Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

ARTICLE TYPE

Evolving Software Forges: an Experience Report from Apache Allura

Damian A. Tamburri[*1,2] | **Fabio Palomba**[3]

[1]Eindhoven University of Technology, The Netherlands
[2]Jheronimus Academy of Data Science (JADS), The Netherlands
[3]University of Salerno, Italy

Correspondence
*Damian A. Tamburri, Email: d.a.tamburri@tue.nl

Abstract

The open-source phenomenon has reached unimaginable proportions to a point in which it is virtually impossible to find large applications that do not rely on open-source as well. However, such proportions may turn into a risk if the organisational and socio-technical aspects (e.g., the contribution and release schemes) behind open-source communities are not explicitly supported by open-source forges *by-design*. In an effort to make such aspects explicit and supported by-design in open-source forges, we conducted empirical software engineering as follows: (a) through online industrial surveying, we elicited organisational and social aspects relevant in open-source communities; (b) through action research, we extended a widely known open-source support system and top-level Apache project Allura; (c) through ethnography, we studied the Allura community and, learning from its social and organisational structure, (d) we elicited a metrics framework that support more explicit organisational and socio-technical design principles around open-source communities. This article is an experience report on these results and the lessons we learned in obtaining them. We found that the extensions provided to Apache Allura formed the basis for community awareness by design, providing valuable and usable community characteristics. Ultimately, however, the extensions we provided to Apache Allura were de-activated by its core developers because of performance overheads. Our results and lessons learned allow us to provide recommendations for designing forges, like Github. Architecting a forge is a participatory process that requires active engagement, hence remarking the need for mechanisms enabling it. At the same time, we conclude that a more active support for the governance is required to avoid the failure of the forge.

KEYWORDS:
Software Forges; Software Refactoring; Empirical Software Engineering

1 | INTRODUCTION

Since its early inception, the open-source phenomenon has become an extremely efficient and effective example of global software engineering [1]. However, the organisational and social characteristics reflecting this phenomenon are often left implicit in open-source forge support systems, or *meta-forges*, such as Apache Allura[1]. For example, imagine you want to make use of, contribute to or even start your own open-source communities: What are the best ways to self-organise across GitHub? What are the characteristics of successful communities that belong to the Apache Software Foundation? What is the average participation ratio that you're expected to comply with in any top-level Apache Project? These and similar research questions refer to design characteristics and architectural properties often latent in a software forge. Very little is known on these characteristics

[1]https://forge-allura.apache.org

and how architectural properties for software forges may influence them - we loosely refer to this body of knowledge as *forge design*, that is, *the set of design principles, patterns and decisions that support the socio-technical and organisational aspects in open-source software development communities* [2,3]. Forge design aspects would cover, for example, the means to measure and support community informality or self-organisation degrees or community openness. This body of knowledge is yet to be fully explored and may offer a bounty of governance and design practices for the benefit of current and future software production commons, from industries to forges to crowdsourcing [4,5,6]. Exploring forge design and its principles is paramount for many reasons [7], e.g., to understand and tackle nasty social and organisational problems or barriers that may emerge in large distributed software communities [8] such as distance or culture clash, shared understanding or situation awareness [9,10,11]. Also a clear overview of effective forge design principles is critical, e.g., for software practitioners to choose the appropriate community where to contribute and whose software may benefit systems under development [12,13,14]. In a first attempt at exploring the intriguing discipline of forge design we experimented with a once well-known forge support system called Apache Allura[2], supporting the once well-known SourceForge open-source common[3].

The choice of Allura meets almost perfectly with our research goals of exploring forge design, since: (a) contrarily to GitHub or Bitbucket, Allura is an open-source top-level Apache project, well recognised across the Apache foundation and still reasonably active[4]; (b) we could experiment and extend Allura with requirements elicited though our web-survey; (c) while designing and developing our extensions, we were able to learn from its organisational and social scenario, elaborating and validating metrics for key community organizational and socio-technical characteristics.

More specifically, in our experimentation, we conducted a survey to understand what community aspects need more explicit support in open-source forges. From this survey we distilled a key design principle for forge design, namely, *software community awareness*, defined as the ability of a software forge to show its well-established social (e.g., user background) and organisational characteristics (e.g., collaboration patterns) both towards insiders and outside of its community boundaries.

In open-source terms, supporting this principle explicitly means supporting (e.g., by adding specific fields for projects) and tracking (e.g., computing statistics for) aspects such as contributor profiles, code-review processes, openness, etc. Although this design principle is increasingly being applied (at least partially) in more actual open-source forges such as GitHub, it is still lingering or absent on less widely known and used but still powerful forges such as SourceForge or Bitbucket. In our experience we observed that this lack of support to software community awareness may be responsible for Allura's steady decline in interest and power[5].

Stemming from survey results, and with the goal of making direct experience with forge design our main research question can be phrased as follows:

RQ. *To what extent can we make a previously existing open-source forge community-aware by design?*

This article is an experience report on the 15-month research dive into forge design required to address the research question above, and offers 3 novel and original contributions: (a) a set of socio-technical and organisational requirements elicited through a web-survey designed using concepts from organizations and social networks management research - survey results led us to elaborate *Software Community Awareness* as an essential forge design principle but survey results can be used as starting points for further forge design research; (b) TITANS, that is, "Tools for Injecting communiTy Awareness in opeN-source forgeS" - a validated set of tools to enable community awareness by-design; (c) the evaluation of the TITANS extensions and community awareness metrics' value in a well-known open-source forge such as GitHub.

The rest of this article is structured as follows: Section 2 outlines TITANS features and metrics illustrating our online survey results. Sections 3 elaborates on TITANS implementation, while Section 4 outlines its evaluation. Section 5 elaborates discussions and lessons learned and, further on, Section 6 outlines related work. Finally, Section 7 concludes the article.

2 | ELICITING RELEVANT FEATURES

This section highlights the requirements emerging from our industrial survey on social and organisational community aspects for software development. We distinguish between two sets of elicited requirements: (a) community requirements - i.e., requirements that call for insight on key community characteristics and their tracking; (b) community typing requirements - i.e., requirements that call for distinguishing between multiple community types. The next two sections elaborate on these two sets of requirements.

[2]https://forge-allura.apache.org/p/allura/wiki/Home
[3]https://sourceforge.net/
[4]https://sourceforge.net/blog/apache-allura-becomes-top-level-project/
[5]https://sourceforge.net/blog/advertising-bundling-community-and-criticism/

2.1 | Survey Design

We elicited Community Requirements in Open-Source by conducting a structured web-survey [15] composed of 24 open questions[6] prepared starting from the general characteristics for community awareness and other technical aspects we observed in GitHub, i.e., the most prominent software forge competitor to Allura at the time of inception for this study. The 24 questions were split evenly and arranged—via a card-sorting exercise [16] performed by one of the authors of this paper—in the following categories: (a) acquiring information concerning the characteristics of the respondents; (b) establishing the importance of introducing support for certain (organisational and social) information in the forge, e.g., information concerning contributors' background, their organisations, etc. The questionnaire was also designed taking into account community types as well as social and organisational characteristics from organisational and social networks theory [17,2], a major target for this study. We distributed a link to our questionnaire through different means of communication, more specifically: (a) mailing lists of open-source communities; (b) researchers mailing lists for open-source research summer schools: (c) online open-source user-groups. The entire population to which the questionnaire was sent could not be precisely established but we were able to collect answers from 52 open-source practitioners. 46% of the survey respondents were researchers also operating and contributing to open-source. Also, employees of for-profit businesses represented 26% of the survey respondents, while freelance programmers and members of non-profit organizations were 22% and 4% of the respondents, respectively. Furthermore, it is worth reporting that 62% of the participants were non-core developers, as opposed to the 15% of integrators, 15% having a top-responsibility role (e.g., CEO), and the remaining 8% covering other community roles. More than 60% of the respondents worked in open-source since more than 5 years.

2.2 | Survey Results

To prioritise requirements we ordered them based on frequency, i.e., a higher number of respondents for a desired feature indicated a higher priority. As a result of this ordering, 3 clusters were made by slicing the preferences ratio (in percentage) into three chunks: requirements that received 55-60% preference by respondents were given highest priority; requirements that received 35-50% of respondents' consensus were given medium priority; finally, requirements that received 0 to 35% consensus from respondents were given low priority. General results are summarised on Table 1, along with the information on the support achieved for each of the aspects considered.

The survey highlighted that virtually all open-source projects involve the collaboration of different organizations, with different types of contributor communities and sub-communities therein. Also, different kinds of relationships among collaborating contributors emerge, e.g. working relations on shared artefacts ("co-operations", according to [18]) or collaborations on other projects outside the forge, e.g., collaboration in architecting and maintaining software products ("collaborations", according to [18]). The need for supporting these various aspects was underlined.

However, often none of these relations are supported by technological means. Software forges provide little or no information about developers and the community characteristics surrounding their projects. For example, 50% of respondents stated that it would be useful for them to see the list of projects in which a single organization is involved. Other details about an organization which were deemed as important include its contact information, a description of its activity and the working areas in which the organization operates.

Our survey suggests four foci to be instrumented for effective support of community aspects in open-source.

First, respondents recognised the need for more information on users and their organisational profile. Missing information in this area spans from personal user details (e.g. location, gender, interests, age, etc.) to personal skills' self-assessment possibilities (e.g. self-reported interests). These details are necessary to foster the creation of a contributor community rather than a blind forge of code-writers. Second, respondents recognised the need for being aware of what we called user statistics. These span from information about personal commits and project contributions (e.g. average code lines written per project participated) to personal project type preferences. These details are necessary to foster mutual trust and the creation of collaborativeness through reciprocity within the forge. Third, respondents recognised the need for knowing more (or more explicitly) about the key organization and community governance details [2,19]. Required information in this area spans from including the concept of "organization" in the forge altogether, to organisational success rate for communities and organizations. These details are necessary for contributors to visualise the collocation and exploitation of their contributions. Lack of such details reflects missing social retribution schemes that may increase members' engagement. Fourth, finally, respondents recognised the need for more organisational statistics. These entail the aggregation of key organisational details into metrics necessary to establish both the need and value of contributions within a certain organization. These details are needed to foster contributors' participations to certain organizations instead of others.

As a result, the requirements and survey results reported above lead to conjecture that users (rows 1 and 2 on Tab. 1) and organizations (rows 3 and 4 on Tab. 1) require more explicit detail both in terms of atomic attributes (e.g., personal user details) and in terms of aggregate statistics (e.g., communication characteristics within the forge). We phrase these organizational and socio-technical forge design requirements [20,21] into a single forge design principle, summarized in the following:

[6] The questionnaire is available online: http://tinyurl.com/hcrqbuq.

TABLE 1 Survey results, four foci for improvement.

Focus	Requirement	Priority	Support
User profile	Including personal details about users	High	14
	Including the association between users and existing organizations	High	18
	Allowing users to self-assess their skills	High	10
User statistics	Including data about user's contribution in terms of submitted code to projects on the forge	High	21
	Including data about user's contribution in solving bugs within projects on the forge	High	13
	Including data about user's contribution to the communication within the forge (reported bugs, posted messages, ...)	Medium	24
	Including split data for each category of projects or programming language	Medium	11
	Including preferences of each user for specific categories of projects	Medium	26
Organization profile and Form	Including the concept of organization within the forge	High	21
	Including general details about registered organizations (description, contacts, working areas, ...) and their relations	High	8
	Including the community type, list of projects developed by the organization or to which it has contributed	High	27
	Including the list of users of the forge working for the organization, including their roles and relations	Medium	23
	Including the list of the organization's previous partners	Low	14
	Including details about the success of developed projects	High	11
Organization statistics	Including statistics obtained by aggregating data for each organization member	Medium	20

Community-Awareness By-Design. The ability of a community of practitioners to make its organisational and socio-technical characteristics known to its internal and external environment, fostering correct engagement in the community's practices.

With the term "correct" referred to engagement across the software community, we refer to the definition of engagement from previous work [2], namely, the ability of a community to define and enforce a code-of-conduct specification to govern all community interactions. For example, such codes of conduct are typical in Apache communities which need to adhere to the Apache Software Foundation bylaws and common practices. In the next sections we elaborate on our research attempt at designing community awareness into the Allura forge-support system.

2.3 | Community Types in Open-Source

To proceed with designing for community awareness, we used a combination of content analysis [22] and community typing algorithms [23,19] on our survey data (location, size, frequency of contribution, organizational type, possible roles, and partnership capabilities) to find the typical community types [2] recurring in open-source that would require explicit support. Four types emerged, namely:

- *Formal Networks:* A Formal Network (FN), is an open-source community with frequent and established/agreed commits as well as formal guidelines/rules for contribution, as overseen by a formally appointed Organisational Sponsor (i.e., an organisation which retains governance rights and royalties over produced goods and services [24]) much like Allura itself.

- *Informal Networks:* Informal Networks (IN) are distributed informal communities whose key characteristic is the presence of informal communication alone, e.g. supported by informal agreements among community members. An example of informal networks are code-developer forums such as DreamInCode[7].
- *Networks of Practice:* a Network of Practice (NoP) corresponds to an open-source project made up of large industrial organizations sharing open-source codebases and the practices around them for mutual benefit, much like the Eclipse IDE project. The key characteristic of such communities is the global dispersion (i.e. distance in time, space and culture together) of its members.
- *Informal Communities:* an Informal Community (IC), is more likely a free-lance, small open-source project using informal and occasional commits by sparse contributors with very little formal core-contributors. For this community type, it is the engagement of community members that drives community goals to success.

In support of the above types, we elicited from the state of the art [2,23,25] the following key community characteristics that require explicit support by design:

- **Formality/Informality**: an evaluation of the established procedures, behaviours and or rituals existing within the observed community that limit or widen open collaboration, communication or cooperation - this characteristic is needed to identify FNs and, by inverse, INs;
- **Hierarchisation Degree**: an evaluation of the chain of power imposed or emerging across the community - this is also a key characteristic of FNs, however, its opposite is essential to identify ICs;
- **Membership Selectivity**: an evaluation of the practices enacted to select, scrutinise and/or continuously evaluate would-be or current members of the community - this key characteristic is essential to spot NoPs and FNs;
- **Previous Relations**: an evaluation of the previous relations (social, organisational and technical) across members in a community - this characteristic is key to ICs;
- **Community Openness**: an evaluation of the ease-of-participation in the community, in terms of mentorship, knowledge and contribution transfer - this characteristic is key to ICs and INs;
- **Governance Degree**: an evaluation of the established processes for (self-)governing people and collaborations in the community - this characteristic is key to NoPs and FNs and its opposite is essential to identify INs as well as ICs;
- **Geodispersion**: an evaluation of the cultural and geographical dispersion of members across a community - this is key to establish the existence of NoPs and INs;
- **Self-Organization**: an evaluation of the degrees of organisational freedom existing in the community - this characteristic is essential in ICs;
- **Self-Similarity**: an evaluation of the background and experience similarity across the observed community - this characteristic is essential in NoPs;
- **Engagement**: an evaluation of the degree of participation across the community - this is particularly key in ICs but is relevant to all Networked community types as well, that is, NoPs and INs;

We use these general characteristics[2] to structure our experimental extensions to the Allura platform (see Sec. 3). Later on, we instrument our extensions with statistics and metrics that allowed us to detect the above types, thus evaluating these extensions in action (see Sec. 3.2).

Stemming from the requirements outlined previously, our card-sorting exercise revealed that any tool for supporting community aspects in open-source should cover:

- R1, User Profiling: Open-source community users require a higher set of personal as well as professional information to foster community engagement - for example, more granular skills definition and profiling may foster explicit or implicit mentorship relations to be created across the community;
- R2, User Statistics: statistics on the above profile items are needed to keep track of the value generated by community members as part of the community - for example, the more contributions a member has the more valued his membership across the community;

[7] http://www.dreamincode.net/forums/forum/32-java/

- R3, Organisational Profiling: profiling the organisations across the community is needed to understand community diversity and allow community members to use such diversity to their advantage - For example, some open-source communities may use such diverse sets of contributors as a record of industrial adoption and therefore an indicator for software quality;
- R4, Organisational Statistics: statistics concerning the organisational and social structure of the community is needed, for a variety of aspects, for example, to understand the degree of openness of the community to external contributors. Aspects included in this subset, concern the support of at least four community types, namely:

Technical requirements scope limitations and potential threats to validity are discussed subsequently in Sec. 5.3.

3 | INTRODUCING TITANS

This section describes the implementation of TITANS in the Allura open-source forge. TITANS extensions were designed, developed, tested and validated to support requirements outlined in Section 2.2.

3.1 | Mapping of Relevant Features

TITANS consists of four tools:

- **UProfile** allows adding personal details, skills and availability timeslots to a user profile.
- **UOrg** allows to include the concept of organizations within the forge.
- **MetroForge** collects statistics concerning the previous activity of a user.
- **MetrOrg** is a module which collects statistics about the overall contributions provided by an organization to the projects hosted on the forge.

The first two components, namely **UProfile** and **UOrg**, allow TITANS to tackle requirements of a more functional nature while the latter two components, namely, **MetroForge** and **MetrOrg** allow TITANS to tackle community typing and analytics. More details on the community typing part of TITANS are in Section 3.2 while the rest of this section focuses on outlining **UProfile** and **UOrg** respectively.

First, **UProfile** expands user profile definitions within Allura. The model component of Allura was modified to include additional attributes and methods in the class representing a single user of the forge. This class, simply named `User`, is directly mapped to the Mongo collection with the same name, which persistently stores data about users of the forge. The new personal details are set as optional to guarantee backward compatibility with user profiles created before the introduction of TITANS. They are:

- *Gender*. By default, this field is set to the value 'Unspecified', but each user can freely set it to 'Male', 'Female'.
- *Date of birth*.
- *Country of residence*. This data is mainly useful to help understanding the cultural environment in which the user is currently living.
- *City of residence*. This allows to identify the user's cultural environment with a finer granularity than the country of residence.
- *User's time zone*, specified according to the IANA Timezone Database. [8] Knowing a user's time zone allows to understand whether a user is in his or her working time or not in a certain moment.
- *Weekly availability timeslots*. This data consist in weekly time intervals during which a user is usually working on the projects hosted on the forge. Together with the user's time zone, it allows to easily assess when a user is expected to be available for working on a project. This is useful, for example, when another user is waiting for an answer to a certain message or needs the user to start working on a reported issue.
- *Inactive periods*. By specifying an interval of dates, each user can notice his or her collaborators that, despite usually being available at certain timeslots, he or she is not contribute to the projects hosted on the forge for a certain period, because of a vacation or because of other commitments.

[8] http://www.iana.org/time-zones

- *Links to personal accounts on social networks and auto-summary* Mining sites like Twitter, Facebook, Google+ or Linkedin, delivers personal information about user interests. This promotes more open and sociable information exchange across the forge.
- *Telephone number, or a list of phone numbers*, to provide users with a way of personally contact someone else.
- *Skype account*, giving a further opportunity to directly contact users.
- *A list of personal Web-sites*, considered relevant by users to provide other developers on the forge with personal details, or about current activities.
- *A set of technical skills*, allowing each user to self-assess himself or herself. For each skill, the user is also required to specify his or her level in the selected field, by choosing between the values 'Low', 'Medium' or 'Advanced'. Finally, the user is allowed to enter an optional comment, useful to describe how the skill was developed or enforced.

Skills are represented through the `TroveCategory`, following scheme definitions in the TROVE categorisation system used in Allura[9].

Second, **UOrg** elaborates explicitly the concept of real-life organizations within the Allura forge. The concept of organization consists three different kinds of entities:

- Software firms developing or contributing to develop software projects.
- Foundations supporting software projects, such as the Apache Software Foundation.
- Education and research institutions such as universities, which could contribute to software projects, for example, as part of their research activities.

To elaborate on data for organisational structures within the forge, we developed a separate **UOrg** package. The new package provides users with functionalities to create or update the profile of an organization, to explicitly link a project to the organizations contributing to its development, and to connect each organization to the profiles of its members registered on the forge. The package includes three separate components:

- *Modules representing the concept of organization*. This component also implements the functionalities to create an organization's profile, and allows users to monitor the list of their organizations, to change details related to their memberships.
- *A tool which represents an organisation's public profile*. The tool includes functionalities to represent the profile of the organization and provides the administrators of the organization with a set of functionalities to update the profile of the organization itself.
- *A tool that is automatically installed within a project to include the list of organizations directly contributing to the project itself*. The tool also allows to specify a different level of participation for each organization.

Similarly to users, organizations are represented based on the projects to which they participate in the forge. Consequently, an *organization project*, is automatically created when a user registers a new organization. The registration includes automatically all tools to manage, track and present organisational details.

Following principles of organizations research for software engineering [2], each organization is linked to one or more users by means of the class `Membership`. Details provided by the `Membership` class include:

- A free description of the user's role. For example, typical roles within a corporation are *developer*, *software engineer* or *CEO*, while some of the most common roles within an education institution include *student*, *teacher* and *assistant*.
- The status of the membership, which is used, for example, to specify that the user is no longer active within the organization.
- The date in which the user was recognized as a member of the organization.
- The date in which the user left the organization.

Similarly, the relationship between an organization and its projects is implemented through a class `ProjectInvolvement`, which is directly mapped on a collection of the underlying database and which provides data such as:

[9] available here: http://sourceforge.net/api/trove/index/rdf

- The organization's involvement type. In particular, two values are allowed: *cooperation*, meaning the organization plays a key role in the project development, or *participation*, in case the organization only provides some support, for example developing some portions of code, while core decisions are taken by the remaining involved actors. This allows to understand whether an organization has a primary or secondary role in the development of a project hosted on the forge.
- The status of the organization's involvement.
- The date in which the organization started to work on the project.
- The date in which the organization stopped to collaborate to the project, if applicable.

3.2 | TITANS Community Typing Metrics

In essence, as part of our ethnographic study, we set out to define ways in which relevant community aspects can be measured, either by means of TITANS or by means of field-observation. The rest of this section elaborates TITANS community metrics framework, outlining the metrics that take part to the framework arranged according to the community type they reflect and help identify. All metrics were inferred using the Goal-Question-Metrics approach [26]. The section concludes presenting a sample demonstration of these metrics' representation condition[10] following specifications in [27,28].

3.2.1 | FN: Formality, Hierarchisation Degree, Membership Selectivity

First, we found that *Formality* can be determined through TITANS by calculating the ratio between milestones assigned to the project and lifetime of the project itself. The more milestones are assigned to a shorter lifespan of the project, the more formal the network is. TITANS allow this calculation through a combined effort by the **MetrOrg** and **MetroForge** components. Both components allow to gather and keep track of how many members commit to a project as well as how many milestones have been set through the lifespan of the project (i.e., from first to last commit).

Second, a high formality can be confirmed by observing a high *Hierarchisation Degree*. This can be estimated by computing the ratio between number of user groups and different set of permissions. TITANS makes this possible by keeping track of relevant statistics, and performing computations.

Finally, another attribute of FNs is represented by the rigour adopted to select the members of the community. This attribute is referred to as *Membership Selectivity*.

To understand the selectivity of membership, it is necessary to verify wether non-members are allowed to access the project and actively participate in its evolution by committing contributions uncontrollably. If the project is private, then a selection process is adopted to filter contributing members. Also, governance rules are applied to select contributions that are deemed appropriate for the community's aims, as is for some projects in GitHub, e.g., Ruby[11]. However, if a project is ungoverned, TITANS allows us to estimate membership selectivity as well as milestones/day ratio by means of the **UProfile**, **UOrg** and **MetroForge** components. These components realise three critical functions: first, **UProfile** and **UOrg** are responsible for activity-tracking of members, organizations and project milestones; second, **MetroForge** is responsible for assessing if inactive developers or organizations can contribute/are contributing to the project. If so, there is no *Membership Selectivity*.

Formality:
$$\frac{\sum m}{\sum LT};$$

where "m" is the number of milestones in the project and LT is the lifetime of the project itself (both are expressed as a number of days); In essence, the higher the ratio, the higher the formality;

Hierarchisation Degree:
$$\frac{\sum Ug}{\sum Up};$$

where "Ug" is the number of user groups and "Up" is the number of user permission levels; in essence, the higher the ratio, the higher the hierarchisation degree;

[10] Remaining metrics' demonstration are omitted for the sake of space and are available upon request.
[11] http://edgeguides.rubyonrails.org/contributing_to_ruby_on_rails.html

Membership Selectivity:

$$Uc - EUP;$$

Where “Uc" is the number of uncontrolled contributions allowed to non-members and EUP is the number of External User Privileges (e.g., browsing the code, browsing the issues, etc.); in essence, the lower the number the lower the higher the selectivity;

FN Thresholds: Based on what we observed in our samples for Allura, the *Hierarchisation Degree* is high when the number of permission groups exceeds 2 and an average of 80% of the users of the same organization are concentrated within a single permission group. Also, *Formality* is high if the milestones per day ratio is equal to or exceeds 10%. Finally, the metric *“Membership Selectivity”* In the case of Allura is high since members are only appointed if subject to a rigorous contribution scrutiny - there are no uncontrolled contributions for external members although there are some EUPs (e.g., browsing and downloading code, inspecting contributors, etc.). Therefore, Allura is membership-selective.

3.2.2 | INs: Previous Relations, Community Openness, Lack of Governance

First, informality is the result of stronger social ties among members. To establish whether there is a relationship between two members we looked for a continuative collaboration involving both of them across different projects, i.e., *Previous Relations*. The metric depends on the frequent interactions between two developers. This results mainly from a steady relationship between said developers. Therefore, the number of members that have a significant number of previous collaborations suggests the level of informality within a development community. Thanks to the **UProfile** component, TITANS allows tracking of members collaboration history. In the same way, TITANS compute the frequency of interaction between developers that previously collaborated together. Another key attribute of INs is its degree of *Openness*.

The metric allowing to understand the degree of openness of a community is represented by the level of permissions granted to non-members. If external developers are allowed to directly change the code of the project, for example, it means that the community is very open; on the other hand, communities which deny external users messaging permissions, are very closed closed communities (not necessarily closed-source, however).

The openness degree of the community is therefore represented by the ratio between permission level granted to external users as opposed to the levels granted to internal users (“Read”, “Create”, “Update” or “Admin”). TITANS played minor role in community Openness. TITANS’ role was limited to averaging the number of non-members that can freely “Create”, “Update”. This figure offers the basis for estimating community openness.

Third, INs are also characterised by the *lack of governance* practices. This means that there is no strict policy to control the development process. We observed that the reduced use of milestones is one of the indicators of low governance circumstances. Therefore, the metric represented by the ratio between the number of milestones and the life time of the project can be considered to evaluate governance “strength” across the community. This metric is computed as the inverse of a metric computed for FNs, that is, the numbers of milestones per project lifetime ratio, as discussed previously.

Previous Relations:

$$\frac{\sum c(a,b)}{T(a,b)};$$

where “c(a,b)" is the number of projects on which contributors “a" and “b" contributed together while T(a,b) is the total number of possible couples of contributors in the community;

Community Openness:

$$\sum EPL - \sum IPL;$$

where “EPL" is the number of permission levels granted to external users while “IPL" is the number of permission levels granted to new users who have not accumulated commit experience yet (i.e., they just joined); in essence, the lower the number, the higher the community openness;

Lack of Governance:

$$\frac{1}{Formality};$$

INs Thresholds: Analysing probe samples, we found that *“Previous Relations”* can be established if social ties within a community are not loose, i.e., when the average number of projects to which each pair of community members collaborated together exceeds 1. Based on what we observed in our experience with Allura, this metric is therefore binary (i.e., either it does exceed 1 or it does not). Moreover, The *“Openness”* degree is high if the permission levels assigned to non-members is different from “Read” (based on policies we observed in Allura). According to our experience, there is no project granting “Admin” permissions to non-members, since this would violate basic security policies. Moreover, external developers do not need this permission to fully participate in the project. This essentially means that EPL and IPL values for Allura are identical meaning the

distance between the two values is 0. Finally, *"Lack of Governance"* can be ascertained using the same threshold previously defined for Formality in Formal Networks: an organization has the attribute "disorganisation" when it has a very low number of milestones per project lifetime. In our case, we observed that within Allura the relevant threshold is fixed at an average of no more than 0.02M per day.

3.2.3 | NoPs: Geo-dispersion, Self-Organization, Self-Similarity

First, a key attribute for a NoP is *Dispersion* among developers. We evaluated dispersion within a community by considering geo-cultural distance as a metric. Geo-cultural distance refers to the geographical distance (i.e., distance in time and space) between developers, multiplied by the cultural differences ratio among members of the community (i.e., the deviation average between the fixed Hofstede indexes for each pair of community members). TITANS allow each user to state his or her country and city of residence, as well as the time zone in which the user is located. This data allows us to estimate the distance separating members co-working in a community. The average physical distance within a community is obtained by considering the geographic distance between the coordinates of the city of reference, for developers in the same time-zone. These coordinates are defined by the IANA, which was adopted by TITANS as the standard set of time zones among which developers are allowed to choose. A large physical/time distance between members means that there is a high dispersion among them. In some cases, however, a limited physical distance among the members of a community is counterbalanced by very relevant cultural differences, which could have an equally important impact on the results of the community's efforts. To cover these cases, we allow the estimation of the cultural distance considering the indexes defined by Hofstede in a study on cultural dimensions conducted for IBM between 1967 and 1973 [29]. To support the identification of dispersion, TITANS compute Hofstede metrics for countries in a project and uses it in combination with time/space distance to determine geodispersion as defined in [23].

Cultural distance across a community can be estimated by computing the deviation average between (fixed) Hofstede indexes for each pair of community members. The maximum empirical threshold for this computation was elaborated by Hofstede himself as 40% [29]. This figure means that there is a maximum cultural deviation average possible across any community, this average is 40%. TITANS implement the Hofstede index and threshold values and use them in combination with countries reported by community members to evaluate cultural distance for communities using such a deviation average.

Second, to characterise a NoP *self-organization* of the community should be measured. In a self-organising community, collaborating members freely decide how to organise their work, schedule and progress estimation. We observed that averaging the number of tasks that are allocated to contributors (of any kind) by others givers a reliable measure of self-organization.

Third, finally, a NoP is characterised by a high *self-similarity*, resulting in communities whose members share common skills and interests. Self-similarity can be evaluated by evaluating the average number of shared topics of interest and skills between member couples (i.e., expertise overlap as a measure of cognitive distance [30,31]).

Finally, NoPs also show a number of automatically computable second-class attributes (i.e., found only on some their instances) such as *community openness* that we discussed already or *community size*. NoPs are usually groups including a high number of people, therefore the attribute *size* was considered as the last relevant attribute for a Network of Practice. This attribute can be simply evaluated through a metric expressing the number of project members.

Geo-dispersion:

$$\frac{d}{(\delta(a,b))};$$

where "d" is the geographical distance between contributors "a" and "b" while "δ (a,b)" is the average deviation between the the deviation average between the fixed Hofstede indexes for each pair of community members; in essence, the higher the number the higher the geo-dispersion

Self-Organization:

$$\frac{Ta+Tt}{Tt};$$

where "Ta" is the number of tasks assigned to contributors by other contributors while "Tt" is the total number of tasks currently driving the community; in essence, the closer the number to zero, the higher the self-organization across the community;

Self-Similarity:

$$\frac{\sum Eo(a,b)}{M};$$

Where "Eo" is the expertise overlap between each pair of contributors while M is the total number of contributors in the community; in essence, the higher the number the higher is community self-similarity;

NoP Thresholds: Based on our ethnographical considerations taken among Allura participants, it results that it is a highly-dispersed community, with an average distance between committers of 4926 Km and a standard deviation of about 3199.5 Km. High dispersion was corroborated by the Allura community itself. In general, based on these reference observations, a highly dispersed a community has an average distance among its members exceeding 4000 Km. In addition, by computing cultural distance for Allura committers, we obtained a value of approximately 15.4%, with a standard deviation of 13%. Having experienced ourselves the high cultural difference in Allura, confirmed by a high geo-dispersion, we fixed at 15% the threshold for deviation averages above which a community is highly geo-dispersed (i.e., both in space and culture). Moreover, In our experience from Allura, we found that a low degree of hierarchisation averaged together with level of formality are indicative of high *self-organization*. We found a threshold value for this metric as 3.69 in our observations for Allura. Finally, to be considered *self-similar*, a community needs to show a very high (i.e., 90%) percentage of members with at least one shared skill.

3.2.4 | ICs: Engagement

To evaluate members' *Engagement*, we used statistics for members' participation levels. We found that combining per-month number of posts and responses, allows to measure the involvement of the community in discussions. The metric also considers the number of active members, so that average individual efforts are computed. Unique commenters of the community are members with a higher number of contributions related to the one of the remaining members. Therefore, unique commenters are a group of supporters who significantly contribute pursuing the success of the project and its popularity outside the community itself, and they can be identified by comparing the amount of contributions they submitted to the total number of contributions. Within TITANS, the same computation can be applied to organizations instead of single users. This allows us to identify those organizations which have greater visibility and engagement within and outside of the community.

Moreover, *Engagement* is further characterised by observing the number of comments belonging to a thread, as well as the average per-month number of messages posted within the thread, allows us to evaluate the interest gathered by the discussions related to the project. Considering the breadth of each discussion thread is also useful to understand the engagement of the community, since it shows whether the discussion is alive or not, and it indicates the thread impact on the members of the community.

The amount of threads that generate from a post further contribute to understanding the involvement of community members and the value they recognise in a discussion thread.

Finally, contributors may show interest in a discussion simply by subscribing to the discussion itself. In that case, members do not directly contribute to the thread, but their subscription indicates an interest in being in the know. Therefore, the number of subscriptions to feeds and events related to a discussion can also be considered as an indicator to evaluate the engagement within the community.

TITANS aid the identification of a high engagement within the community by making it possible to compute the average of the above measurements for observable communities, such that *Engagement* can be established. All extensions we provided are required to interact together to compute engagement. More in particular, the **UProfile** and **UOrg**, allow users to store their own details about people and organizations that contribute in the forge. These can later be used in combination with statistics from **MetroForge** to measure, e.g., how many threads were initiated by whom across the development network. The **MetrOrg** component performs a similar function for organizations.

Engagement:

$$(\sum Pmq + Pmr) + \frac{Pc}{Tc} + Gtl + Sn;$$

Where "Pmq" is the number of per-month questions for each member, "Pmr" is the number of Per-month responses for each member, "Pc" and "Tc" are the number personal contributions (Pc) per member against the total number of contributions (Tc), "Gtl" is the generated topic liveliness, or better, the number of messages in a thread initiated by every contributor, while Sn is the number of subscribers to threads generated by each member;

ICs Thresholds: Averaging the factors described above, we found that engagement within a community is high if each member posts no less than 30 comments per-month. Also, a member is considered to be a unique commenter if he or she is responsible for 30% of contributions or more. In case there are at least a user and an organization which are unique commenters, engagement within the community can be considered significant. Finally, the size of a discussion thread is considered high if, on average, each thread has at least 3.5 comments, or there are at least 0.1 comments per month for each thread, on average. Given our experience, the average number of discussion spreading from a thread post is high if it exceeds 0.5. Finally, in average, we observed that the number of discussions, comments or threads spreading from a thread or discussion is comprised between 0 or 1.

4 | EVALUATION

Our evaluation efforts rotate around evaluating TITANS from three perspectives: (1) usage; (2) usefulness; (3) technical generalisability.

The first, most intuitive evaluation of our results in action is reflected by the constant community-based evaluation part of our action research strategy (see Sec. 4.1). *This type of evaluation addresses the usage of TITANS extensions*.

Second, our contributions were evaluated in action by using TITANS extensions in practice on the 4 top open-source projects hosted on SourceForge at the time our extensions were included in the Allura baseline. *This type of evaluation accounts for TITANS usefulness in detecting community types and valuable statistics which would otherwise remain latent (see Sec. 4.2)*.

Finally, In order to evaluate the generalisability of our technical contributions we report on the evaluation by replication of the TITANS metrics and community awareness by design in another well-known open-source forge which already provides community-awareness by-design, namely, GitHub.

4.1 | An Evaluation of TITANS Within The Allura Community: Experiences Report

As an essential and continuous ingredient of our action research approach, the first key evaluation of the developed work was conducted by the Allura community itself, which constantly analysed the committed proposals and inspected the submitted code, evaluating its quality and fitness to uphold the goals pursued by TITANS. All throughout these phases, we took notes and records of the evaluation process. The results of said process can be summarised as follows. First, by accepting the proposed code, the reviewers, on behalf of the whole community, considered our contributions as extensions fit to upgrade Allura with good-quality code implementing needed functionalities. This was further confirmed during various discussion upon acceptance of TITANS extensions. This verification and validation process harnessed a many-eyes phenomenon that allowed us to produce good-quality extensions. This process however, was far from simple. For example, during the discussion following the development of the tool extending user's personal data, the reviewers initially praised the idea of the tool, considering it as "very promising" and claiming to be "very pleasantly surprised, actually, how far you've come without needing to ask for help or pointers", but also suggested improvements due to a lack of initial adherence to some standards adopted by the community to develop the software. Most notably, we were required to adopt EasyWidgets for HTML markup and input validation, and to use JQuery instead of plain JavaScript. We were also required to write additional tests for our software, and to improve the Python syntax by removing some blank spaces preceding symbols like ":" or by avoiding to write the single-instruction body of conditional statements on the same line of the condition. Second, after committing the required fixes, subsequent work was considered "excellent" and "very nice" by the reviewers. These improvements required four iterations, and the code was then merged into the main repository[12]. This also led the first of us to be granted committer rights on the Allura repository. Shortly after the acceptance of the first tool, we submitted the code related to statistics of single users. In particular, the code was submitted within a branch of the original repository. The review process started, but it was delayed due to concurrent reviews involving the same developers. The process consisted of six iterations, during which some improvements were proposed by the community. Most notably, the implemented functionalities, initially developed as a separate feature, were included as an additional tool within the project related to a single user.

In addition, these changes also led to the introduction of other minor modifications and triggered a discussion that involved several developers in order to identify a way to automatically install the newly created tool on the projects related to previously existing users. The discussion led to the decision to set it as an *anchored tool*, a tool which is automatically installed when a user accesses a project which does not include it yet. During the review process, some errors in provided tests, related to missing updates following the developed modifications, were also identified and fixed. Another required change was related to the registration date, which was initially considered to be equal to the date of creation of the object including user statistics, and later became exactly equal to the user's registration date on the forge.

Finally, the code related to organizations were submitted and accepted. Similarly, the code allowing to gather statistics related to single organizations was accepted for the Forge support system.

A positive evaluation of the software emerges from its real exploitation. In fact, those proposed tools which have been accepted by the community are now installed on SourceForge.net, making them available to a worldwide audience, by means of one of the most popular software forges in the world, hosting more than 300,000 projects with cumulative daily downloads exceeding 4,000,000 units. By using our software, GeekNet Media, the Dice Holdings Inc. company owning SourceForge.net, certified the industrial value of the developed software, using it for its commercial purposes. In the words of Dave Brondsema, chief core-committer for Allura, TITANS was evaluated as follows: "We were very happy to have these contributions. [...] We didn't [feel we needed to] do any user testing, just our own bucket testing before merging and releasing the code. The user profile enhancements were useful [...] and used extensively in the following months". Also, GeekNet Media people provided us with the following statement: *"contributions like the ones we have provided represent the main reason why a company releases the code adopted for its commercial purposes: our tools allowed Allura to evolve by including innovative contributions developed by external contributors, with limited additional costs"*. These extensions are currently active in selected forge areas, but disabled in the public forge for scalability reasons.

From a quantitative perspective, the community at Allura provided the following utilisation stats for TITANS users within Allura:

[12] http://sourceforge.net/p/allura/git/merge-requests/7/

1. 1409 users have set their skills after TITANS extensions were activated. These users are currently browsing projects using skills as well;
2. 854 users have set social networks and relate to others through stubs there provided;
3. 7070 users have set timezone and reportedly adjusted communication based on this variable;
4. 7356 users have set sex;
5. 1375 users have set webpages to increase their own community visibility;
6. 11 core developers have reportedly set availability to ease reachability;
7. 328 users have reportedly set phone numbers to ease reachability;

These numbers suggest that forge users chose deliberately to enhance their profiles leveraging on TITANS. This result is encouraging and shows that TITANS is a promising set of extensions towards a number of mutual and community awareness aspects, e.g., enhancing the visibility of community members' across big open-source communities such as Allura. Although these numbers do not offer a frame of reference for comparison, they do represent a series of valuable statistics to reflect on TITANS' extensions usage and the extent to which they penetrated software development processes at large in the forge.

Evaluation Result: we conclude that TITANS successfully extended Apache Allura with valuable and usable community characteristics which form a basis for community awareness by design.

4.2 | Community Analysis in Action: Experiences from Four Projects

In order to evaluate TITANS as applied to compute the metrics introduced in Section 3.2 we applied said extensions and metrics on four widely known "Projects of the Month" communities supported by Allura, on the SourceForge open-source forge, namely: (a) ProjectLibre[13]; (b) Kiwix[14]; (c) DOSBox[15]; (d) JStock[16]. More specifically, these four projects were selected out of the Allura community-elected projects-of-the-month by ranking all said projects using their #commits, #users, and #downloads and selecting the top four.

To proceed with our validation, we studied all freely available material describing the aforementioned projects and carried an *in-sylico* observational study of their activity over a period of 3 months, logging daily such activity in lab journals. Our goal in this study was to formulate a community type hypothesis for said communities based on the taxonomy and decision-tree proposed in our previous work [2,23,19]. Our hypotheses for community types of selected projects are summarised as follows:

- **ProjectLibre**: an open-source project management software. The project engages a community of about 10 000 users worldwide, with over 8000+ downloads per week. Studying the community around ProjectLibre without using TITANS we hypothesised ProjectLibre to be a *Network of Practice*. Indeed, the sheer size and dispersion of the community (multiple charters in 7 continents worldwide) suggests the community around ProjectLibre to be a *Network of Practice*, rotating around the practice of project management and evolving support for this activity. However, the hybrid formal/informal nature of work (e.g. rotating around milestones, blog-posts and feature requests alike) inside ProjectLibre is also consistent with a hybrid type, fusing together *Formal Networks* and *Networks of Practice*.
- **Kiwix**: an offline reader of web content, especially developed for offline usage of wikipedia. Kiwix engages a similarly sized community as Project Libre, with 3000+ DLs per week. The informal nature of governance policies across Kiwix suggests an *Informal Community* or *Informal Network*. However, the absence of an explicit organizational sponsor and increased dependency on members' engagement strongly suggests Kiwix to be an *Informal Community*, according to type definitions in [2].
- **DOSBox**: a DOS emulator available on a large number of modern operating systems. The project sees an extremely active user-groups community, engaging over 150 000 users worldwide, with an average weekly DLs of 40 000+. The very limited size of the core-team behind DOSBox suggests a community much similar to a tightly knit *Project Team* or *Workgroup* according to the classification in [23]. However, the frequent interaction with a steady community of global users who request and validate functionalities, suggests a strong dependency on informal communication and shared practice by members in the community: these characteristics led us to consider DOSBox consistent with *Informal Networks* or *Networks of Practice*.

[13] http://www.projectlibre.org
[14] http://www.kiwix.org/wiki/Main_Page
[15] http://www.dosbox.com
[16] http://jstock.sourceforge.net

TABLE 2 Sample for TITANS metrics and community types prediction: thresholds in bold.

Attr.	Metric	Allura	Proj.Libre	Kiwix	DOSBox	JStock
Engag.	Active Members	5	3	1	**11**	1
	# Commnts./Active Members	**136.46**	12.50	**33.25**	2.02	**31.95**
	Unique Commenters	0	0	**1**	0	**1**
	Monthly Comments/Thread	0.112	**0.280**	0.030	0.030	0.045
Formality	Milestones/Day	**0.0101**	**0.0306**	0.0012	0.0048	0.0005
Self-Org.	#Comments/Formality	**3.69**	1.83	2.43	**3.98**	3.08
	Comm. Type	*FN*	*FN*	*IC*	*˜ NoP*	*IC*

- **JStock**: a software supporting users in tracking stock investments. JStock has a smaller but very domain-specific community, engaging a few thousand users in 26 countries, with an average weekly DLs of 2000+. The community around JStock features many organizational bodies with more or less formal involvement within the project. However, very little data concerning the community around JStock was available. This suggests the community to be small, well organised and governed as a "closed" body, controlled with formal guidelines and supported with donations-based systems other than organizational sponsors. These indications seem to suggest a *Formal Network*, even though the little data available did not allow us to be confident on our hypothesis.

Our evaluation conjecture was that TITANS and metrics would successfully infer a community type if the resulting metrics would match our typing hypothesis.

To obtain such checking of the typing hypothesis we operated via ethnomethodological research [32] against the set of sample measurements performed by the TITANS extensions.

First, the set of sample results for TITANS measurements are contained in Table 2. In essence, the table contains a sample set of measurements operated by TITANS - these measurements are then matched with the type automatically inferred by TITANS (last row). Column 1 shows the relevant attribute being observed. Column 2 shows related metrics. The last row on the table contains the community type inferred by TITANS. Remaining metrics are not shown for the sake of space. TITANS measurements led to the following type predictions.

First, ProjectLibre has a structure featuring significantly higher formality levels with respect to the remaining ones. In particular, ProjectLibre, quite a recent project, has 0.03 milestones per day, meaning that, on average, a new milestone is created every 33 days. This appears to be a significantly high frequency of milestones delivery compared to the other projects. Thus, contrarily to our initial hypothesis, ProjectLibre is a (highly dispersed) *Formal Network*, much like Allura itself. Second, Kiwix is indeed an *Informal Community*, given the high level of engagement within. Third, DOSBox, given its high self-organization value, indicates a potential *Network of Practice*, perhaps in the primordial stages due to the low number of participants. In this case, certain community typing was impossible. We did not have enough data to compute geo-dispersion and self-similarity since DOSBox was not using TITANS **UProfile** and **UOrg** extensions. Finally, JStock is an *Informal Community*, again, due to its high levels of members' engagement.

To confirm these measurements the corresponding typing, two students and a senior researcher previously involved with this study embedded within the targeted communities and observed the organisational process enacted within them for a period of 45 days, across which, weekly notes were taken concerning the organisational activity, and discussed with the authors of the paper. The aim of the discussion was to confirm/disprove the typing performed by TITANS without such authors knowing this typing. Indeed a total overlap was reported between the typing associated by the authors and the typing associated by TITANS extensions.

Evaluation Result: we conclude that TITANS successfully provides a framework to infer community types and characteristics from known metrics and statistics.

5 | DISCUSSION AND LESSONS LEARNED

This section elaborates on discussion points, lessons learned, and threats to validity.

5.1 | Discussion

Stemming from our work, several discussion points emerged.

1. **The TITANS extensions, and the principles with which they were engineered offer a valuable example of re-architecting software forges, which procures unique limitations and design constraints**. For example, during TITANS design, we had to assume and design for several performance limitations of Allura itself and its underlying operational infrastructure. As previously remarked, Allura core committers required us to design TITANS extensions so as to make them an optional bundle since the underlying Allura infrastructure may not be able to sustain the load of TITANS metrics. On one hand, this limitation of TITANS may reflect an architectural flaw of Allura from a deployment and operational perspective. On the other hand, TITANS extensions were actually deactivated due to performance overheads. This leads to conjecture that designing (or re-designing) to support community governance needs to be done contiguously with operational and infrastructure concerns, since a trade-off needs to ensue between meta-forge operations and meta-forge intelligence.

2. **Several areas of the general open-source community at large**[17] **openly criticized the organizational and architectural directions of the Allura forge just after TITANS extensions were rolled out of the Allura baseline**. From these anecdotal evidence, we can argue that, on the one hand, TITANS extensions may have given Allura a heads-up warning as to their eventual fate but, on the other hand, a misguided performance-focused architectural and organizational decision eventually led to Allura's fall from grace. This insight should be further investigated to understand the role of social and organizational aspects in software community failure scenarios such as Allura.

3. **Open-source community governance is far from trivial in large projects and involves following multiple community patterns blending their characteristics together - community-specific organizational and socio-technical metrics may be needed**. A degree of uncertainty we reported across the Allura community concerning socio-technical and organisational decisions suggests that mechanisms such as those exploited in[33] or[23] and partially implemented in TITANS, could be used in combination with our metrics and social community types to determine effective community steering. On the one hand, more research is needed to characterise with reasonable certainty observable community types, beyond the simplistic experiments recapped in this paper. On the other hand more research is needed to develop and validate governance practices tailored to certain community types and their typical characteristics. This research track could start on top of our. For example, our measurements and the community types associated to them can be used to evaluate whether organizational decisions are driving the community more consistently to its type. This evaluation becomes increasingly important when the community changes type altogether, e.g. as a consequence of outsourcing, offshoring or similar networked development decision.

4. According to the Apache policies, when developers' involvement within the project becomes significant and the submitted code is considered a qualitatively and quantitatively significant contribution, these become **core committers of the project - we observed however that this status rarely coincides with higher organizational or governance output**. This process is not new and is adopted by the community to elect new committers based on proposals made by other core committers. If properly enriched and tool-supported, e.g., by means of TITANS extensions, these processes would offer a valuable basis to self-examine and steer community governance, e.g., to make the community more attractive from the external environment or making its organizational and socio-technical processes known and more successful, rather than more transparent. On one hand, these processes could be used in closed-source as well, e.g., to create more specialised and effective communities of developers. On the other hand, the same process should be further studied and enriched with an organizational self-assessment and evolution part which is missing and yet may prove extremely beneficial for community success. In general, traditional team formation in software engineering (open-, closed-source or otherwise) could benefit from further research on such specific practices and policies.

5. **Open-source communities pose particular attention to keeping track of developers' reputation across the development network**. This and similar mechanisms increase the level of confidence and engagement in community members. Based on our experience, closed-source does little to adopt such mechanisms, e.g. steering teams based on employees' reputation. We found that there is still much to learn from studying open-source communities in this regard.

6. **Public software forges value speed and performance over measurement and governance**. Computing metrics across large open-source forges, although useful for many governance-related reasons, revealed itself a costly endeavour, to be balanced out with the convenience return on performance invested. As a consequence, our **MetrOrg** and **MetroForge** extensions, while present in Allura, were deactivated in SourceForge given their performance requirements. Striking a balance between needed and effective measurements to steer open-source communities is still an open research question that deserves further attention.

[17] for example see here: https://news.ycombinator.com/item?id=6262347

5.2 | Lessons Learned

In the scope of this experimentation and the discussion of the results we learned four key lessons, potential design principles to be reapplied in other software forges or concurrent programming environments.

Community Activity-Tracking Aids Awareness. All reported evidence discussed previously leads to conclude that members awareness and their engagement in the practice of the community is increased. Even though we cannot currently show measurements for such an increase—an endeavour we aim to compendiate in the future—we learned that providing for community-tracking is beneficial towards encouraging the engagement on the practices across the community. Specifically, even simply the statistics provided by the Allura people (see Sec. 4.1) reflect engagement of open-source developers into completing their profile on the platform, and this in turn remarks their engagement and a mutual positive effect on their peers derived from a better elaborated developer profile.

Architecting the Forge is a Participatory Process. Like any type of software, the software that supports software design, maintenance, and evolution—such as the forge that we have studied in the context of this research—needs to engage more actively on the public around its usage and therefore become a participatory and more social process. On the one hand, this is evidenced by the critics moved to the Allura community after certain modifications to it were applied. On the other hand, to the best of our knowledge, not even GitHub has a fully inclusive process behind the maintenance and evolution of its inner workings; we are confident our findings may help in redesigning the forge more in line with this principle.

Community Support Needs to address Governance. Our findings and discussions—specifically the user feedback that led Allura to deactivate the TITANS extensions or the Allura focus on its perceived reputation by its users—seem to indicate that support for the governance and community-tracking aspects behind forges is still rather scarse. Conversely, such mechanisms would be needed to keep track of essential community characteristics, avoiding nasty phenomena such as the emergence of so-called *community smells* [25,34] or even more traumatic community-forking events[18].

Governance trades-off with Performance. Although the previous principle points out clearly the need for automated governance support, our experience suggests that performance needs around the forge operations surpasses the necessities for governance. Although this seems to be the classical innuendo between short-term gains—in terms of performance—and long-term benefits—stemming from governance—one conclusion is clear: an architecture trade-off analysis process needs to be enacted every time such a governance-related automated support is embedded into the operations of the forge since such an addition might influence the software-related operations but also the communication and coordination activities across those operations, which are equally, if not even majorly important.

5.3 | Threats to Validity

We found at least two validity threat areas, i.e., external and internal [35].

5.3.1 | Threats to external validity

These threats concern the applicability of the results in a more general context. TITANS are intrinsically linked to a single context, i.e. that of Allura. Also, their design and the requirements they were designed to pursue, were obtained from empirical research using a community of 50+ inquirers, whose responses however, we noticed to lean towards academics who contribute in open-source. These threats limit and bias the nature of our action research to the opinions of the people involved. We already increased external validity of TITANS extensions by exercising them in alternative open-source forges such as GitHub though a prototype that served also to demonstrate the representation condition of said metrics. Moreover, from a design perspective, we made sure that our development was incrementally validated by using constant feedback and evaluation sessions by diversified forge contributors in Allura. These contributors included companies, and members of other open-source “institutes" such as the Apache Foundation[19]. Although validating the metrics we have introduced is something we plan for the future, we do argue that they represent a first attempt at characterising *community awareness* within open-source forges, allowing their members to know their own community more deeply and contribute with a higher awareness as a result. Moreover, as previously stated, our results were driven by an initial list of requirements gathered empirically by an online survey. This can compromise the initial list of requirements, biasing it towards the perception of the observers. Also, the action research was driven by constant feedback by the community. This constant validation however, suffers from the bias intrinsic to many-eyes phenomena. Finally, we modified our action research approach to integrate the constant community support, in a manner compatible with the community’s contribution policies. Hence, our metrics and reference thresholds, as well as discovered community types and characteristics were double-checked with core Allura contributors before implementation but still lack further objective evaluation beyond the work reported in this article. At the same time, we understand that the targets and our own analysis for the survey results may have biased the extent to which specific requirements have been pointed out in the scope of the TITANS extensions. For example, we aimed to provide support towards user-profiling,

[18] https://www.pingdom.com/blog/10-interesting-open-source-software-forks-and-why-they-happened/
[19] https://sourceforge.net/blog/apache-allura-becomes-top-level-project/

which itself could be critical or even risky to support gender-related aspects and similar tool support. We recognise these limitations and argue that future work should look further into such interplays stemming from our work.

5.3.2 | Threats to Construct Validity

In the scope of construct validity, we recognise several key shortcomings. For example, the way in which the community typing was applied is elaborated fully in our previous work [36] but not validated in the scope of Allura. While self-containing all necessary details in this manuscript would have made it impractical, the generalization of the same procedure outside of its validation boundaries might constitute a threat. Furthermore, any biases introduced by our own analysis as well as the selected lens or scope, itself constitutes cause to threaten construct validity. In the future, we plan to strengthen such validity shortcomings by automating community type detection and conducting more empirical experimentation around it.

5.3.3 | Threats to internal validity

Threats in this category concern the generalisability of the methods used to study and analyse data (e.g. the types of bias involved). We used goal-question-metrics (GQM) to structure the process of devising TITANS measurements for quantifiable and observable attributes during ethnography - although we proved the metrics' correctness by evaluating their representation condition, our approach and metrics framework currently offers no guarantee for the metrics' completeness. We are planning further study of these aspects and shortcomings in the future, to address this limitation. From a more methodological and analysis perspective, we however also report the possibility that some of our requirement exercises might have suffered from observer bias. On the one hand, the requirements emerged directly from survey respondents and the authors or any research connected to this work did not apply any interpretation on top of the mere content and frequency analysis of the survey responses. Conversely, we cannot exclude that in the process of providing for the TITANS extensions an implicit and subconscious prerogative was given to specific features - this remains a limitation of this study.

6 | RELATED WORK

This section discusses related literature highlighting the importance of supporting organisational and social community aspects in open-source [37].

Also, works that aim at profiling OSS software and surrounding communities is definitely related to TITANS. For example, works such as Ayala et al. [12] or Samoladas et al. [14] aim at eliciting such a profile to make sure that it is actually in line with envisioned expectations during software planning or evolution. Although these works offer profiling mechanisms from multiple perspectives, they elaborate only loosely on key OSS community aspects such as engagement or formality. We have observed that such aspects are critical to make informed decisions concerning OSS and OSS-community adoption. In support of these and similar aspects, we devised TITANS as a series of critical technical extensions that can support open-source communities from a more organizational governance perspective and using automated means. TITANS does not wish to support software evolution per se; rather, TITANS is meant as a tool that aids software communities in successfully making explicit and steering their organizational and socio-technical dynamics.

In line with the idea of making explicit the structure and type of software development communities for their better steering, works in Siemens by Joblin et al. [38] and even earlier work by Bohwmik [39] - these works are definitely related to the work outlined in this paper. Joblin et al., worked extensively from a quantitative perspective to infer community structure in a number of open-source project, proposing a tool called CodeFace. Their goal, however, is to learn from said structure and related aspects rather than instrumenting forges in making organisational and social structure more explicit. TITANS aims at filling this gap by making explicit the organisational and social characteristics which are key, e.g., to aid the identification of the types of governance and organization across the community. This information is useful for free-roaming contributors that may be willing to commit to the open-source community. In our approach we gathered requirements to address these and similar aspects as Scacchi et Al. [40]. Also, in our approach we strived to deliver a powerful organizational tool in support of socio-technical and organizational community aspects in an automated fashion. Our assumption behind TITANS is that automated tools that keep track of key organizational and socio-technical metrics are essential to make explicit and support community awareness aspects in open-source.

On the basis of the work by Joblin et al. [38], empirical investigations have been conducted with the aim of enlarging the scope of CodeFace. This is, for instance, the case of CodeFace4Smells [41], which builds on top of the collaboration and communication networks identified by CodeFace in order to identify instances of the so-called *community smells*, i.e., a set of sub-optimal conditions that may induce and/or increase social debt [42]. Tamburri et al. [41] have experimented with CodeFace4Smells in the context of an empirical study aiming at assessing the diffuseness of community smells in open-source projects, understanding the developer's perceived harmfulness of the sub-optimal conditions identified by the tool, and identify correlations between a set of known socio-technical metrics (e.g., the socio-technical congruence [43]) and the emergence of community smells. Later on, the correlations identified were also exploited by Palomba and Tamburri [44] to devise an automated mechanism, based on machine learning,

able to predict the emergence of community smells. Along this direction, Huang et al.[45] exploited sentiment analysis to boost the community smell predictive capabilities, while Almarini et al.[46] devised a genetic programming-inspired mechanism for detecting community smell instances.

More research on community smells has been conducted with the aim of verifying the community characteristics leading to the emergence of socio-technical issues as well as on the impact of community smells on source code quality. Catolino et al.[47] devised a statistical modeling approach to study the effect of mixed teams on the likelihood of having community smells, showing that, in some cases, female developers can act as mediators, hence reducing the risks of damaging community health. The same authors later extended this work with a survey study involving practitioners, where they indicated that the gender-related considerations are not perceived as relevant indicators of community health[48]. More recently, Catolino et al.[49] investigated the variability of community smells over time, pointing out the existence of community-related metrics that may serve as a monitoring system for establishing community health, as well as the "refactoring" strategies applied by practitioners when mitigating the effects of community smells, proposing a catalog of best practices[50].

When considering the impact of community smells on source code quality, Palomba et al.[51] proposed a machine learning-based instrument that can estimate the harmfulness of code smells, i.e., poor implementation choices in source code[52], based on a combination of community smells, socio-technical metrics, and code quality factors. Similarly, a number of researchers have attempted to identify relations between socio-technical aspects and properties of source code like defect-proneness[53,54,55,56], change-proneness[57,58,59], continuous integration build failures[60,61], and long-term sustainability of software communities[62,63,64].

A complementary view on software health concerns with the analysis of open-source projects and their characteristics. In this respect, a vast body of knowledge has been reported and synthetized through systematic literature reviews[65,66,67,68,69]. Furthermore, Tourani et al.[70] investigated the code of conducts and its impact on the longevity of software communities, while Tamburri et al.[36] devised a tool to measure relevant characteristics to identify the underlying structure of a community. Lopez et al.[71] instead applied social network analysis to identify the main characteristics of developers working in FLOSS systems.

With respect to most of the papers discussed above on community smells, patterns, and socio-technical metrics, there are two main points that make our work different. From a conceptual point of view, TITANS represents an attempt to make software communities aware by design: this means that it does not primarily aim at identifying socio-technical problems or patterns, but rather it has the goal of measuring community characteristics and making them explicitly available. In this sense, TITANS can be seen as a more general, conceptual approach that can be used by practitioners to monitor software communities in a comprehensive manner. Perhaps more importantly, the goal of TITANS is that of supporting the work of community shepherds through the identification of the characteristics that might require actions. In other words, it can be seen as a decision support system for steering the organizational and socio-technical dynamics of software communities. Other instruments, like community smell detectors, community structure identifiers, or techniques for estimating the impact of socio-technical factors on products and processes, can be ideally complemented and fully supported by TITANS as a complementary part of the measurements it performs.

Other works related to TITANS aim at establishing which governance practices are active in which types of open-source communities, e.g., for further evaluation of said practices against the productivity or liveliness of the community itself. An example is the work by Prattico[72] who considered six communities supported by active open source foundations: Apache, Eclipse, GNOME, Plone, Python and SPI. Using computer-aided text analysis of each foundation's bylaws, Prattico noted that, although each foundation adopted different terms, it was possible to identify three common main power centers: the *members of the community*, the *board of directors* and the *chairman of the community*. For example, the chairman of the community can be named by the board of directors, as in the Eclipse foundation, or elected by the members, as in the Debian project. The board of directors is composed of people elected by the members. The board takes decisions about the piece of software it is in charge of. Also, different communities showed a different distribution of power. Prattico concludes that said aspects are organisational and social and may impact dearly on the future and well-being of the community. Nevertheless, the aspects Prattico observed were only made explicit after extensive analysis. With TITANS we aim at making these aspects explicit by direct measurement of organisational and social indicators. differently from TITANS, some tools are explicitly aimed at tackling one or more communication barriers in GSD, e.g., by allowing practitioners to talk with remote colleagues in an easy and informal way. For example, the Jazz project, sponsored by IBM, added instant messaging features to the Eclipse Platform, together with other tools that show which files are currently being edited by whom[73]. Also, a number of tools focus on supporting activities such as distributed problem analysis, requirements elicitation and activity planning. For example, the tool MasePlanner is an Eclipse plug-in with features for simple agile planning in distributed contexts. Users can create story cards shown in a common virtual workspace. Cards can be organised and modified by project members to plan their activities. These tools, however, aim at supporting organisational and social aspects in open-source by extracting information from forges. For example, tools proposed by the Libresoft group mine data extracted from code repositories, malinglists, discussions and tracking systems[74]. The SeCold portal adopts mining techniques to build a shared knowledge base about several open-source projects, including explicit facts like code content and statements, as well as implicit data, such as the adopted license and the number of clones produced from a project[75]. In similar studies, (e.g., the ALERT[20] project) authors use ontologies to increase awareness by gathering and

[20] http://www.alert-project.eu

linking related information obtained from the different tools adopted by the community, including structured and unstructured sources [76]. TITANS makes no such technological assumptions and is not centred around an ontology. Rather, TITANS supports the work of a distributed software development community by: (a) making it possible to express organisational and social characteristics of every individual; (b) using the intel behind those characteristics to infer organisational and social community aspects inherited from organisations and social networks research [2,23] by direct observation and measurement.

7 | CONCLUSIONS

This article discusses an approach to make explicit and study community awareness in open-source. We gave an answer to our main research question, namely:

RQ. *To what extent can we make a previously existing open-source forge community-aware by design?*

with several technical contributions among which: (a) four tool-extensions that implement elicited requirements into the Allura forge support system; (b) a list of key lessons learned on the principles of forge design. Also, we answered the research question with a community awareness and typing framework through which we also learned that community types represent *patterns* of organizational performance in software engineering since they correlate strongly with relevant organizational performance metrics typical in open-source. We concluded that TITANS presented us with several benefits, for example, key design patterns and lessons we learned on *Forge Design*, a very intriguing software engineering topic of which we barely scratched the surface. Moreover, the results we provide offer ways to investigate the social and organisational aspects beyond open-source success (or failure) across Allura, making it more similar to major successes such as GitHub. Using these results and TITANS extensions may reveal essential lessons and best-practices from open-source software communities hosted on SourceForge exactly as much as GitHub FLOSS communities are extensively studied. In the future we plan to further validate the mechanisms introduced in this paper both in open- and closed-source environments. Also, stemming from feedback received from the Allura community, we plan to research and validate innovative extensions focusing on three components: (a) user indexing based on skills and contribution levels; (b) "help-wanted" advertising that dynamically matches project categories with user skills and, eventually reputation tags (e.g. new users can find potentially interesting projects); (c) more granular check-pointing of contributions by members, to enhance community interaction. According to community feedback from Allura, these extensions could constitute additional forge design principles that contribute to enhancing the engagement and effectiveness across open-source communities. Moreover, we plan to further analyse SourceForge in a few months using data gathered by TITANS. Furthermore, this study is setting the base to explore community-based open-source software governance. For example, TITANS metrics and community support mechanisms developed in this paper can be further elaborated, e.g., in view of studying emerging concepts such as *social debt in software engineering communities* [42]. In the future, we plan to generalise our approach and the metrics within into additional community-centric software development environments, such as GitHub, for which we already developed a working prototype. Our goal is to use gathered metrics a basis for community maturity models that extend process-based models such as CMMI. Finally, since many research questions emerge from the topics we highlighted in this work—for example, how would other types of organizational dynamics influence the software process or product? And how does this affect open-source software development?—we aim at (1) generalising available knowledge in the state of the art through a conceptual and theoretical framework to instrument further work in this area and (2) establishing the actual usefulness of supporting the various community aspects mentioned in the paper: while some research has been conducted in this respect [36,47,51], we believe that additional analyses would be fruitful and needed.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

ACKNOWLEDGMENT

The authors would like to thank Dr. Dave Brondsema ad Dr. Roberto Galoppini for all the time and energy they invested in involving us in Allura. Furthermore, the authors would like to thank Simone Gatti and Stefano Invernizzi for performing some of the work behind TITANS and its evaluation, as well as Prof. Dr. Elisabetta Di Nitto for her invaluable aid in this work. Fabio gratefully acknowledges the help and financial support of the Swiss National Science Foundation through the SNF Project No. PZ00P2_186090 (TED). Furthermore, Damian's work is supported by the European Commission grants no. 787061 (H2020), ANITA, no. 825040 (H2020), RADON, and no. 825480 (H2020), SODALITE.

References

1. Weber Steven. *The Success of Open Source*. Harvard University Press; 2004.
2. Tamburri Damian A., Lago Patricia, Vliet Hans van. Organizational Social Structures for Software Engineering. *ACM Comput. Surv.*. 2013;46(1):3:1–3:35.
3. Riehle Dirk, Ellenberger John, Menahem Tamir, et al. Open Collaboration within Corporations Using Software Forges.. *IEEE Software*. 2009;26(2):52-58.
4. Boehm Barry W.. Some Future Software Engineering Opportunities and Challenges.. In: Nanz Sebastian, ed. *The Future of Software Engineering*, :1-32Springer; 2010. Festschrift for Bertrand Meyer on the Occasion of His 60th Birthday, November 2010.
5. Feiler Peter, Sullivan Kevin, Wallnau Kurt, et al. *Ultra-Large-Scale Systems: The Software Challenge of the Future*. Software Engineering Institute, Carnegie Mellon University; 2006.
6. Nanz Sebastian, ed.The Future of Software Engineering.Springer2011. Festschrift for Bertrand Meyer on the Occasion of His 60th Birthday, November 2010.
7. Latham Robert, Sassen Saskia, eds.*Digital formations: IT and new architectures in the global realm*. Princeton, NJ: Princeton University Press; 2005.
8. Berdou Evangelia. Managing the bazaar: Commercialization and peripheral participation in mature, community-led F/OS software projects. PhD thesisLondon School of Economics and Political Science, Department of Media and CommunicationsLondon2007.
9. Herbsleb James D., Mockus Audris, Finholt Thomas A., Grinter Rebecca E.. An empirical study of global software development: distance and speed. In: :81–90IEEE Computer Society; 2001; Washington, DC, USA.
10. Hofstede G., Hofstede G.J., Minkov M.. *Cultures and Organizations: Software of the Mind, Third Edition*. McGraw-Hill Companies,Incorporated; 2010.
11. Damian Daniela, Izquierdo Luis, Singer Janice, Kwan Irwin. Awareness in the Wild: Why Communication Breakdowns Occur.. In: :81-90IEEE; 2007.
12. Ayala Claudia P., Cruzes Daniela S., Franch Xavier, Conradi Reidar. Towards Improving OSS Products Selection - Matching Selectors and OSS Communities Perspectives.. In: Hissam Scott A., Russo Barbara, Mendonca Neto Manoel Gomes, Kon Fabio, eds. *OSS*, IFIP Advances in Information and Communication Technology, vol. 365: :244-258Springer; 2011.
13. Lopez David, Pablos Carmen, Santos Roberto. Profiling F/OSS Adoption Modes: An Interpretive Approach.. In: IFIP Advances in Information and Communication Technology, vol. 319: :354-360Springer; 2010.
14. Samoladas Ioannis, Gousios Georgios, Spinellis Diomidis, Stamelos Ioannis. The SQO-OSS Quality Model: Measurement Based Open Source Software Evaluation.. In: Russo Barbara, Damiani Ernesto, Hissam Scott A., Lundell Bjorn, Succi Giancarlo, eds. *OSS*, IFIP, vol. 275: :237-248Springer; 2008.
15. Frazer Lorelle, Lawley Meredith. *Questionnaire Design and Administration: A Practical Guide*. John Wiley & Sons Australia,Ltd; 2000.
16. Zimmermann Thomas. Card-sorting.. In: Menzies Tim, Williams Laurie A., Zimmermann Thomas, eds. *Perspectives on Data Science for Software Engineering*, Academic Press 2016 (pp. 137-141).
17. Otte Evelien, Rousseau Ronald. Social network analysis: a powerful strategy, also for the information sciences. *Journal of Information Science*. 2002;28(6):441–453.
18. Swart J., Henneberg S. C.. Dynamic knowledge nets - the 3C model: exploratory findings and conceptualisation of entrepreneurial knowledge constellations.. *J. Knowledge Management*. 2007;11(6):126-141.
19. Tamburri Damian A., Nitto Elisabetta, Lago Patricia, Vliet Hans. On the Nature of the GSE Organizational Social Structure: an Empirical Study. *proceedings of the 7th IEEE International Conference on Global Software Engineering*. 2012;:114–123.

20. Schlichter Johann, Koch Michael, Xu Chengmao. Awareness The Common Link Between Groupware and Community Support Systems. *Computing*. 1998;:77–93.

21. Mase Yasuyuki Sumi Kenji. Supporting Awareness of Shared Interests and Experiences in Community. In: ; 2000.

22. Krippendorff Klaus. *Content Analysis: An Introduction to Its Methodology (second edition)*. Sage Publications; 2004.

23. Tamburri Damian A., Lago P., Vliet H.. Uncovering Latent Social Communities in Software Development. *Software, IEEE*. 2013;30(1):29-36.

24. Bryson John M., Crosby Barbara C., Bryson John K.. Understanding Strategic Planning and the Formulation and Implementation of Strategic Plans as a Way of Knowing: The Contributions of Actor-Network Theory. *International Public Management Journal*. 2009;12(2).

25. Tamburri Damian A., Kazman Rick, Fahimi Hamed. The Architect's Role in Community Shepherding.. *IEEE Software*. 2016;33(6):70-79.

26. Basili Victor R., Caldiera Gianluigi, Rombach Dieter H.. The Goal Question Metric Approach. Wiley 1994.

27. Fenton Norman E., Pfleeger Shari Lawrence. *Software Metrics: A Rigorous and Practical Approach*. Boston, MA, USA: PWS Publishing Co.; 2nd ed.1998.

28. Vliet Johannes C.. *Software engineering – principles and practice*. Wiley; 1993.

29. Hofstede Gert Jan Hofstede Michael Minkov. *Cultures and Organizations: Software of the Mind*. McGraw-Hill Publishing Company; 2010.

30. Sasovova Zuzana, Tamburri Damian Andrew, Selakovi? Marija, Matthesius Melvin. Leveraging Expertise Diversity. In: Tech. Rep. Available upon request; 2013.

31. Nooteboom Bart, Van Haverbeke Wim, Duysters Geert, Gilsing Victor, Oord Ad. Optimal cognitive distance and absorptive capacity. *Research Policy*. 2007;36(7):1016-1034.

32. Hammersley Martin, Atkinson Paul. *Ethnography*. London: Routledge; 2003.

33. Bird Christian, Pattison David, D'Souza Raissa, Filkov Vladimir, Devanbu Premkumar. Latent social structure in open source projects. In: SIGSOFT '08/FSE-16:24–35ACM; 2008; New York, NY, USA.

34. Almarimi Nuri, Ouni Ali, Mkaouer Mohamed Wiem. Learning to detect community smells in open source software projects.. *Knowl. Based Syst.*. 2020;204:106201.

35. Wohlin Claes, Runeson Per, Höst Martin, Ohlsson Magnus C., Regnell Björn, Wesslén Anders. *Experimentation in software engineering: an introduction*. Norwell, MA, USA: Kluwer Academic Publishers; 2000.

36. Tamburri Damian A., Palomba Fabio, Serebrenik Alexander, Zaidman Andy. Discovering community patterns in open-source: a systematic approach and its evaluation.. *Empirical Software Engineering*. 2019;24(3):1369-1417.

37. Feller Joseph, Fitzgerald Brain, Hoek Andr?. Open Source Software Engineering. *IEE Proceedings - Software*. 2002;149(1):1-2.

38. Joblin Mitchell, Mauerer Wolfgang, Apel Sven, Siegmund Janet, Riehle Dirk. From Developer Networks to Verified Communities: A Fine-Grained Approach.. In: :563-573IEEE; 2015.

39. Bhowmik Tanmay. Stakeholders' social interaction in requirements engineering of open source software.. In: Gorschek Tony, Lutz Robyn R., eds. *RE*, :467-472IEEE Computer Society; 2014.

40. Scacchi Walt. Understanding Requirements for Developing Open Source Software Systems. *IEE Proceedings - Software*. 2002;149(1):24-39.

41. Tamburri Damian Andrew Andrew, Palomba Fabio, Kazman Rick. Exploring community smells in open-source: An automated approach. *IEEE Transactions on software Engineering*. 2019;.

42. Tamburri Damian A., Kruchten Philippe, Lago Patricia, Vliet Hans. What Is Social Debt in Software Engineering?. In: :40-49; 2013.

43. Cataldo Marcelo, Herbsleb James D, Carley Kathleen M. Socio-technical congruence: a framework for assessing the impact of technical and work dependencies on software development productivity. In: :2–11; 2008.

44. Palomba Fabio, Tamburri Damian Andrew. Predicting the emergence of community smells using socio-technical metrics: A machine-learning approach. *Journal of Systems and Software*. 2021;171:110847.

45. Huang Zijie, Shao Zhiqing, Fan Guisheng, et al. Predicting Community Smells' Occurrence on Individual Developers by Sentiments. *arXiv preprint arXiv:2103.07090*. 2021;.

46. Almarimi Nuri, Ouni Ali, Mkaouer Mohamed Wiem. Learning to detect community smells in open source software projects. *Knowledge-Based Systems*. 2020;204:106201.

47. Catolino Gemma, Palomba Fabio, Tamburri Damian A, Serebrenik Alexander, Ferrucci Filomena. Gender diversity and women in software teams: How do they affect community smells?. In: :11–20IEEE; 2019.

48. Catolino Gemma, Palomba Fabio, Tamburri Damian A, Serebrenik Alexander, Ferrucci Filomena. Gender diversity and community smells: insights from the trenches. *IEEE Software*. 2019;37(1):10–16.

49. Catolino Gemma, Palomba Fabio, Tamburri Damian Andrew, Serebrenik Alexander. Understanding Community Smells Variability: A Statistical Approach. In: :77–86IEEE; 2021.

50. Catolino Gemma, Palomba Fabio, Tamburri Damian A, Serebrenik Alexander, Ferrucci Filomena. Refactoring community smells in the wild: the practitioner's field manual. In: :25–34; 2020.

51. Palomba Fabio, Tamburri Damian Andrew Andrew, Fontana Francesca Arcelli, Oliveto Rocco, Zaidman Andy, Serebrenik Alexander. Beyond technical aspects: How do community smells influence the intensity of code smells?. *IEEE transactions on software engineering*. 2018;.

52. Fowler Martin. *Refactoring: improving the design of existing code*. Addison-Wesley Professional; 2018.

53. Zhang Weiqiang, Cheung Shing-Chi, Chen Zhenyu, Zhou Yuming, Luo Bin. File-level socio-technical congruence and its relationship with bug proneness in oss projects. *Journal of Systems and Software*. 2019;156:21–40.

54. Bird Christian, Nagappan Nachiappan, Gall Harald, Murphy Brendan, Devanbu Premkumar. Putting it all together: Using socio-technical networks to predict failures. In: :109–119IEEE; 2009.

55. Di Nucci Dario, Palomba Fabio, De Rosa Giuseppe, Bavota Gabriele, Oliveto Rocco, De Lucia Andrea. A developer centered bug prediction model. *IEEE Transactions on Software Engineering*. 2017;44(1):5–24.

56. Izquierdo-Cortazar Daniel, Gonzalez-Barahona Jesus M, Duenas Santiago, Robles Gregorio. Towards automated quality models for software development communities: The QualOSS and FLOSSMetrics case. In: :364–369IEEE; 2010.

57. Catolino Gemma, Palomba Fabio, De Lucia Andrea, Ferrucci Filomena, Zaidman Andy. Enhancing change prediction models using developer-related factors. *Journal of Systems and software*. 2018;143:14–28.

58. Catolino Gemma, Palomba Fabio, Fontana Francesca Arcelli, De Lucia Andrea, Zaidman Andy, Ferrucci Filomena. Improving change prediction models with code smell-related information. *Empirical Software Engineering*. 2020;25(1):49–95.

59. Qu Yu, Guan Xiaohong, Zheng Qinghua, et al. Exploring community structure of software call graph and its applications in class cohesion measurement. *Journal of Systems and Software*. 2015;108:193–210.

60. Kwan Irwin, Schroter Adrian, Damian Daniela. Does socio-technical congruence have an effect on software build success? a study of coordination in a software project. *IEEE Transactions on Software Engineering*. 2011;37(3):307–324.

61. Zhang Weiqiang, Chen Zhenyu, Luo Bin. Does Socio-Technical Congruence Have an Effect on Continuous Integration Build Failures? An Empirical Study on 10 GitHub Projects. In: :333–343IEEE; 2018.

62. Tamburri Damian Andrew, Blincoe Kelly, Palomba Fabio, Kazman Rick. ?The Canary in the Coal Mine?? A cautionary tale from the decline of SourceForge. *Software: Practice and Experience*. 2020;50(10):1930–1951.

63. Qiu Huilian Sophie, Nolte Alexander, Brown Anita, Serebrenik Alexander, Vasilescu Bogdan. Going farther together: The impact of social capital on sustained participation in open source. In: :688–699IEEE; 2019.

64. De Stefano Manuel, Pecorelli Fabiano, Tamburri Damian A, Palomba Fabio, De Lucia Andrea. Splicing Community Patterns and Smells: A Preliminary Study. In: :703–710; 2020.

65. Manikas Konstantinos, Hansen Klaus Marius. Software ecosystems–A systematic literature review. *Journal of Systems and Software*. 2013;86(5):1294–1306.

66. Hauge Øyvind, Ayala Claudia, Conradi Reidar. Adoption of open source software in software-intensive organizations–A systematic literature review. *Information and Software Technology*. 2010;52(11):1133–1154.

67. Syeed MM Mahbubul, Hammouda Imed, Systä Tarja. Evolution of open source software projects: A systematic literature review.. *JSW*. 2013;8(11):2815–2829.

68. Steinmacher Igor, Silva Marco Aurelio Graciotto, Gerosa Marco Aurelio, Redmiles David F. A systematic literature review on the barriers faced by newcomers to open source software projects. *Information and Software Technology*. 2015;59:67–85.

69. Ghosh Rishab A, Glott Ruediger, Krieger Bernhard, Robles Gregorio. *Free/libre and open source software: Survey and study*. 2002.

70. Tourani Parastou, Adams Bram, Serebrenik Alexander. Code of conduct in open source projects. In: :24–33IEEE; 2017.

71. López-Fernández Luis, Robles Gregorio, Gonzalez-Barahona Jesus M, Herraiz Israel. Applying social network analysis techniques to community-driven libre software projects. *International Journal of Information Technology and Web Engineering (IJITWE)*. 2006;1(3):27–48.

72. Prattico Ludovico. Governance of Open Source Software Foundations: Who Holds the Power?. *Technology Innovation Management Review*. 2012;:37-42.

73. Cheng Li-Te, Hupfer Susanne, Ross Steven, Patterson John. Jazzing up Eclipse with collaborative tools. In: eclipse '03:45–49ACM; 2003; New York, NY, USA.

74. Robles Gregorio, Gonzalez-Barahona Jesus M., Izquierdo-Cortazar Daniel, Herraiz Israel. Tools and Datasets for Mining Libre Software Repositoriesch. 2, :24–42. Hershey, PA: IGI Global 2011.

75. Keivanloo I., Forbes C., Hmood A., et al. A Linked Data platform for mining software repositories. *Mining Software Repositories (MSR), 2012 9th IEEE Working Conference on*. 2012;:32–35.

76. Stojanovic Ljiljana, Ortega Felipe, Dueñas Santiago, Cañas-Díaz Luis. ALERT: Active Support and Real-Time Coordination Based on Event Processing in Open Source Software Development. In: :359–362IEEEIEEE; 2011; Oldenburg, Germany.

How to cite this article: D. A. Tamburri, and F. Palomba (2021), Evolving Software Forges: An Experience Report from Apache Allura, *Journal of Software: Evolution and Process*, *2021;00:1–6*.