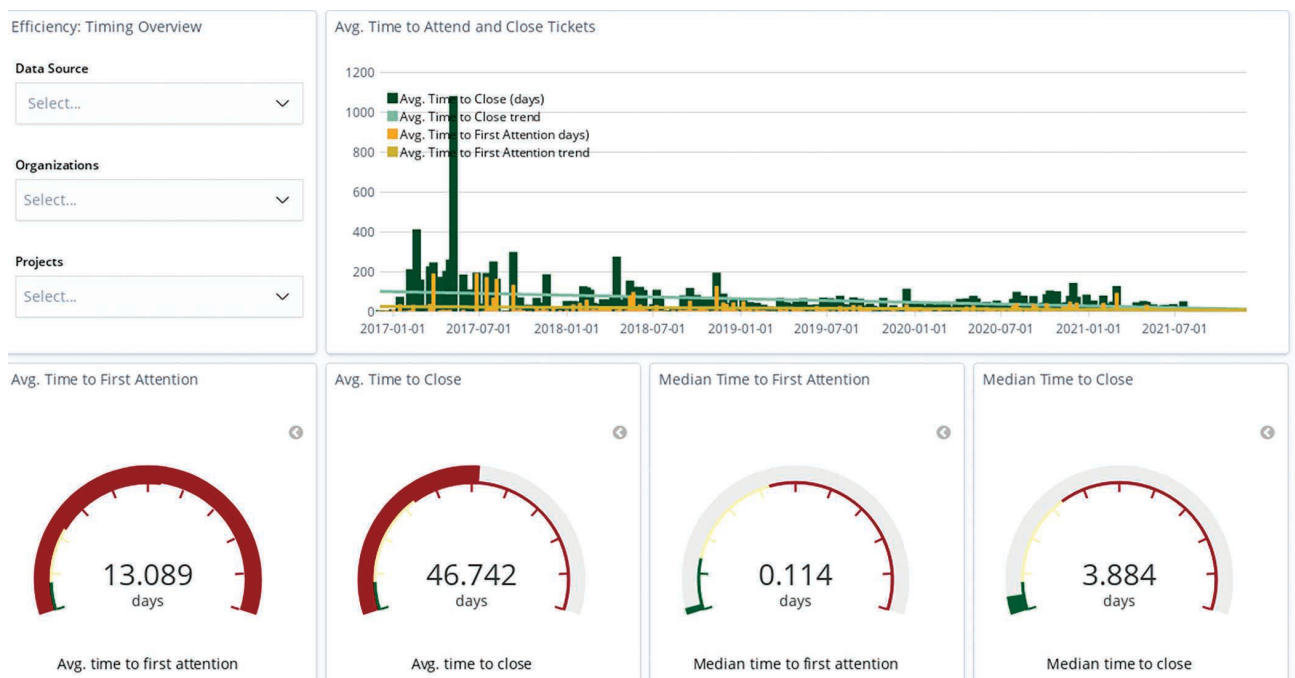


FIGURE 4. An example of a dashboard designed for the consumer persona. It shows several metrics related to how the project reacts to issues. Data are about CHAOSS, shown by GrimoireLab. Avg.: average.

Software Foundation or the Cloud Native Computing Foundation do). Using data and not just opinions helps to cre-

levels. (For an example, see Cloud Native Computing Foundation: <https://landscape.cncf.io/>)

a part of the critical technological stack of corporations, the characteristics of the projects producing those components become a very relevant topic. Company consumers of a certain FOSS component may also decide to collaborate in the project producing it, or, maybe, the project was promoted by them because they wanted to share resources for producing a new component that was key to them. In these cases, the need for data to understand those projects is even more clear.

In any case, FOSS consumers want data to make sensible decisions about the components they consume as well as the projects that produce and maintain them because they are a part of their software supply chain. For them,

Using data and not just opinions helps to create a predictable environment where corporations and other actors can collaborate and work together.

ate a predictable environment where corporations and other actors can collaborate and work together. This is becoming more relevant as FOSS communities are growing in size, complexity, and scale, in some cases acting as an umbrella to thousands of developers and with hundreds of organizations participating at different

FOSS consumer persona

This persona, usually a company, is interested in understanding the sustainability, maintenance efforts, development pace, and process predictability and takes all of these into account when planning and estimating risks and potential benefits (Figure 4). As more FOSS components are

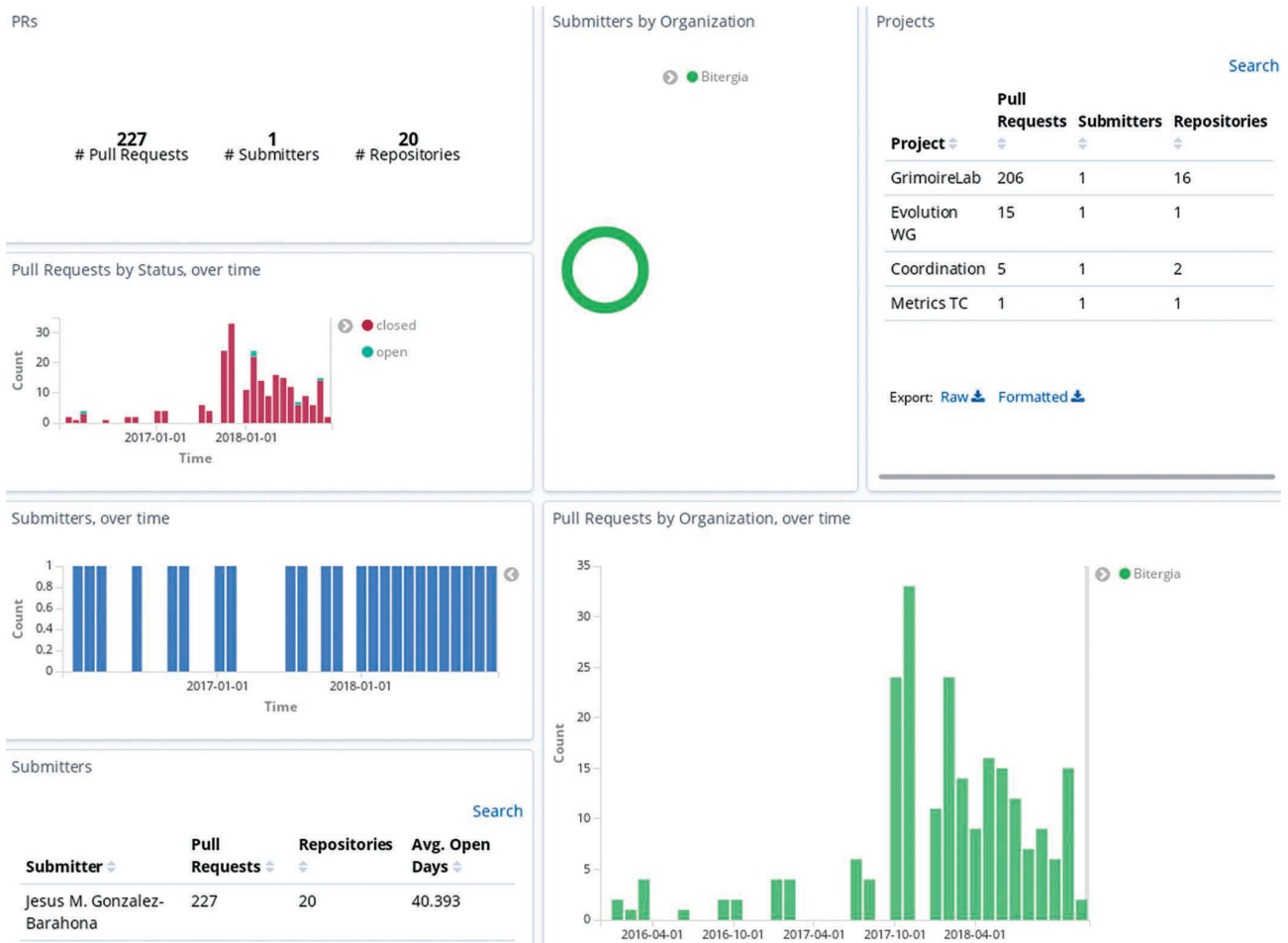


FIGURE 5. An example of a dashboard designed for a contributor persona. In this case, it allows a developer to track its own contributions (pull requests, in this instance) and compare them with contributions by other developers. Data are about CHAOSS, shown by GrimoireLab. TC: technical committee; WG: working group.

many relevant questions can be answered using software development analytics, such as the impact of certain actions; the influence they have on the communities they are part of and dependent on; and many other aspects, such as leadership, growth, community, engagement, diversity, transparency, community health, performance, collaboration, resilience, adoption, and so on.


FOSS contributor persona

This persona represents the case of people participating as individual contributors in FOSS projects (Figure 5). They may do this on behalf of some of the previous personas or just individually, with their own purposes and goals. They may play different roles within the project: developer, project leader, documenter, community manager, developer advocate, and many more. In any case, software development analytics again plays an important role to help them be more efficient or make better decisions. They can use software development analytics for their own purposes and business goals. Some examples of this are community managers nurturing and developing the community; developer advocates making life easier for developers, looking for bottlenecks and providing the best development environment; or individuals learning their current status in the community, who is who, or whom to ask for help for a specific issue.

METRICS WITH A PURPOSE

All of these personas have in common that they find software development analytics useful for them, but each persona needs to use the metrics in different ways and for purposes due to their individual interests, motivations, and context. Therefore, the key question is not which metrics about FOSS are interesting but which ones are important for a certain persona or, more broadly, for a certain goal.

This is the main reason why the new-generation toolsets, such as GrimoireLab, are relevant: instead of providing a hard-coded, specific toolchain targeted to the needs of a specific persona, they can be configured to satisfy many different needs. Their approach of “get all relevant data from software repositories and then let components enrich, query, and visualize it” allows for targeted dashboards and reports for each persona, thus reusing most of the infrastructure but still satisfying very different needs. They allow for setting up a general platform to help different personas effectively make decisions based on data that can be traced back, are reproducible, and are fully accessible to any other player. The fact that the toolset is also FOSS itself is key: all of the details of how the metrics are computed and the final information is produced are available for anyone to inspect, leading to increased transparency and trustability in a domain where nuances may mean errors in key decisions.

Summarizing, metrics can be a key part of the daily life of anyone with an interest in FOSS projects. However, individuals’ takes on which metrics and information are important for them may be very different depending on what they want from those projects. For dealing with such diversity in interests, we need toolsets that are flexible enough but, at the same time, provide the basics to efficiently satisfy all of these needs. 

ACKNOWLEDGMENT

The work presented in this article has been funded in part by the Spanish Government under grants RTI-2018-101963-B-I00 and RTC-2017-6554-7.

REFERENCES

1. K. Stewart, S. Khan, and D. German, “2020 Linux kernel history report,” Linux Foundation, Version v5, Aug.

- 8, 2020. [Online]. Available: <https://www.linuxfoundation.org/tools/linux-kernel-history-report-2020/>
2. D. Izquierdo, N. Huesman, A. Serebrenik, and G. Robles, “OpenStack gender diversity report,” *IEEE Softw.*, vol. 36, no. 1, pp. 28–33, Jan./Feb. 2019, doi: 10.1109/MS.2018.2874322.
3. D. Izquierdo, J. M. Gonzalez-Barahona, L. Kurth, and G. Robles, “Software development analytics for Xen: Why and how,” *IEEE Softw.*, vol. 36, no. 3, pp. 28–32, 2018, doi: 10.1109/MS.2018.290101357.
4. S. Dueñas et al., “GrimoireLab: A toolset for software development analytics,” *PeerJ Comput. Sci.*, vol. 7, p. e601, Jul. 2021, doi: 10.7717/peerj-cs.601.
5. S. P. Goggins, M. Germonprez, and K. Lombard, “Making open source project health transparent,” *Computer*, vol. 54, no. 8, pp. 104–111, Aug. 2021, doi: 10.1109/MC.2021.3084015.

JESUS M. GONZALEZ-BARAHONA

is a professor at Universidad Rey Juan Carlos, Fuenlabrada, 28943, Spain. Contact him at jesus.gonzalez.barahona@urjc.es.

DANIEL IZQUIERDO-CORTÁZAR

is one of the founders of Bitergia, currently holding the position of CEO Leganés, 28929, Spain; part of the governing board of the Community Health Analytics for Open Source Software Working Group; and a member of the board of directors at the InnerSource Commons Foundation. Contact him at dizquierdo@bitergia.com.

GREGORIO ROBLES is a professor at the Universidad Rey Juan Carlos, Fuenlabrada, 28943, Spain. Contact him at greg@gsync.urjc.es.